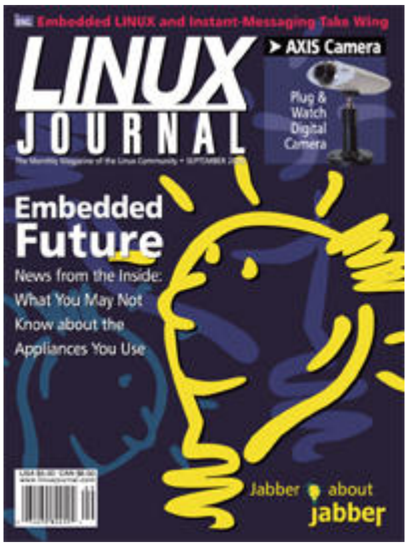
Advanced search

Linux Journal Issue #77/September 2000



The Axis 2100 Network Camera

Jason Schumaker

Issue #77, September 2000

Jason reviews the Axis camera. It has a built in web server and runs Linux.

The Good, The Bad

- Manufacturer: Axis Communications
- E-mail: info@axisinc.com
- Price: $499
- Reviewer: Jason Schumaker

Axis Communications sent *Linux Journal* one of their 2100 network cameras for review. I was lucky enough to spend a week playing with the thing, and I am very impressed. The camera is running Linux (version 2.0.33) and is powered by Axis' own chip—the Etrax-100.

The Camera

Axis bills the 2100 as a “plug and watch digital camera with direct network attach”. Accurate enough. It truly does have an “appliance” feel. A PC is not needed—the camera operates stand-alone, requiring only a LAN or modem connection. It has a built-in web server, which allows you to view images through a browser and throw those images directly out on the Internet (or not). So, you can interact with friends, jazz up your web page, create small movies, monitor your home/office or, ahem, start a business.

The camera weighs, excluding mini-tripod and power supply, a mere half of a pound. There is a feeling of fragility to the thing—one of the few “bad things” about the camera (see Sidebar). Axis does have an industrial strength camera in the works, but no release date has been set. I actually knocked the camera down at one point (please don't tell the folks at Axis!). It fell about two feet and landed on the floor. The images on the screen froze. After two hours and a $500 transfer from savings to checking, I pushed the camera's reset button, which reboots the camera. Everything was fine after that. A heavier, more durable version will certainly be needed, especially for brutes like me.

For $450-$499 you will get the camera, a mini-tripod, one 12V AC adaptor, a null modem cable, a PS-D extension cable, along with the user's guide. The price seems quite fair, considering the myriad of functions it can serve, not to forget the entertainment value. The lens can be replaced with any standard C or CS lens, which I would recommend. The factory lens is fine, but having more lens power is always good.

Installation

The Axis 2100 is itching to be used. The set up is painless and quick, though personal error, lack of certain permissions and little patience on my part added time. Linux/UNIX users may need to be logged in as root, which might not sit too well with some sys admins. I needed to be root in order to use the arp command.

The camera can be installed on a network or to a modem. I went the network route, so will not be providing the modem installation procedures here. That method is a bit more complicated, but after a quick reading, I wouldn't think it to be difficult. Axis has made the installation process easy, unless you're completely unfamiliar with Linux.

It took nearly as long to free the 2100 from the packaging as it did to get it running. There were some “non-camera” related issues to work through. To begin with, plugging the AC adaptor into the outlet really helped. Then, choosing a *working* Ethernet port got me one step closer. And, finally, reading the enclosed “Addendum”, which informed of a possible need to “reinstate the Factory Default settings”, proved instrumental in getting the camera operational. What can I say, excitement got the better of me!

The accompanying “User's Guide” provided installation commands for both Windows and UNIX systems. The first command wasn't quite right for a Linux system, but a quick query to the *LJ* tech department provided clarification. The following commands are what worked for me and are different from those in the 2100 User Guide. To install the camera to my Ethernet network, I typed (in an xterm):

```
  /usr/sbin/arp -s 192.168.3.72 00:40:8c:10:00:86 temp
```

where the IP address was issued from my sys admin and the Ethernet address is the serial number from the bottom of the camera. I then ping'd the IP address:

```
ping 192.168.3.72
```

That's it. I opened a web browser and entered the IP address (i.e., http://192.168.3.72). A page quickly opened and there I was, caught in a side view, bad posture revealed. I was immediately struck by the quality of the image and the speed at which my movements were followed. This camera delivers between 3-10 images/second. Speed can be effected by bandwidth, your computer and browser type, monitor resolution and more. The images produced were close enough to real-time to amaze the entire *LJ* staff. Laurie Tucker, *LJ*'s Special Projects person, described it best by saying, “It looks like a moving painting.” Images are a bit “trippy”, but the quality is one step closer to real-time.

Viewing Images and Configuration

Images can be viewed straight through a customized web page or by uploading them to a remote FTP server. I didn't use the latter, but it can be configured to alert you, via e-mail, when a new image has been uploaded. Setting the camera to upload an image every five minutes could work nicely for home or office security. An adaptor on the back of the camera allows for hook up to devices such as motion detectors and doorbells.

All camera configuration is web-based, by using the “Installation Wizards” and the “Application Tools”. These are presented in GUI form and are used to define the system, set security preferences and tweak image and layout settings. Simply click on either link, which conveniently sit just to the right of the image(s) being displayed. The “Application Tools” allow you to modify anything from image resolution, to color, time, background color, image title and so on. The “Installation Wizards” is nicely organized into the following sections: security, date and time, image settings, focus, modem or network and TCP/IP. You are also given the option of uploading images to an FTP site.

Security and the FTP Daemon

By default, the camera is set to allow anyone access. This can be changed by registering the camera to a single user. That single user is then given the ability to set security permissions to individual users—requiring a user name and password for access. Similar to most Linux systems.

The 2100 has telnet turned off but does run an FTP daemon. To log into the camera, I FTP'd to the IP address of “my” camera's web page, then logged in as root, using “pass” for the password. Doing this allowed me to root through /proc files (see Listings 1 and 2). This is also the method for downloading upgrades to the camera, or uploading code, which is the coolest thing about this camera. Unlike most appliances running embedded Linux, you can go in and change things, if you are so inclined. As Jon Corbet writes (see Resources), “Axis makes available everything that is needed to do this: versions of the compiler and libraries (licensed under the GPL) are at the Axis developer site. It's mostly a matter of cross-compiling the code and FTPing into the camera.”

Listing 1. 2100 Cpu Info

Listing 2. 2100 Memory Info

Hardware and Specifications

There is a Linux port running on the Etrax 100. Axis ported Linux to their products in December 1997 (kernel 2.0.33) This port, along with their developers not wanting to spend time developing a new OS, made Linux an obvious choice to run the 2100 web camera. The source code to the Linux port is licensed under the GPL and can be downloaded from the Axis site. Here's the hardware, as listed in the user's guide:

- ARPTEC-1 compression chip; ETRAX-100
- 32-bit RISC
- 100 MIPS CPU
- 8MB RAM
- 2MB FLASH PROM

The camera specifications, for those of you in the know:

- Digital, 24-bit color
- Image pickup device: 1/4'' inch HAD RGB progressive scan CCD
- HxV: 659x494
- Backlight Compensation
- Automatic AGC
- White balance: automatic, fixed indoor, fixed fluorescent, fixed outdoor and hold
- Electronic shutter: 1/30s - 1/30000s, light condition adaptive

Requirements

High bandwidth is the place to start. If you plan to transfer files from the camera over the Internet, a DSL (or equivalent) connection is the way to go. To view images, Axis recommends a “PC, ideally Pentium 300 or higher, with a high-speed graphics card and 100MB Ethernet network card”. The only software necessary is TCP/IP and either Netscape Navigator (their recommendation) or Internet Explorer. You will also need an IP address, Ethernet cable(s) and a network hub.

A Cool Camera and Company

The 2100 network camera seems solid, as does the company producing it. Axis has been around since 1984. Headquartered in Lund, Sweden, they have more than 500 employees working at 28 offices worldwide. There Etrax-100 chip is a product in its own right and is used in numerous Axis products, ranging from cameras to print servers. Any product that can work with Linux, does. Axis makes source codes available on their developer site. This includes the code to their own Journaling Flash File System (JFFS), which the Axis site says is “aimed at providing a crash/powerdown-safe file system for disk-less embedded devices”. It allows the camera to be turned off and on without having to reboot—providing the necessary “always on” functionality.

I have spent the past week toying around with the camera, and I am still excited about it. I want one. Though it is designed for indoor use only, it does well with that. Images are crisp and quick. This is embedded Linux in action and the results are exciting. It's a Sunday, and I'm at work, playing with the camera. That should speak volumes.

Jason Schumaker (info@linuxjournal.com) is the Assistant Editor and a staff writer at *Linux Journal*. Outside of work he plays left field for the defending Core softball league champion *Monkey Pub Pounders*. Like you care. GO POUNDERS!

Archive Index Issue Table of Contents

Advanced search

The Next Bang: The Explosive Combination of Embedded Linux, XML and Instant Mess

Doc Searls

Issue #77, September 2000

As potential soars for the collaboration between embedded Linux and instant messaging, we are bracing for a deluge of new applications and services. More importantly, we stand on the threshold of a new way to think about—and use—the Internet.

Linux. It's not just for computers any more.

That's the message Linus Himself has been giving us ever since he beat a career path to the Mobile Market when he went to work for Transmeta in 1997. That market is just the computer-like edge of a much larger category of devices that run on “embedded” or “real-time” microprocessors and their operating systems.

Lately, that market has developed a huge appetite for Linux—so huge, in fact, that it's not hard to imagine Linux quickly becoming the commodity embedded OS. Here's why:

- Open source code
- World's largest and most active Open Source development community
- Mature, proven and still evolving rapidly
- Small size—one embedded version fits on a floppy
- Modular and easily customized
- Net-native
- Many development tools
- Free—no runtime royalties required
- A hot topic that gets lots of press coverage

Even though Linux is not a real-time OS (RTOS), changes in the hardware world make it much more desirable in many cases than traditional RTOSes. The traditional RTOS worked in a world where hardware was—by today's standards—slow, large, expensive and specialized. An RTOS had to make the most of many different 4-, 8- and 16-bit microprocessors, low clock speeds and tiny memory footprints in relatively large packages. In addition, an RTOS had to be crafted precisely to an application.

In the RT world, applications were usually in highly isolated markets. Even in the home, just about every potentially “intelligent” real-time application belonged to a very isolated professional specialty that had little if any interest in other specialties. Home security, outdoor irrigation, fire-safety sprinkler systems, heating and air conditioning, kitchen appliances, entertainment systems, indoor and outdoor lighting, telephones, computers and networks...these were all served by different businesses with different kinds of expertise. For proof, try to find a single contractor to install (much less service) telephone, security, computer network and home entertainment systems. The Net was never a factor for many of these categories, so it was pointless to think of all of them in a connected context. Their vendors didn't, even if their customers did.

Now, connected-context is increasingly the point. “People are going to have to think differently here—even people already in the embedded space,” says David Rieves, Director of Product Marketing at Lineo, one of the leading embedded Linux companies. “Now, 32-bit hardware is cheap commodity stuff. So you hold overhead down in the OS, which is why Linux is so appealing. As hardware costs go down, interest in Linux goes up”, especially for devices that live on the Internet.

This means we need to start thinking about real-time development in a network context. While the network context is old hat for Linux, real time is not. The most familiar network services—e-mail and the web—are both essentially store and forward concepts. But to a growing community of Open Source developers, the missing piece here is *instant messaging* (IM), which, by definition, is much closer to “real” time.

IM is a familiar concept to the tens of millions of AOL customers who use AIM (AOL Instant Messenger) or ICQ, which AOL also owns. Neither is open and deployable like Sendmail and Apache for the obvious reason that AOL owns the services and enjoys keeping them to itself. Even if you can create your own clients, the servers are still AOL's. They control it all. As a result, most of the world still understands IM in AOL's terms. This is why a huge percentage of IM usage is what one wag (yours truly) once called, “fourteen-year-old girls cheating on their homework and talking about boys in real time.” Worse, AOL's breed of IM is limited by its business model, which is all about selling captured eyeballs to advertisers.

Once again, geeks are fixing the problem.

As usual, it started with a guy looking for a way to scratch his own itch. The guy in this case was Jeremie Miller. In 1998, Jeremie was working for an ISP when he started to think about IM as a UNIX solution:

> I found that more of the people I knew were fooling around with AIM, and I ended up having a need to have all of my UNIX utilities, all of my logging stuff, a lot of the UNIX environment—wanting to use this new buddy list interactively in real time...as a channel to feed my stuff through. Later on, I saw there were open-source libraries to get back into ICQ and AIM and stuff...and I thought about it for a couple weeks. The question was, “What's the best way to do this?” I didn't want to build my own client, because I can't do GUI stuff at all. I can only do back-end server stuff. I thought “Well, I could build a back-end server, and make my own protocol; maybe I can convince somebody else to write a client, and then they wouldn't have to update it when I want features.” Then I just start adding all the features into the server. Then I can bridge that server to my UNIX things that I want to add in...some logging announcements and any sort of little things I want to push up into it. I can have that server bridge over into pagers. I can have it bridge into ICQ and AIM so I can talk to my friends using other clients. And it just grew from there.

The result was Jabber, the first Open Source instant messaging platform. A cadre of developers quickly gathered around the project—their site is Jabber.org—and built instant messaging that any geek can deploy in the familiar manner of Sendmail and Apache. Here is a description of Jabber's architecture, condensed from the Jabber Technical White Paper :

- Every user interacts through a local server that transfers messages to and through any number of other servers, each with its own domain
- Jabber Identifiers are also expressed like email: yourname@domain.com
- Clients and servers converse among themselves through XML streams. In client/server conversations, the XML stream is always initiated by the client to the server
- The architecture can also support simple clients (e.g., a direct telnet connection) as well as AIM, ICQ and other proprietary clients
- Since it is built on XML, Jabber is extensible and able to express just about any kind of structured data

- Jabber's own protocol consists of XML fragments passed over XML streams between clients and servers. There are three primary protocols that define the basic types of XML fragments used in Jabber: Messages, Presence and Info/Query
- Server-to-server communication involves routing these protocol elements over an XML stream from one server to another (there are no special server-to-server protocols or features)
- A Module API lets the server use external modules to handle message filtering, storage facilities (off-line messages, rosters, user info), user authentication and other functions
- A Service API allows integration of security, special connections for alternate clients and message logging
- Transport servers are used to bridge the Jabber protocol to other services, such as IRC, ICQ and AIM

Jabber is built so wide open that it's hard to see the limits of what can be done with it. But that scope has less to do with IM than with XML (Extensible Markup Language). Again, Jeremie:

> As soon as I found out about XML, I kind of saw it as doing to the IT industry or to the Internet the same thing that silicon chips first did for the computing world—moving from the transistor into ICs. It's almost the same kind of phase. XML feels like that large of a change. It's morphing all the underlying structures, speeding everything up, enabling everybody to structure data in a way that other people can understand. Even if you don't understand some of the pieces of the data, you can at least look at the data and see the structure of that data. So XML was always part of that server I was building in the background. And I was using XML to push it onto the client.

XML was created to package and exchange structured data—files, blocks of text, drawings, transactions, settings, whatever—on its own terms, outside the context of the programs that produce them and (most importantly) outside the API and protocol walls that keep services from interoperating over the Net. XML provides the basic rules for organizing content but not for identifying it. So, while XML resembles HTML in its use of tags, it differs radically from HTML by not forcing a meaning for those tags on the data itself. Those agreements are up to those communicating within an XML framework. As for devices, they can include anything.

XML's advantage is that it's text, plus a structured dynamic framework for what you put in that text. So you can add an XML tag to control anything from an industrial valve to a point-of-sale terminal to a light switch in your home. A novel feature of XML is the DTD, the Document Type Definition. With DTDs, you get to specify document type from inside the document. This allows an infinite variety of tags. The real estate industry, for example, could create a tag called “price.” Everybody in the real estate trade can know what that tag means and program their applications accordingly. Anybody can build a DTD and publish it, which makes the framework especially flexible. It also puts control in the hands of developers working on real world projects and not just standards bodies or large vendors with inflexible market mass.

Table 1. Jabber Architecture Connection Map

T1 = N1 = C3

/

C1 -- S1 - S2 = C4

/

C2 - T2

Table 1 is a map of the typical connections within the Jabber architecture.

o “-” Jabber XML Protocol

o “=” Any other protocol

o C1, C2 - Jabber Clients

o C3 - Client on another IM Network

o C4 - Client using an alternate protocol to access the Jabber Server

o S1 - Jabber Server

o T1 - Transport, translating between the Jabber XML Protocol and the protocol used on another IM Network

o T2 - Transport providing other real-time data to Jabber, such as log notifications or headline news feeds

o N1 - Third-party IM Network

Perry Evans is perhaps best known as the founder of MapQuest. But lately, as President & CEO of Webb, Inc., http://www.webb.net/, he and Webb have taken a greater interest in Jabber than any other company to date. After learning about Jabber, Perry decided to fund its development. Webb now pays Jeremie and other core Jabber.org team members to work full-time on the open-source project. Webb also created Jabber, Inc., http://www.jabber.com/, to explore commercializing Jabber in ways that leverage and reward its open-source development (and developers).

“It is extremely significant that Jabber adopted XML as its transport technique,” Perry says, “effectively allowing conversations to incorporate structure—placing documents, applications and message mining in the middle of a conversation.” He continues, “This plants the seeds of a whole new generation of enterprise, mobile and embedded application possibilities. We use the term “Connected Messaging” to label the way Jabber becomes connected to customer service, business exchange connections, device-based applications—or whatever. Jabber opens up all kinds of possibilities for Net-based conversations between companies and their customers, business partners and mobile employees. Plus the entire world of connected devices with a reason to traffic in live messages.''

For Linux developers, this opens the world in two directions: 1) Linux now operates in, and drives, virtually any kind of device; and 2) protocol-free and dynamic relationships can be created and improved on a constant basis. Of course there are enormous directory and security issues left to solve, and those are just two of the most obvious issues. But now they begin to appear much more solvable.

Craig Burton, the network guru who guided Novell to success in the Eighties and did the same for The Burton Group in the Nineties, sees XML as a development as significant to our time as calculus was to the Renaissance; where calculus gave us a dynamic mathematical model, he says, XML gives us a dynamic protocol framework—one that allows us to circumvent protocol bottlenecks by making communication independent of them. He explains, “XML provides the framework for discovering accessibility—or a protocol—to a service during initialization. Because XML exists, developers can design communications independently of static protocols. With a dynamic protocol framework, independent—or discrete—services can exist and innovate at their own pace without the limitations and boundaries of a fixed protocol. The freedom that a dynamic protocol framework will give to innovation and implementation of Internet infrastructure will have an immeasurable impact on the future of network computing, as we know it.”

Craig's idea of XML as a protocol framework makes IM a potentially critical Internet service for every logical entity with a reason to exchange messages of any kind in “real-enough” time, rather than just for a few million people chatting with their buddies.

Since Linux is in an ideal position to become the universal OS on which “embedded” IM will run, let's take a closer look at where this new industry is headed.

Apply Anything

Real Time and Linux have been acquainted for a long time. RT tasks have been implemented as kernel modules in the manner of device drivers. In the practical sense, RT tasks can be developed and implemented in Linux user space where they cannot overwrite critical areas of kernel memory.

But for real-time precision with minimal interrupt latency, more dedicated solutions need to be found. Victor Yodaiken and Michael Brabanov's approach was embodied in RT-Linux, which was first released in February 1996. They didn't think working along Linux's POSIX lines would deliver that. So they created an emulation layer that insulates the upper layers of the OS from interrupts, intercepting those interrupts on behalf of the OS.

In this scheme, when Linux thinks it turns an interrupt off, it sets a flag in the emulation layer. While interrupts stay on, the emulation layer intercepts the interrupts that occur while Linux believes it is uninterruptible. Intercepted interrupts are queued, so no interrupts are lost. But the real-time work is done by a deterministic adjunct real-time kernel. This division of labor essentially creates a real-time Linux OS—except it's not really Linux where the rubber meets the road.

More recently, another version of that approach has emerged. RTAI (Real-Time Application Interface) also has a separate real-time module that operates below the Linux kernel. Lineo is a prime developer for RTAI, and has contributed a significant portion of the RTAI code base. “Although we provide support for our customers who choose RTLinux, we recommend RTAI because we believe that it has a cleaner design, equivalent performance, and includes enhanced features, such as the ability to deploy a hard real-time application from standard user space ”, says Lineo's David Beal, Product Manager for Real-Time Technology. He adds:

> The advantage of hard real-time over the soft real-time approaches is that when you implement a hard real-time task, you can effectively remain unconcerned with all of the computer's internal and external events. RTAI provides task guarantees because these external are serviced with a lower priority than the real-time task, thus they can not disrupt the critical task's timing. In a world where embedded devices are becoming increasingly network-connected and multi-tasking, this is a luxury that none of the soft real-time approaches can promise.

> RTAI features are modular and need only be added as they are required. Minimal real-time scheduling services take up 65 Kbytes of additional code space but if you add everything in, you get to about 300 Kbytes.

That's just one approach, but it's the one that speaks directly to the inadequacies of Linux in the real-time world. Here's IDC analyst Dan Kusnetzky's overview of the embedded Linux market as it now stands:

> Many organizations are working to find ways to use Linux as an embedded operating environment. Red Hat and Lineo are only the most recognizable names working on this. Some are interested in providing Linux for Internet appliances. Some are focusing on server appliances. A few are focusing on Linux as a real-time operating environment. The last is quite a trick because the Linux kernel doesn't have the features for the most intensive real-time applications.
>
> Some focused on real-time Linux have developed their own real-time kernel to replace the Linux kernel. Others have Linux boot up their own proprietary real-time environment. Although the run time environment isn't Linux, they can talk about real time and Linux in the same sentence because Linux is what people see when it boots up.
>
> IDC is expecting the Linux community to settle on standard approaches to support both embedded and real-time applications. Until then, we have a number of competing offerings from different suppliers.

The competition is heating up fast. James Ready, the embedded pioneer who founded Ready Systems a few years back, now runs MontaVista, http://www.mvista.com/, which launched HardHat Linux last year. Red Hat acquired Cygnus last Fall, then added WireSpeed in June of this year. (The sidebar is by Michael Tiemann, the Red Hat CTO who founded Cygnus.) Last summer, Caldera, an early Linux distribution leader, spun off Lineo as an embedded Linux company. Both Lineo and MontaVista have substantial venture capital backing and are themselves in aggressive acquisition modes. With all this movement in a short time, traditional embedded publications and events are suddenly abuzz with Linux coverage.

However, the traditional RTOS isn't going away quickly. Wind River (which makes both VxWorks and pSOS) is the market leader, with around $200 million in sales and a multi-billion-dollar market cap. “Wind River is the 2000-pound gorilla in the embedded OS market,” says Bryan Sparks, President & CEO of Lineo. “But we're finding a lot of interest in moving away from Wind River products. If we can support the same platforms that Wind River does, we can take more and more business away from them," he explains. In addition to Intel, AMD, PowerPC and other usual suspects, microprocessor platforms include Hitachi SH, MIPS, ARM and StrongArm, ndcore, ColdFire, Dragonball and others. "We can get Linux on all those platforms," Sparks adds.

Some of those platforms are very small micro-controllers that don't have full-featured CPU sets. Typically they lack an MMU (Memory Management Unit). So Linux needs to be modified accodingly.

Why Embedded Linux?

> A happy consequence of the Open Source model is that Linux evolved, by necessity, to be more modular and to support cleaner interfaces than a monolithic design that assumes less anarchistic control. As a result, Linux kernels as small as 300KB can not only boot up with networking support but serve web pages as well. This is really exciting to people who want to put web servers (and administrative consoles) *everywhere*.
>
> You can also use the same APIs for development and deployment. While this is not a new goal, we now have a system that (1) works, (2) people actually like to program and (3) supports tons of content. This has expanded the realm of embedded system development from a highly specialized art to something any Linux programmer can manage. This is going to have a profound impact on the embedded system market.

—Michael Tiemann, Chief Technology Officer, Red Hat

The move to Linux is happening most rapidly around mobile systems. Intel is seeing strong Linux support for StrongARM. Compaq has created the Open Handheld Program and released a Linux version for its iPAQ handheld PC. The company also sponsors Handhelds.org, which supports open-source software development for handheld devices. Given Linux's popularity as a hermit crab on x86 boxes, Windows CE handheld devices are an obvious embedded Linux target. While Palm charges more for a software license (around $20/unit) than Microsoft charges for Windows CE (around $10-$15/unit), Palm's units are more arcane and less powerful than typical Windows CE devices, so moving Linux to the latter is a shorter step. This reportedly is HP's approach. Boris Elisman, worldwide marketing manager at HP's information appliances and services division, says the company is developing a number of multifunction Linux devices that should come out in 2001.

Designing embedded systems around Linux is an extremely easy choice for OEMs to make. John Bork of Intel, which is building its new set-top boxes and

Dot.Station web appliances on a combination of embedded Linux and customized Mozilla explains, “We just take the source off the Net, pull out the stuff we don't need, seeing what libraries are used and not used, and constraining ourselves from there. See, the biggest challenge we face is getting the OS, the browser, the plug-ins and a few applets into eight megabytes of flash. So a lot of our work has gone into analyzing what pieces we need and how we skinny this thing down.”

This work has a lot of appeal. “I enjoyed working on servers at my last job,” says one Linux programmer at Kerbango, a startup using embedded Linux in its web radios, “but I love working on cool new stuff here.” In the past, learning a whole new operating system and all its tools was expensive and time-consuming. Not so with Linux. An embedded application might be extremely arcane, but the operating system knowledge required is not, because Linux and its tools are so familiar to so many developers. This fact can hold down NRE (non-recoverable engineering) costs, which are always critical factors.

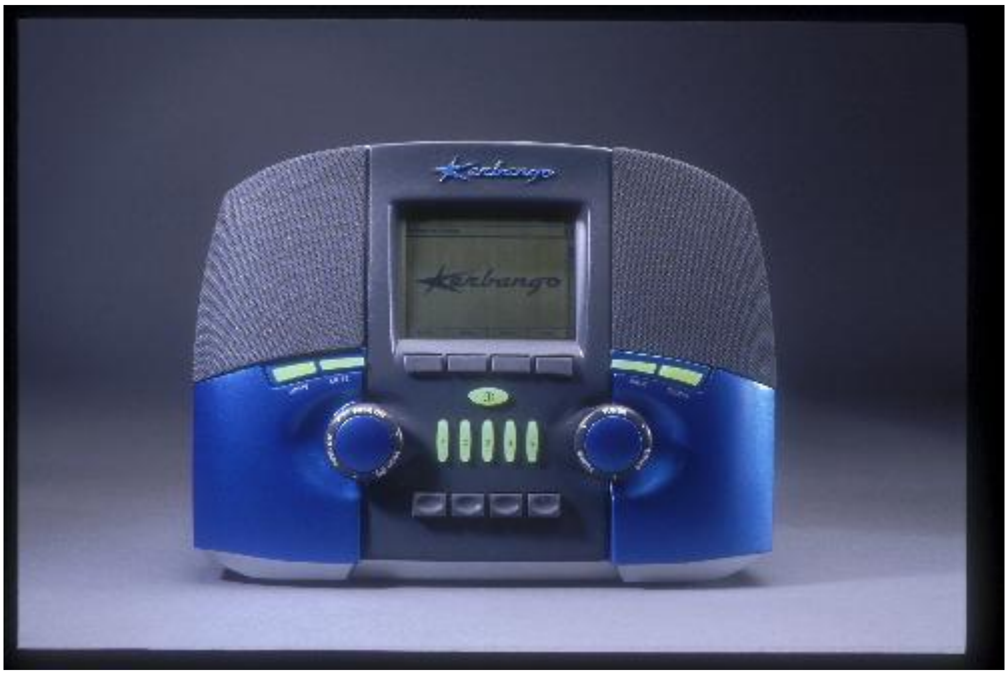


Figure 1. Embedded Linux Appliances

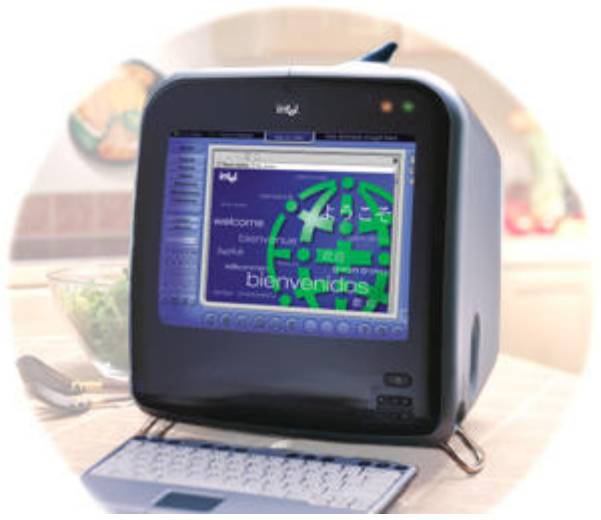


Figure 2. Intel's Dot.Station

Kerbango's radio and Intel's Dot.Station are just two among the first wave of Net-native appliances built using embedded Linux (the Dot.Station also uses Mozilla).

“Skinnying down” Linux, however, is not an easy job. That's why companies like Lineo and MontaVista are in business. Here's Kim Clark, VP Engineering at Lineo is familiar with these issues and says, “The first thing you need to understand is, what are all the packages? Which ones do I need and which ones don't I need? What are the dependencies associated with each of them? Choosing one package may require choosing two or three other packages. But they also might conflict with packages from another module. So you have this hairy and complicated task of figuring out all the things that need to be there.

Among Lineo's tools is one called “LIPO,” which goes in and, well, sucks out all the fat. “The shared libraries are big and fat,” Clark says. “If you have several modules sharing libraries, each with a copy of that library, you've got a bigger footprint. We cut fat out of that. We also reduce the size of each library's footprint. We do the same for Apache. By cutting out icon support, for example, you save 22k of memory. We make this possible by taking Apache, cutting it into microcomponents, giving developers the choice of using only the parts they need.” (See Figures 3 and 4.)

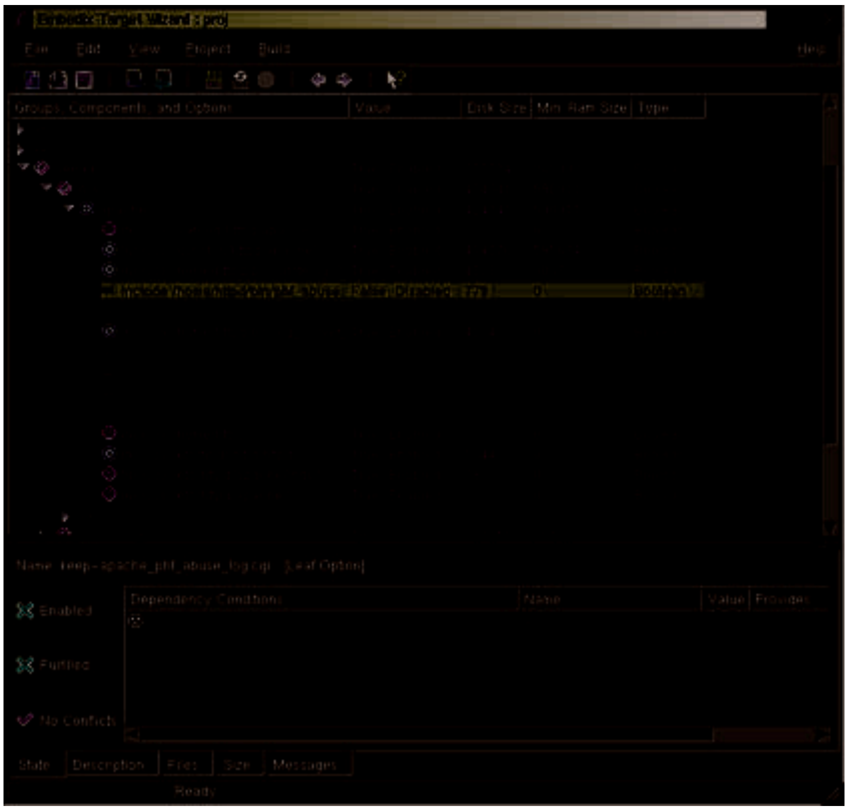
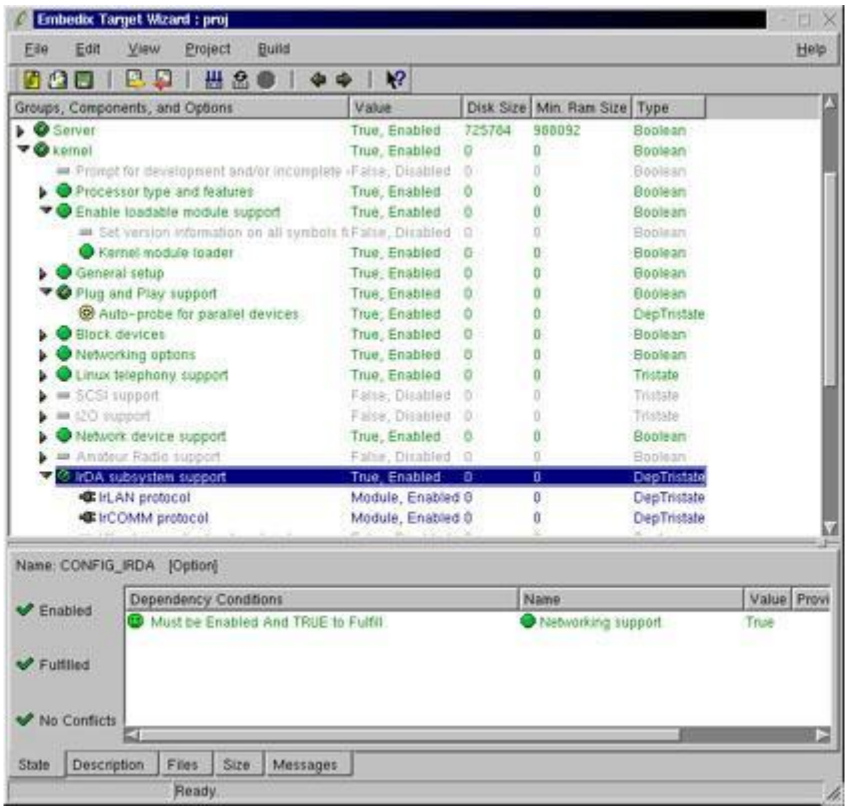


Figure 3. Lineo's Embedix Target Wizard Selection and Configuration Interface



Groups, Components, and Options	Value	Disk Size	Min. Ram Size	Type
Server	True, Enabled	725764	988092	Boolean
kernel	True, Enabled	0	0	Boolean
Prompt for development and/or incomplete	False, Disabled	0	0	Boolean
Processor type and features	True, Enabled	0	0	Boolean
Enable loadable module support	True, Enabled	0	0	Boolean
Set version information on all symbols f	False, Disabled	0	0	Boolean
Kernel module loader	True, Enabled	0	0	Boolean
General setup	True, Enabled	0	0	Boolean
Plug and Play support	True, Enabled	0	0	Boolean
Auto-probe for parallel devices	True, Enabled	0	0	DepTristate
Block devices	True, Enabled	0	0	Boolean
Networking options	True, Enabled	0	0	Boolean
Linux telephony support	True, Enabled	0	0	Tristate
SCSI support	False, Disabled	0	0	Tristate
I2O support	False, Disabled	0	0	Tristate
Network device support	True, Enabled	0	0	Boolean
Amateur Radio support	False, Disabled	0	0	Boolean
IrDA subsystem support	True, Enabled	0	0	DepTristate
IrLAN protocol	Module, Enabled	0	0	DepTristate
IrCOMM protocol	Module, Enabled	0	0	DepTristate

Name: CONFIG_IRDA [Option]

✓ Enabled

✓ Fulfilled

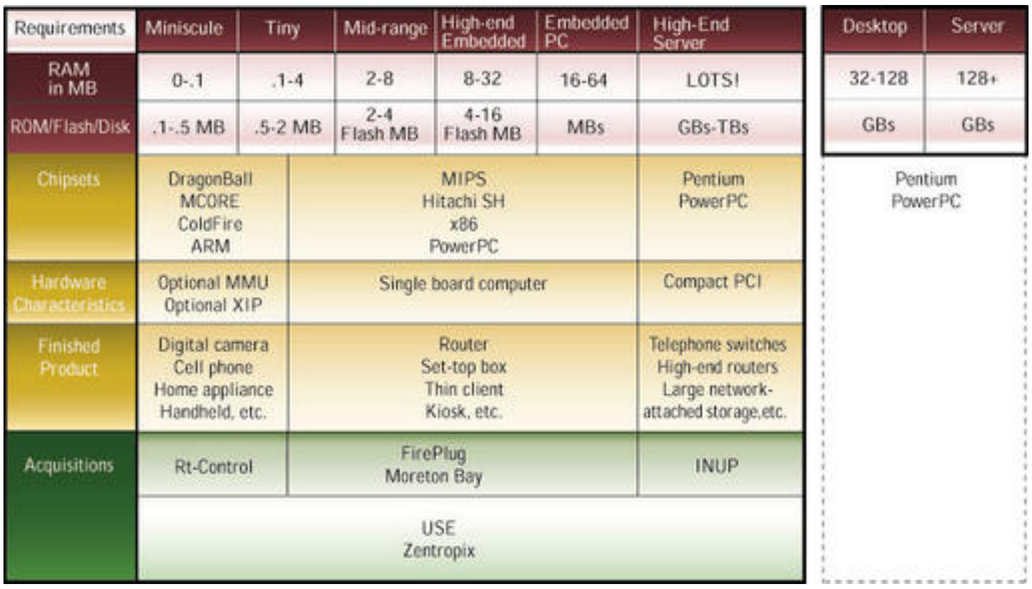
✓ No Conflicts

Dependency Conditions	Name	Value	Provi
Must be Enabled And TRUE to Fulfill	Networking support	True	

Figure 4. Embedix Target Wizard Screen Shot

These screen shots show Lineo's tools at work, selectively enabling parts of both Linux and Apache.

Using tools like Lineo's helps Linux apply across a huge assortment of embedded devices, as we see in Figure 5.



Requirements	Miniscule	Tiny	Mid-range	High-end Embedded	Embedded PC	High-End Server	Desktop	Server
RAM in MB	0-.1	.1-4	2-8	8-32	16-64	LOTS!	32-128	128+
ROM/Flash/Disk	.1-.5 MB	.5-2 MB	2-4 Flash MB	4-16 Flash MB	MBs	GBs-TBs	GBs	GBs
Chipsets	DragonBall MCORE ColdFire ARM		MIPS Hitachi SH x86 PowerPC			Pentium PowerPC	Pentium PowerPC	
Hardware Characteristics	Optional MMU Optional XIP		Single board computer			Compact PCI		
Finished Product	Digital camera Cell phone Home appliance Handheld, etc.		Router Set-top box Thin client Kiosk, etc.			Telephone switches High-end routers Large network-attached storage,etc.		
Acquisitions	Rt-Control		FirePlug Moreton Bay			INUP		
	USE Zentropix							

Figure 5. Lineo's Embedded Linux Market Spectrum

This chart differentiates between the space where the company's embedded product offerings (and acquisitions) apply and the non-embedded PC/Server space, where they don't.

This is Lineo's view of the world, but it could be anybody's. None of the lines are definitive, which is why the boxes are shaded.

Lyle Ball, VP Marketing for Lineo, says, “The important thing is that, while the desktop and server areas are familiar territory, the rest of the world—the embedded device world—is not. There is a lot of existing expertise everywhere on this chart, but when you get down to the finished product level, that's where you've got the feudal system that Linux and these other developments begin to eliminate.”

Real World Real-Time

Behind every cell in Figure 5 is the Net. And behind the Net's popularity is the growing expectation that just about everything will eventually be connected to it and communications over it will be instantaneous. This is the real world meaning of “real-time.” In a fully distributed computing world, any device should be able to communicate with any other device in real-time—or close enough. What's the infrastructure for that?

The Jabber People think it's XML-based Instant Messaging. IM by itself is about two functions: presence and messaging. The first involves detecting and revealing the availability of another identity on the network. The second involves one-to-one (or one-to-few) live text exchanges with one or more present members of a list.

In enterprises where IM is widely adopted, life isn't the same for other technologies. “It changes the sociology of telephony,” says Udi Shapiro, an instant messaging pioneer whose company, Ubique, created an IM system that was owned for awhile by AOL before Udi and his partners bought it back and sold it to Lotus, which now offers it as SameTime. According to Udi, IM in corporate settings becomes an accessory to the telephone, replacing brief or trivial calls in some cases and initiating them in others. The next step would naturally be to meld IM and telephony technologies: in wired telephone systems; in Voice over IP (VoIP); in wireless communications between handheld devices; and in combinations of all those, plus web, e-mail and other Net-native communications.

It's not hard to imagine what IM can do for communications between humans. The harder but necessary step is to imagine communications between humans and remote machines, and among machines themselves, on an ad hoc, real-time basis. This is where embedded Linux comes in.

When IM is combined with Embedded Linux, IM provides the framework for registration and lookup, while embedded Linux works as the device driver in the API. Through the instant messenger, the client says to the server “I'm on-line,” and the server notifies other clients of that fact, providing the path for direct communications. Thus, instant messaging becomes an ideal way to set up point-to-point communications between two unknown entities—human or otherwise.

With Jabber, XML messages become an “envelope” or container (see Figure 6). The contents of that envelope are infinitely flexible. XML's data is “structured”, but also potentially dynamic—able to change constantly. It can even define the terms by which it is understood.

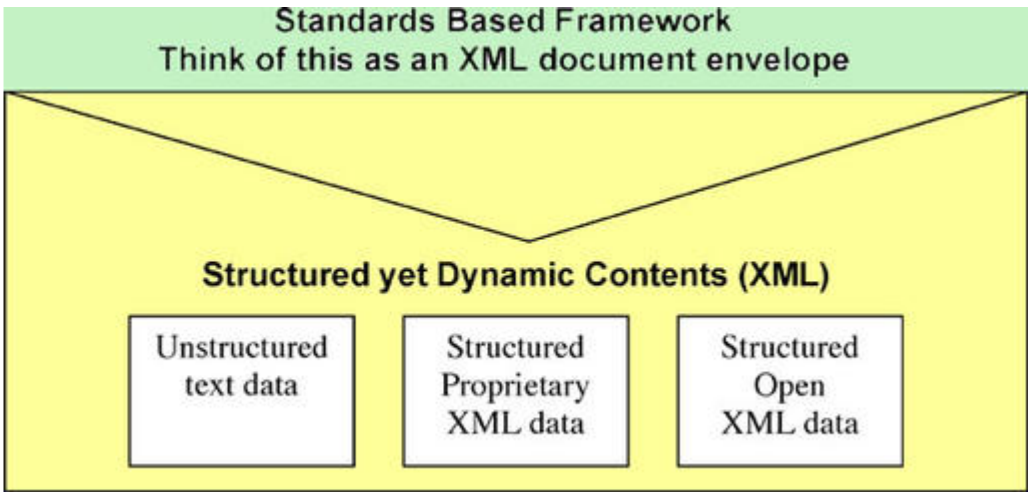

Standards Based Framework
Think of this as an XML document envelope
Structured yet Dynamic Contents (XML)
Unstructured text data
Structured Proprietary XML data
Structured Open XML data

Figure 6. One Jabber View of XBIM—XML Based Instant Messaging: XML as a Container

XML is designed to carry “structured” data. But that data can change constantly and—by design—define the terms by which it is understood. In essence, this creates a standards-based framework for dynamic relationships between communicating entities on the Net. When those communications become “instant”, there is nothing to stop relationships from being defined and changed whenever necessary.

When communications become “instant” (which in practical terms means near real-time), there is nothing to stop relationships from being defined and changed whenever necessary by agreement between the communicating parties, even if those parties are intelligent devices. With Jabber, any two identities, whether human or machine, can send and receive real-time messages that contain pretty much anything, and do so in a structured way, independent, if necessary, of intermediating protocols.

Although Jabber was originally envisioned as a distributed, XML aware instant messaging platform, its uncomplicated methods of combining presence with XML document routing puts it in an ideal position to become much more than just another instant messaging system. Instead, it could become infrastructure for messaging enabled, embedded applications. It becomes the way they communicate through the Net—whether with each other or with applications on the PC (which are already talking to each other).

Figure 7 shows a Jabber client/server platform built for the purpose of routing XML data in real time. Note that the underlying messaging infrastructure and protocol is XML and not a proprietary messaging protocol. Messages can be sent from device-to-device, device-to-PC, or translated from any of the above to other networks and protocols. The dynamic, self-determining nature of XML makes this infrastructure dynamic: in other words, to accommodate constantly changing relationships between logical entities on the Net.

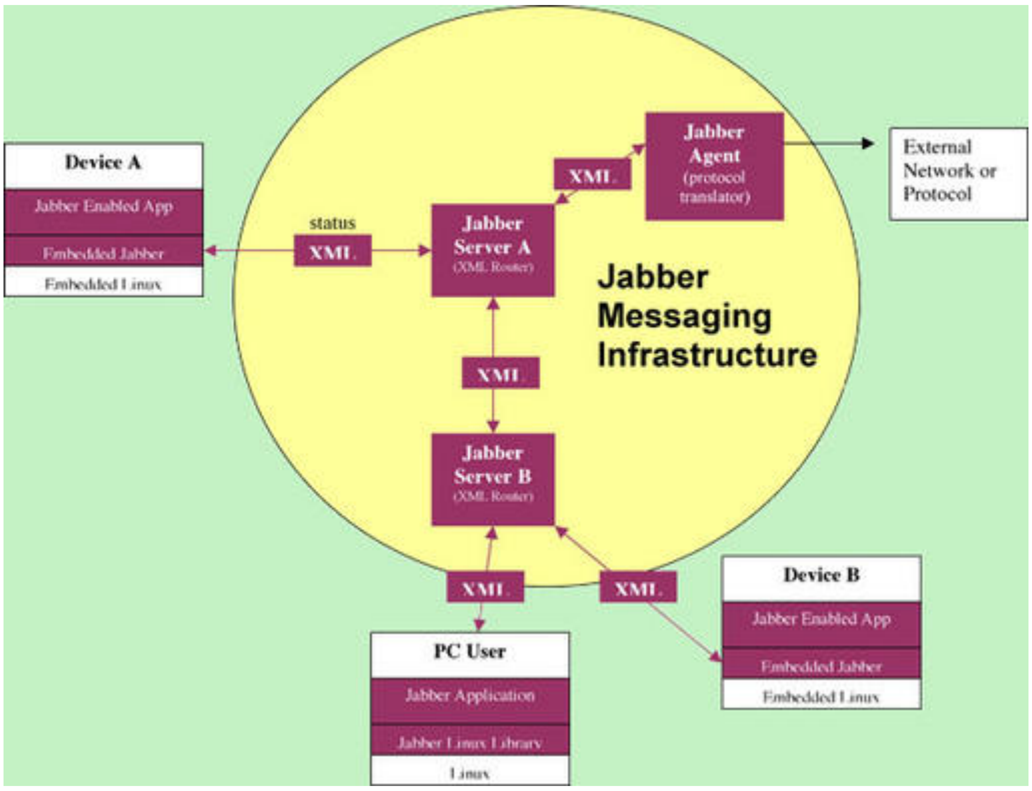


Figure 7. Jabber as a Real-Time Messaging Platform

Jabber is an instant messaging platform with the ability to packetize messages in XML documents—in effect as data transport. Any PC or Net-native device can instantly communicate with all of the infrastructure on the Internet that can parse and act on XML. This puts new XML databases—which are designed to parse XML and store structured data or documents—in a powerful context. Anywhere XML is the preferred way to receive and send information, we have infrastructure already conformed to instant messaging.

Jeremie's Big Picture

In this sense, Jabber is a ubiquitous XML router, streamlined for (and by) embedded Linux, or any other device-driving OS that supports XML-based messaging. Real (enough) time is the key. While e-mail is technically capable of routing XML documents, it lacks awareness of presence, which is key to Instant Messaging in general. This makes Jabber something of a new message transport breed.

This breed will expand Internet infrastructure to accommodate a vast new range of dynamic real-time relationships and activities, routed through XML. This invites the development of suites of robust client-side libraries on multiple OSs, gathering around embedded Linux. These will constitute an SDK for connected messaging to the device world.

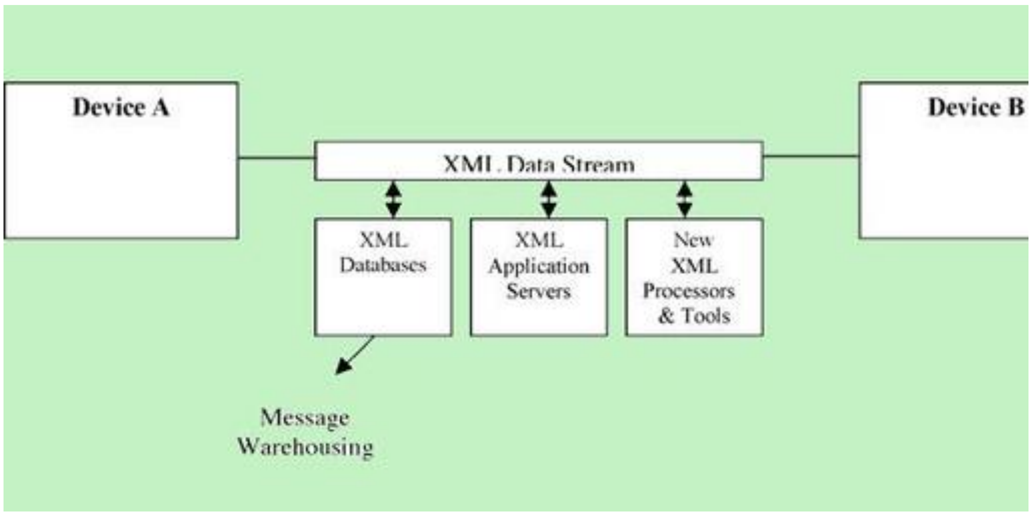


Figure 8. XML developments in the Context of Device Relations across

an XML Data Stream

This shows how new and emerging tools (like XML optimized databases) can make use of (and be used by) XML communications streams between devices, and how they can provide new services. The fact that all communication with Jabber is via XML documents means that all new XML tools and services can be easily “plugged” into the message stream carried by Jabber.

“Emerging tools (like XML optimized databases—see Figure 8) can be used to leverage XML communications streams and to provide whole new services,” says Andre Durand, General Manager of Jabber, Inc., http://www.jabber.com/, the organization formed to help commercialize Jabber in ways that leverage and reward its open-source development (and developers). He continues, “The fact that all communication with Jabber is via XML documents means that emerging XML tools and services can be easily “plugged” into the message stream carried by Jabber. Extrapolate that, and you've got a device driving framework for the known world. Because it is now quite simple for you, or a contractor you hire, to program your home—say through X10, which was designed for that purpose—you can operate, and receive notification about, all the home electronic conveniences you already have: your security system, your lighting, your sprinklers. You can also program those devices to tell you if they're on, functioning and ready to receive instructions. So the concept of presence applies in more than one direction. Two way conversation that includes presence is Jabber infrastructure on a common XML framework.''

Big Bang 2.0

What Jabber does with IM, XML and embedded Linux adds huge new conversational territory to all three topics. Inside that territory is—whatever we want to blow up.

Maybe we're wrong, but we think this is where The Next Big Bang is going to happen. The last time we felt this way, “we” consisted of Phil Hughes, and the spot we watched was Linux. That was in 1994, when both Linux and *Linux Journal* were pre-1.0.

Look what's happened since. If we're right, this is even bigger.

Bringing it Home: Jabber's Jeremie on Instant Messaging and Embedded Linux

Doc Searls (info@linuxjournal.com) is Senior Editor of *Linux Journal* and a member of the Jabber.com advisory board.

Interview with Google's Sergey Brin

Jason Schumaker

Issue #77, September 2000

Learn how Linux came to power one of the best search engines available on the Web.

Google, http://www.google.com/, is the hottest search engine being used on the Internet today (see Doc Searls' “Google Gains...”article, page 10). It's fast and consistently returns relevant links. The company was founded in 1998 by Sergey Brin and Larry Page. The two collaborated on a new search engine technology called PageRank™. Since then, Google has gathered quite a following—Yahoo! recently hired Google to power its search engine—and now boasts the ability to link to over 1 billion URLs. I talked briefly with Sergey Brin, Google's co-founder and President.

Jason: What led to Google's decision to use Linux? When did that start?

Sergey: Well, Larry Page and I were in the Stanford PhD program in Computer Science. And we developed Google there. The way the computer science program worked is there was a hodgepodge of computer equipment lying around, and we would grab whatever scraps we could. We had all kinds of computers: HPs, Suns, Alphas and Intel's running Linux. So, we gained a lot of experience with all of those platforms.

When we started Google, we had to make the decision of what we wanted to use. Of course we chose Linux, because it is the most cost effective solution.

PCs are not only much cheaper these days, but we can also get them very quickly, because they're such a commodity item. That's an incredible benefit. We just installed another 1,000 computers and we got that done in a few weeks. That's really hard to do with any other kind of workstation. I think that's an advantage that people don't entirely realize.

Jason: Did you view it as being better, or was cost the main reason?

Sergey: It was better in some ways. Certainly for our purposes, we felt the support was better. For example, the actual kernel authors will respond to problems pretty quickly. They are especially responsive to Google nowadays, since we're so widely used. We can have a 15 minute turnaround. You can't really beat that for support.

That was an important factor, but frankly, the cost was a bigger issue. PCs are so cheap, which is very important. Sun's Solaris is probably more stable than Linux on PCs. It's hard to determine the blame, whether it's the hardware or the operating system. But, it's a minor difference.

Jason: Then, does all of your support come from newsgroups or do you actually pay for it through Red Hat?

Sergey: We have an operations team of about ten people, which helps a lot. And other than that we check newsgroups and e-mail the authors of the code. Usually, if it's a problem we can't figure out, we go straight to the authors.

Jason: Is Linux used on desktops at Google?

Sergey: It depends. Engineering mostly runs Linux. Business development/ marketing runs Windows. Actually, I use Linux with VMWare running Windows. Some people have two computers, particularly some people in engineering who do UI development and need to test things out on Windows platforms. I find it better to just use VmWare and have one computer.

Jason: In a technical sense, what does Linux lack? What does it not provide?

Sergey: The 64-bit file system, which I know they are working on. It's slowly coming around. I think there are still occasionally some stability issues. I'm not saying Linux is unique in that respect, but you definitely want to have reliability. There are some issues dealing with higher memory systems. If you get to 2GB, and you try to push it past that, we encounter various problems. I know we've had some trouble with the network stack when we really push it hard. In terms of having lost most connections from lots of different machines.

Jason: Well, you're getting quite a few hits per day, aren't you?

Sergey: Yes, we are. We do about ten millions searches per day at Google.com. And another six million or so from OEM customers. So, we get a lot of hits. And when we crawl the Web, we crawl it pretty quickly, which can really stress the system.

Jason: Has your system been down entirely?

Sergey: No, but we certainly have individual computers go down. Our system has a lot of redundancy built into it, so the users don't see it from the outside.

Jason: I've read that you have developed your own network installation tools ...

Sergey: Yeah. We've re-used various components of things that people have built; we've had to now re-do them quite a bit ourselves. We have 5,000 computers now, and that's actually a fair amount of work to install. So we have our own network install system—where we can bring up 80 computers at a time. And we have our own testing software and monitoring tools to keep track of what the computers are doing, what state they're in. So, we've had to do a fair amount of development.

Jason: Of the 5,000 computers used by Google, can you roughly breakdown what they are used for, i.e., 3000 perform searches, 1000 do OEMs, 500 do web crawling, etc.?

Sergey: Without giving specific numbers, we can say approximately 80% of the machines are used for performing searches (google.com and partners); about 10% of the machines are used for Research and Development and another 10% of the machines are used for preproduction (crawling and indexing the web).

Jason: Are the tools worth releasing to the Open Source community?

Sergey: That's an interesting question. I mean, I don't know of too many installations that are of comparable size to ours, but it certainly is, now that you mention it, something we would consider. I don't think that any of them are robust enough or clean enough at this point and time. But, I think we can get them to that state if other people would take over the maintenance and contribute. I just don't think that there are too many people who would end up using them.

Jason: Could you briefly tell us something about yourself and how you came to work at Google?

Sergey: I was born in Moscow and came to the United States at the age of six. I grew up in Maryland, then went to the computer science program at Stanford. I started there in 1993, where I worked on data mining, which basically involves analyzing vast amounts of data to find interesting correlations and patterns. Then Larry joined in 1995. He started downloading the Web and we analyzed its link structure. We've worked together from then on.

Jason: Well, thanks so much for your time. Take care.

Sergey: Thank you.

Jason Schumaker (info@linuxjournal.com) has worked for *Linux Journal* for nearly two years. He is Assistant Editor and a staff writer.

Archive Index Issue Table of Contents

Advanced search

VoIP and Embedded Linux

LJ Staff

Issue #77, September 2000

Aplio offers an alternative to MS-Windows-based long-distance telephone calls on the Internet. Under the hood, you will find embedded Linux.

Aplio offers an alternative to MS-Windows-based long-distance telephone calls on the Internet. Under the hood, you will find embedded Linux.

Telecommunications is a huge market that continues to grow at a frightening pace. As the market becomes more sophisticated, the need for intelligent devices grows as well. A relatively new piece of the telecommunications market is voice over ID, or VoIP. This is where speech is digitized and sent over an IP-based network.

The most common products in this market combine an MS-Windows PC and sound card with proprietary software to offer the equivalent of free long-distance telephone calls by using the Internet.

This approach has its drawbacks. You must have an MS-Windows PC turned on and available in order to place or receive phone calls. Enter Aplio or, more specifically, Aplio/Pro, a stand-alone appliance that allows you to make telephone calls over the Internet.

Like any other VoIP solution, you need a system at each end to make the call. Unlike most solutions, with Aplio you don't need computers on each end. While Aplio has had products in this market, Aplio/Pro amounts to their transition to open-source software.

Henri Tebeka, chief technology officer for Aplio, said, “Linux is the ideal operating system for our technology. Its built-in Internet connectivity, royalty-free accessibility and open-source structure will allow us to streamline our development process, and ultimately strengthen our position as the leading provider of Internet telephony appliance technology.” Combining Linux with the growing improvements in VoIP technology, Aplio brings a progressive development model to bear on one of the most vital areas of the Internet's growth as a truly multimedia, mass network capable of delivering voice data as seamlessly as it does text, still graphics and video.

Aplio/Pro features a built-in processor along with a modem, flash RAM, hardware-based full-duplex voice compression and a full-duplex speakerphone.

Jerome Calvo, president and CEO of Aplio, Inc. said, “Are users going to want to talk over the phone through their computer? With a headset? The approach we have is a non-PC approach, an Internet appliance approach, a stand-alone device...The voice over IP market is much more mature than it was one or two years ago...Still, some companies have since joined the VoIP market with software-based solutions that often don't deliver the same level of quality.”

Setting up Linux-based Aplio/Pro is easy. It connects to a telephone as simply as an answering machine and plugs into an Ethernet port.

Configuring the Aplio/Pro involves little more than entering Internet account information on the telephone keypad. There is a ten-digit number (the Aplio ID, located under the unit) which represents the ID or phone number of the Aplio unit. It can then send calls to other Aplio systems, as well as to those using multimedia computers with Internet telephony software such as Microsoft's NetMeeting.

To place a call, you pick up the telephone receiver (assuming the speakerphone is not being used), press the “Aplio” button on the unit's console, dial the Aplio ID of the individual being called and wait for the phone to ring on the other end of the line. Calling to or from an Aplio unit with an Internet telephony-enabled PC is just as easy.

To use Aplio behind a firewall or NAT (Network Address Translation) box, all that's necessary is to open up the appropriate ports much like you would do with ICQ or other real-time transfer protocols.

Evolution of the Aplio Product Line

How does Linux fit into Aplio's picture? Fresh on the heels of its Internet telephony appliance success, Aplio embarked upon a project to develop a broadband version for the enterprise. Whereas the Aplio/Phone works over average PSTN (public switched telephone network) lines and runs the proprietary PSOS real-time operating system developed by Integrated Systems, Inc. (now a part of Wind River Systems, Inc.), Aplio/Pro is a Linux-based, stand-alone, Internet telephony appliance that routes to the Internet. Aplio/Pro, like the Aplio/Phone, has won its share of commendations, namely Product of the Year awards from both *Internet Telephony* and *Communications Solutions* magazines earlier this year. *Internet Telephony* said, “in our opinion, the best VoIP appliance is Aplio/Pro...” And while acknowledging the value of some competing products from Innomedia, InterStar and Komodo, the review concluded that “the Aplio/Pro is the best choice for ease of setup, economy of scale and overall usability.”

Aplio/Pro's operating system is Aplio's embedded port of the Linux kernel. The port is adapted from ucLinux/ARM (which is itself a port of uClinux, a Linux version for micro-controllers without a memory management unit or mmu). Aplio's TRIO chip was the first port of ucLinux/ARM. The ucLinux kernel was built by taking the standard 2.0.38 kernel and applying the ARM patches and the ucLinux patches. After some “manual editing”, ucLinux for ARM7TDMI was ready to roll.

Said Vadim Lebedev, chief software engineer for Aplio and the one who led the porting of ucLinux to ARM7TDMI, “the porting of the kernel itself was not especially difficult, the biggest problem was adapting the GCC/ARM compiler to generate position-independent code for user-mode application.” Lebedev first got involved with Linux during the development of the Aplio/Pro VoIP appliance and noted that the decision to use ucLinux for the Aplio/Pro (and for future TRIO chip-based products) was due to three primary reasons. For one, the availability of the source code and access to open-source software tools made modifications and enhancements easier and more efficient. Second, the ability to “comfortably” develop and test the code on Linux workstations, and then embed the code on the device by recompiling, was deemed a major advantage. And third, the high level of support Lebedev and his development team could get from the broader community of Linux developers meant that the project would not be isolated from further innovations in both Linux and embedded systems. Added Lebedev, “[now] we can say it is really much easier to develop Linux-based embedded software than, let's say, PSOS-based software...I do believe that Linux has a pretty bright future in the embedded world—even if [Linux] was not initially meant to be used in this way.”

The newly developed Aplio/TRIO chip provides the connection to the Internet (by way of an internal Ethernet port) while at the same time providing real-time voice compression and echo cancellation. The Aplio/TRIO is based around an ARM7TDMI micro-controller core running at 20MHz and a pair of DSP co-processors running at 40MHz. In a general, stand-alone environment such as the Aplio/Phone, one of the DSP co-processors manages voice processing, including voice compression and echo cancellation, while the other co-processor handles the telephony functions such as dialing, caller ID detection, etc.

Wherefore VoIP? Wherefore Aplio?

Aplio and the VoIP Marketplace

According to statistics from aplio.com, the market for VoIP/Internet telephony is expected to explode. One research company, Probe Research Inc., suggests that the total traffic for Internet telephony will reach or exceed 250 billion conversation minutes by 2005. Aplio recorded its five millionth minute in June 1999.

Calvo notes that 70% of those who use the Aplio/Phone and Aplio/Pro are what he calls members of “ethnic communities” or those “global citizens” who work in one country yet often have family or other strong ties to another country. These core Aplio/Phone users would otherwise face heavy long-distance charges without a solution such as VoIP. Similarly, Calvo estimates that the remaining 30% of Aplio's customer base consists of internationally oriented businesses, who are also looking to avoid long-distance tolls while conducting business around the world.

Asked to characterize the Aplio premium on VoIP Internet appliances, Calvo offered the expected: (1) Aplio/Phone and Aplio/Pro are non-PC products; (2) both appliances have “extremely good” voice quality; and (3) both Aplio/Phone and Aplio/Pro feature easy-to-manage user interfaces. “Our goal is to have many companies build[ing] many different appliances. In that sense, betting on an open-source operating system like Linux is a good thing, because we can very quickly prototype, develop and test our products and get them to market.”

Archive Index Issue Table of Contents

Advanced search

An Interview with Inder Singh

Jason Schumaker

Don Marti

Issue #77, September 2000

An interview with the CEO of LynuxWorks—a company that relies heavily on the success of embedded Linux.

I first met Dr. Singh in New York City, while attending LinuxWorld. He was quite cordial, soft-spoken and informed. His passion for his work was apparent. We talked, over breakfast, about the future of embedded Linux, the release of BlueCat Linux 1.0 and Linux in general. Dr. Singh is the CEO and Chairman of LynuxWorks, a company focused on providing embedded applications to OEMs and beyond. I spoke with him in early July about the current and future state of the embedded Linux world.

LJ: What made you choose Linux? What tools/developments would like to see added to Linux?

Inder: At LynuxWorks (formerly Lynx Real-Time Systems) we have always been a strong proponent of open standards in the field of real-time and embedded systems. We started out with implementing a real-time operating system (RTOS) from the ground up, LynxOS, which was highly compatible with UNIX (just as Linux is today), and we have played a pioneering role in the POSIX standardization effort. We were the first to implement the POSIX real-time extensions and POSIX threads, which are widely implemented in most UNIX systems today, and LynxOS was the first non-UNIX system to obtain POSIX certification.

Now, Linux has emerged as what we all hoped UNIX would be, a POSIX-compliant open system, which is available from a large number of suppliers on a wide array of hardware platforms and rapidly gaining wide software support.

Thus, Linux represents a great opportunity for us and a natural evolution of our open systems strategy. LynxOS is already a POSIX compliant, hard real-time operating system that is scalable down to very small footprints (minimum code size 32 Kbytes). And LynxOS is in many ways closer to Linux than any other RTOS. As a POSIX certified RTOS, it is highly compatible with Linux at the source level.

Additionally, LynxOS uses the same development environment as Linux, since both are based on the GNU tool chain. And LynuxWorks customers have frequently used large amounts of application code from the Linux environment (as well as from other UNIX systems such as Solaris) and moved them to LynxOS. This generally involved little porting work beyond recompiling and building. Similarly, Linux device drivers are easily ported to run on LynxOS.

The LynxOS kernel provides a highly scalable, fully preemptible hard real-time kernel specifically designed for real-time embedded systems. However, much of the non-kernel part of the LynxOS distribution is based on open-source technology and is already very close to Linux. This includes the GNU C/C++ compiler tool chain, X-windows, TCP/IP networking software, a large number of BSDI utilities, Samba and Apache. LynuxWorks is already heavily involved in and very familiar with the open-source model.

So it was a natural step for us to be the first established RTOS vendor to embrace Linux. As a part of the Lynx Linux Initiative (L2I), which we announced last November, we introduced BlueCat, a version of Linux specifically tailored for embedded applications that builds on our experience and infrastructure developed over the last decade serving our embedded customers. At the same time, we decided to evolve our LynxOS technology to bring the manifold benefits of Linux to LynxOS users. Future releases of LynxOS will feature full binary compatibility with Linux. The objective of that move is to insure that a binary executable file that runs on Linux can be loaded and executed directly on a LynxOS system (on the same CPU architecture, of course) without requiring recompilation or porting.

LJ: Can you briefly talk about pros and cons of using Linux?

Inder: For the embedded arena, which is very fragmented and populated with proprietary OS solutions, Linux provides for the first time the potential of an open, multi-vendor platform with an exploding base of software and hardware support.

The Open Source community development model, with the Web playing a crucial role, has been key to both the quality and robustness of Linux, as well as its popularity, and this will continue to be a major strength. In the embedded market, the royalty free nature makes it very attractive for high volume, cost-sensitive applications such as many of the emerging “post PC” appliance type of products. The wide availability of source and a huge and growing web-based developer community facilitates quick availability of support for new semiconductor devices, including ports to new CPU architectures. This makes Linux an important vehicle for tracking the technology curve.

The growing population of Linux-literate programmers, consultants and service organizations is a big plus.

The Linux kernel itself was developed for general purpose use, and its ongoing development is likely to be driven by the needs of the server area, particularly upwards scalability in competition with Solaris and NT. Thus it will fall upon the embedded Linux vendors like LynuxWorks and others to provide distributions tailored for embedded use. These vendors are addressing areas like cross development environments, tools for embedded systems and support for a variety of hardware configurations specific to the embedded world. Such embedded versions of Linux will serve the needs of a significant subset of embedded applications. However, many embedded applications require hard real-time performance, or smaller footprints that have been the province of proprietary RTOSes. There is only so much that you can do with an implementation that was designed for a different purpose. LynuxWorks is extending the benefits of Linux to such applications by its unique approach of supporting the Linux APIs and ABIs with LynxOS, which has been developed from the ground up for such applications. One of the nicest things about open standards is that you can have multiple implementations from different vendors that meet the needs of different constituencies.

LJ: What tools/developments would like to see added to Linux?

Inder: A well defined hardware abstraction layer (HAL) would make Linux a lot easier to port to a variety of different CPU architectures and hardware platforms. This is particularly important for embedded systems, but it would be very helpful for the server area as well, as the world moves beyond the Wintel architecture. In LynxOS, we use what we call a CSP/BSP (chip support package/ board support package) architecture which separates out the hardware dependent code related to the CPU, and the supporting peripherals at the board level, from the hardware independent code in the rest of the OS.

LJ: You recently changed your company name from Lynx to LynuxWorks. Why, and how do you explain the unique spelling?

Inder: The name change makes a clear statement about our commitment to Linux as our corporate strategic direction. It is kind of interesting that our previous name Lynx, which predates Linux, becomes Lynux with the addition of a "u". The unique spelling with the "y" emphasizes our continuing commitment to our current LynxOS technology, which is an integral and important part of our strategy for bringing the benefits of Linux to the embedded world.

We have a taken a unique approach to embedded Linux by offering our customers a choice of two kernels: BlueCat, which is our open-source Linux distribution tailored for embedded applications; and LynxOS, our Linux-compatible, POSIX compliant world-class, real-time operating system (RTOS) technology with a track record of a decade in demanding, mission-critical applications.

The dual OS approach offers choice and flexibility to our customers and enables us to bring most of the benefits of Linux to a wide range of applications, including those traditionally served only by embedded RTOSs.

From the embedded perspective, you can look upon Linux in two ways:

- as an excellent open-source operating system technology which can be adapted to many embedded applications; and
- as an open standard platform for software that runs on Linux, as defined by the Linux APIs and ABIs. Both of these aspects are very significant for the current and future success of Linux.

The former has a lot to do with the where Linux is today, but it is the latter aspect of Linux as a platform for a growing base of open-source as well as commercial applications, in particular, that will make Linux have its greatest impact on the computer industry, and especially so in the embedded industry.

The key to Linux maintaining its momentum, and having the huge impact on the computer industry that many proponents are hoping for, lies in its achieving wide acceptance as an open standard platform for applications.

The fact that Linus Torvalds chose to implement the UNIX interfaces rather than simply create a new operating system, and that Linux substantially conforms to POSIX, has been largely responsible for its popularity, and it further strengthens its position as an open standard.

In the embedded world, this aspect of Linux is particularly significant, because this is such a fragmented industry without any kind of a standard or leading platform. Around half of all embedded designs still use in-house operating systems, and the leading commercial RTOS product accounts for only a fraction of new embedded designs. As a result, the embedded world is missing a robust software industry such as the software industry that developed around DOS and Windows for the desktop. Embedded developers do a lot of reinventing the wheel and developing much more software from scratch, instead of building on top of existing tools and application software like their mainstream counterparts.

Linux as an open standard has the potential to serve as the missing platform that can foster an embedded software industry. As embedded applications are becoming more complex, the boundary between embedded and mainstream applications is blurring. So more and more of the mainstream software becoming available on Linux is already useful for embedded developers.

LJ: Congratulations on being named to the directorial board of the recently formed Embedded Linux Consortium. What role do you see yourself playing within the Consortium? Will it be greater on the technical side, with regard to developing standards, etc., or on the marketing/promotional side, with regard to getting OEMs to consider embedded Linux as platform?

Inder: Thank you.

I strongly believe that Linux has an enormous and beneficial role to play in the embedded area. In fact, Linux will eventually have an even bigger play in embedded systems than in servers, and I believe that the Embedded Linux Consortium can play a very influential role.

The initial focus of the ELC is largely in the promotional, marketing and communications area, to create greater awareness of Linux's benefits and create market momentum, but it will undoubtedly evolve depending in large part on the desire of the membership.

It is very important for the embedded Linux community to avoid a rerun of the “UNIX Wars”. The ELC can play a key role in getting the embedded Linux community to rally behind standards activity such as the Linux Standards Base and POSIX, although the ELC is not planning to directly engage in the development of standards at this time.

LJ: What are the issues that embedded Linux systems providers need to agree on, given the fact that they are competing with each other also?

Inder: Avoiding fragmentation even as we compete with each other is really the most important issue, which I have covered in answer to your previous question on the subject. The key is to establish Linux as an open standard platform for applications to foster the growth of an embedded software industry around Linux. A broad agreement to build in the standardization work of the POSIX and LSB groups instead of creating new APIs or standards that favor a single vendor, and a recognition that it is in our mutual interest to support Linux as an open standard in preference to proprietary single-vendor solutions, even as we compete and add our own kinds of unique value, is something we should all work towards to avoid a replay of what happened to UNIX over the last decade.

It is also important for embedded Linux providers to be good members of the Open Source community, make meaningful contributions to the open code base and adhere to not only the letter but also the spirit of the open-source rules. I believe in this strongly, even though at LynuxWorks we provide both open-source and licensed products.

LJ: In regards to embedded Linux deployment: How promising is the handheld arena for embedded Linux—given the current preference for OSes like Windows CE/Windows Pocket PC? What are some of the other major areas for embedded Linux that LynuxWorks' has taken or plans to take advantage of?

Inder:In spite of the current dominant position of PalmOS, and the strong Microsoft push with Windows CE and PocketPC, I believe Linux can still establish a major position in the handheld arena. This whole field of “post PC appliances” is still in a state of flux, with a plethora of new Internet connected devices of many kinds about to hit the market over the next year or two. As an open system supported by many vendors, Linux is a very attractive candidate, and many large vendors that we are working with are in fact designing many next generation products around Linux. The low royalty costs (can't get much cheaper than zero!), freedom from dominance by a single vendor and the huge community of potential developers of add-on software makes Linux a very attractive choice.

In addition to the emerging applications in this post PC appliance arena, we are also addressing the automobile, point of sale terminals and wireless markets, as well as the broad Internet and communications infrastructure area, the aerospace and military industry and instrumentation with our Linux based thrust. In fact, I have been pleasantly surprised at the broad interest in Linux across all the markets that we have been addressing.

LJ: The standard C library for Linux is the GNU C library, or glibc, which is rather large. (libc-2.1.3.so on my box is almost 900K, and a statically linked “hello world” is more than 200K.) How can embedded Linux vendors keep embedded code small while at the same time maintain the advantages of a large pool of contributors and testers?

Inder:This is one of the areas in which embedded Linux vendors can add special value by providing facilities that offer more configurability and greater control over the “footprint”. As your question makes clear, this applies not just to the kernel, but compilers, libraries and other parts of the environment. In the case of libraries, the objective would be to include only the minimum amount of code required for a given system, and cutting down on loading any unneeded code.

LJ: Everyone's talking about preventing fragmentation, to the point where fragmentation-prevention strategies seem to be fragmenting. What's the One True Way to prevent fragmentation?

Inder: The success of an operating system is dependent on the applications support that it generates more than any other single factor. So I would say that the One True Way to prevent fragmentation is a strong, industry-wide commitment to ensure that there is a clear definition of what constitutes “a Linux application”, and that an application developer has the assurance that a single version of his or her application will run on any Linux system.

The Linux Standard Base (LSB) project, whose specific goal is “to develop and promote a set of standards that will increase compatibility among distributions and enable software applications to run on any compliant Linux system”, is intended to prevent this very problem. It deserves strong support from application developers and users.

LJ: There's a plethora of cheap boards, free developer tools and Internet discussions about embedded Linux. It's a virtual toy store. If I'm a Linux hobbyist looking to build a cool toy, like an embedded Internet stereo for my car or an autopilot for my model airplane, where do I start?

Inder: First I would look at what are the requirements of the entire system that I am trying to build. Then, I would select a basic hardware platform and a version of embedded Linux.

For a hobbyist application, I would recommend starting with an inexpensive Pentium PC motherboard, perhaps one of the many small variants with smaller form factors. For a commercial application, one would also explore other architectures, including system-on-chip (SOC) solutions which provide a variety of different adjunct processing functionalities (i.e., communications controllers, display controllers, etc.) that are being added onto the core chip and satisfy a variety of embedded system needs.

Next, continuing to focus on a commercial rather than a hobby application, I would find a Linux supplier that fully tests this processor technology with their embedded Linux Product. The available testing on the standard PC is very well supported but on other architectures is more “spotty.” I would also look at the long term stability of the vendor and if they are ISO 9001 quality process certified. Additionally, I would look at what the chances are that the vendor will be around for the next ten years, for support and additional upgrades to the Linux product. Most embedded systems have a much longer lifecycle than the lifecycle of a standard PC. Finding the right software vendor for a long term solution can often be most important to the success of the project. So, in some cases, choosing the right processor might come from the list of the supported platforms from your preferred software provider.

I would also use the Web as a tool to look for other pieces required for the solution, such as digital tuners and Linux drivers to support the required hardware components. This is one of the great benefits of using Linux—the Web based community and all of the useful hardware and software solutions, as well as advice, that are available.

LJ: TiVo printed a notice in their product manual stating that any customer who wanted a copy of their modified Linux kernel could get one by writing to a certain address. This is a great way for a company to both fulfill its obligations under the GPL and build a mailing list of hackers. Is this the plan other vendors of devices containing Linux will adopt?

Inder: This is a reasonable solution to complying with the GPL requirements for device manufacturers who use Linux as the OS for their embedded software. They only need to do this if they make modifications to the Linux kernel itself. If they simply use an embedded Linux product such as BlueCat Linux from a vendor like LynuxWorks and build their embedded application on it without making any changes to the kernel, which is the norm, then this is not an issue for them. It is the responsibility of the embedded Linux OS vendor to provide the sources to all the changes made to tailor Linux for embedded applications. We do this by including the sources in our distribution, and we plan to make the sources available on our web site as well.

LJ: The Linux kernel developers are struggling with the issue of devices that can be added and removed on the fly, like PCMCIA cards and USB devices. Do you think that the need to support these devices will make the kernel too complicated for embedded systems? And how responsive do you think the "mainstream kernel" is to the needs of embedded Linux developers?

Inder: You are going to see increased use of both USB and PCMCIA interfaces and devices in embedded systems. Supporting dynamic plugging and unplugging of such devices is essential, especially for embedded products where you should never have to reset or reboot a system. The important thing for embedded systems is to make sure that, as for many other facilities, this is configurable and you do not carry the code whether your need the facility or not.

The issue is broader than USB and PCMCIA. Dynamic reconfiguration of any embedded system is desirable whether it is for detecting devices on system start or being able to add or remove devices on the fly. This often allows system developers to have one core kernel (for system support and configuration control considerations), especially in embedded devices where the customers might need a minimal memory footprint. In this case, only the needed kernel drivers and applications are added to the system upon detection. In some cases, embedded developers are looking at having the required drivers on the piece of hardware that is inserted and, at device installation, the driver is extracted from the hardware and installed into the system. This level of dynamic reconfiguration increases the usefulness of the system as a whole.

LJ: Any plans to fund more open-source software that can be available to embedded Linux developers? Tools like media compression and decompression or voice and handwriting recognition would come in handy, for example.

Inder: We recently announced the contribution of our LynuxWorks Messenger technology for BlueCat Linux to the Open Source community to establish a new standard for advanced CompactPCI inter-board messaging in distributed and high availability (HA) systems. Implemented as a kernel extension and associated ApplicationProgramming Interface (API) for messaging, LynuxWorks. Messenger is a technology that enables CPU boards (system and non-system) to exchange information on a peer-to-peer basis across the CompactPCI backplane.

We have also contributed multi-threading extensions to the GDB debugger to the Open Source community.

We fully intend to continue to support and make significant contributions to the Open Software community.

LJ: Thank you very much for your time.

Jason Schumaker (info@linuxjournal.com) has been working with Linux since 1998. He once parked in Bill Gates' parking spot (building #8 at the Microsoft campus) and was tortured viciously by security. Living through such an ordeal has helped Jason with fulfilling his pressure-packed duties as Assistant Editor and staff writer at *Linux Journal*.

Don Marti is the Technical Editor for *Linux Journal*. He can be reached at info@linuxjournal.com.

Archive Index Issue Table of Contents

Advanced search

Advanced 3-D Graphics: GNU Maverik—a VR Micro-Kernel

Adrian West

Issue #77, September 2000

A free programming system for high-end interactive computer graphics.

GNU Maverik is a system to help programmers create virtual environments (VEs). It was built by an academic research group to tackle some of the problems found when using existing approaches to VE construction. Maverik's main contribution is its *framework* for managing graphics and interaction, but it comes with extensive built-in functionality to make getting started straightforward. However, Maverik is a tool for programmers—it is not an end-user application.

Maverik is part of the GNU project and is distributed freely with full source under the GNU GPL. The distribution includes documentation, a tutorial and examples. Released in February 1999, Maverik has been downloaded by thousands of sites worldwide, and received positive feedback from both academic and commercial organizations.

In this article, I give some background to the challenges facing the designers of Virtual Environments and then describe how Maverik addresses some of these.

Introduction

It's easy to get enthusiastic about interactive 3-D graphics, and to imagine great things—walking around inside your dream house; conversing face to face with a distant friend in a shared paradise; or rehearsing complex engineering procedures with professional colleagues around the world in a shared virtual environment.

These things are easy to imagine, but surprisingly hard to do—witness the lack of really compelling examples of this kind of technology in use. So what's the problem? Until quite recently the limitations of computer hardware and VR peripherals were seen to be the limiting factor. In the last year or so, PC graphics accelerators have made dramatic strides forward as 3-D becomes a more mainstream facility. Today inexpensive 3-D PC accelerators rival the performance of the most expensive 3-D workstations, and are beginning to include options such as economic stereo shutterglass support.

The limiting factor—the gap between what we can so easily imagine and what we can readily achieve—is now more clearly exposed as being software. Writing 3-D applications is hard work. It takes hundreds of person-hours (programmers and artists) to create the impressive games and animated film sequences we have come to expect of 3-D computer graphics. Yet film animations such as *Toy Story* use techniques that are way beyond what can be achieved at interactive framerates. The best of today's PC graphics cards is something like 10,000 times too slow to do *Toy Story* in real time. That is largely due to the complexity of the modeling and sophisticated lighting calculations used. It is possible to do impressive things in real time, as computer games demonstrate. But games rely upon methods such as texture mapping to give the illusion of complexity within a VE far beyond what is actually present. In a game, the complex metalwork and scenery that you see are mostly stage scenery: you cannot take the girder away from the wall, it's only a picture of a girder.

In contrast, for real engineering tasks, the complexity of the CAD models can be staggering—a model of a real offshore platform can, for example, equate to half a gigabyte of polygonal data which must somehow be processed 10-30 times per second if it is to be any use for interactive work in a VE (see Figure 1).

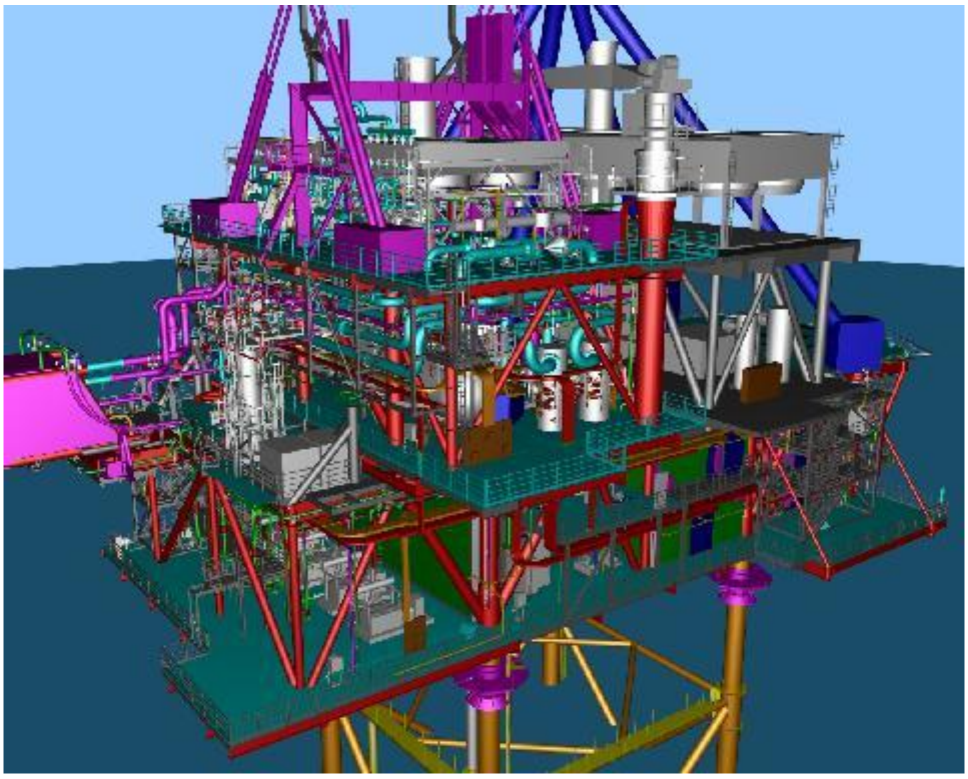


Figure 1. Offshore Platform

Let's think about the kind of things that need to be done to work with a model of this complexity; this will help understand the reasoning behind Maverik. To cope with the offshore platform, we need to find ways of rapidly focusing in on only those parts that need drawing at any one instant. For example, in a building with many rooms, you can quickly discard large parts of the model that you know cannot be seen from the current position. In a large “cityscape” at ground level, we could work out which buildings you can see that are not occluded by others, and rapidly discard everything else. So some application-specific tests can quickly discard a lot of irrelevant data. In some cases, those do not work; for the offshore platform, we can see between the pipes all the way across the rig, so we need a different strategy. In the limit, though, we will run out of time to render the current frame, so some bail-out option is necessary. One example would be to fall back to a wire-frame representation for more distant parts of the view (see Figure 2). That does not make such a nice still picture, but if it lets you work interactively within the VE, then it might be tremendously effective compared to a “jerky” presentation.

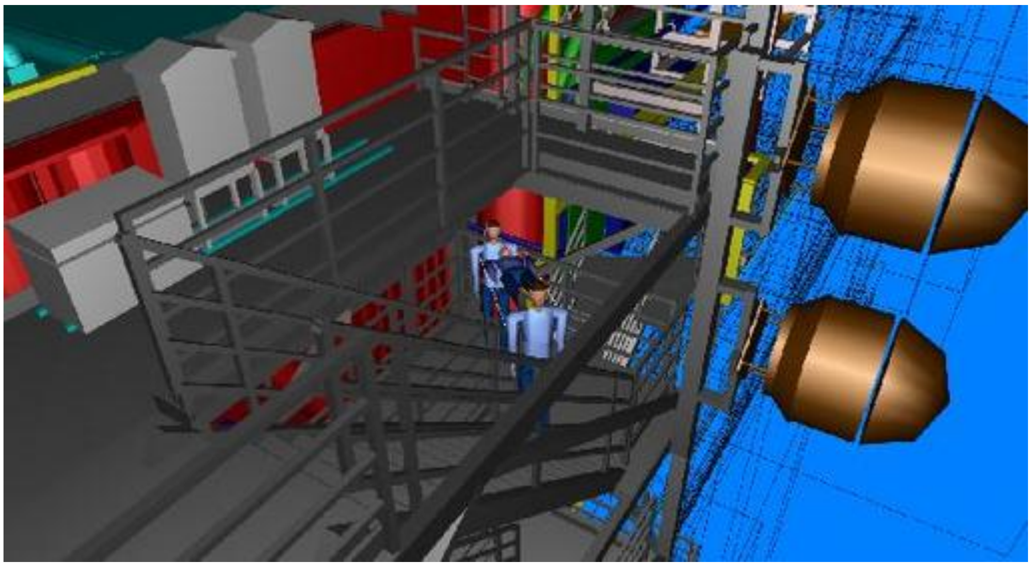


Figure 2. Wire Frame Bounding Boxes

A promising approach is to replace the wire-frame portion with image-based rendering techniques, filling in the background with appropriate still pictures of the rig. These stills are then dynamically distorted so you do not notice that they are stills. That is a subject of current research. For a VE, it is important to make any such transitions between representations smoothly, and avoid objects “popping” between different rendering styles, which is quite disturbing.

Most of these techniques are well understood individually, but the particular combination of tricks employed to get enough speed are often highly application-specific. Putting together each frame, whilst managing other issues such as user navigation and application behaviour, is quite a challenge. At present, if you want to attempt this, you have two basic strategies available.

First, you could program the whole thing from the ground up in your favourite language and a 3-D graphics library such as Mesa or OpenGL. This gives you a lot of flexibility—you can do anything that's possible, and at the best performance attainable. The drawback is that for a sophisticated task, this can be like programming an operating system in assembler. It's too low a level, and hence hard to re-use what you write for different kinds of applications.

Your second choice is to use a proprietary package for building Virtual Worlds. Such a package will get you going quickly, and hopefully has built-in high-level support for your desires—for example, making your VE sharable between several users. To support these high-level features, the VE package has its own internal complexity and representations of the VE, which is, after all, what you are buying. The drawback here is that the complexity within the VE system may not be appropriate for what you want to do (i.e., a “one size fits all” approach may not work). If you have a complex application, it will have its own data structures and algorithms tuned to that application. In the case of the offshore platform, the CAD system that built it talks in terms of ladders, pipes, valves and so on—quite high-level descriptions. To use this data with the VE building package, it will have to be exported from the CAD system, and imported in whatever format the VE system understands internally. The common denominator is, more often than not, a “sea of polygons”. So in exporting the data into the VE system's general-purpose format, much of the interesting information on what those polygons mean is lost, and with it the potential for exploiting that information to win speed.

A second problem with this approach is that if someone within the VE manipulates the objects (disconnecting a pipe or valve for example), then somehow that has to be communicated back to your CAD application which must then update its own data structures. So you get two representations of the world, one within the application, and one within the VE system, and these have to be kept synchronized.

So an “off the shelf” VE building package does what it does, well. But as the desire for complex VEs increases, we encounter problems of programming complexity, and the inherent performance limitations of maintaining two different models of the data—one of which is probably an unwieldy sea of polygons.

Maverik—A Middle Way?

Is there any middle way possible? Can we construct some framework that supports the creation and management of VEs, without limiting what can be built to some lowest common denominator?

We believe there is, and the result of those efforts is GNU Maverik. A framework for interactive VEs has now been used successfully to implement a range of VE applications with markedly different demands and optimization strategies.

Maverik is a C toolkit that lives on top of a 3-D rendering library such as Mesa or OpenGL, providing facilities for managing a VE. It is best viewed as the next layer of functionality you would want above the raw 3-D graphics library. The novel aspect of Maverik is that it *does not have its own internal data structures* for the VE. Instead, it makes direct use of an application's own data structures, via a callback mechanism (rather like callbacks are used in window managers such as X11).

An application tells Maverik about the classes of objects it wishes to render, and supplies functions needed to do the rendering. Maverik supplies facilities for building flexible spatial management structures (SMSes) with which the registered objects are managed. As the user navigates the environment, Maverik tracks the SMSes and makes appropriate calls on the application to render its objects.

What Is the Gain?

The application keeps control of its data, so there is only one copy to manage. More crucially, the application can bring all its knowledge of what the data means to bear when deciding how to render the objects; we find this is where the most telling optimizations come from. For example, if 3-D text is important to the application, then a good data structure would probably involve text strings and position information. However, exporting the 3-D text to a self-contained VE package will probably mean exporting the data in a common graphics format—usually polygons. The VE system will have a lot more data to process than we would like, which limits the size of environment we can construct.

With Maverik, the application might register the class “3-Dstring”. Later, when the user navigates around the environment, Maverik determines from the SMSes that certain objects are in view. For 3-Dstring objects, it calls the application supplied functions to render those, and the application then generates the polygons on the fly. The application can use whatever optimizations are appropriate when generating the polygons, which makes techniques such as dynamic level of detail, and coping with objects that change their shape, straightforward. In this way, Maverik can be a general-purpose framework, and yet benefit from optimal data representations and highly application-specific optimizations.

The 3-D text example is part of a real application—the “Legible City II”, a multimedia artwork by Jeffrey Shaw (ZKM Germany). It uses the above technique to render three large cities (Amsterdam, Manhattan and Karlsruhe) of words (see Figure 3). To render a more game-like environment, the application callbacks will use the more traditional games techniques to render their objects (see Figure 4).

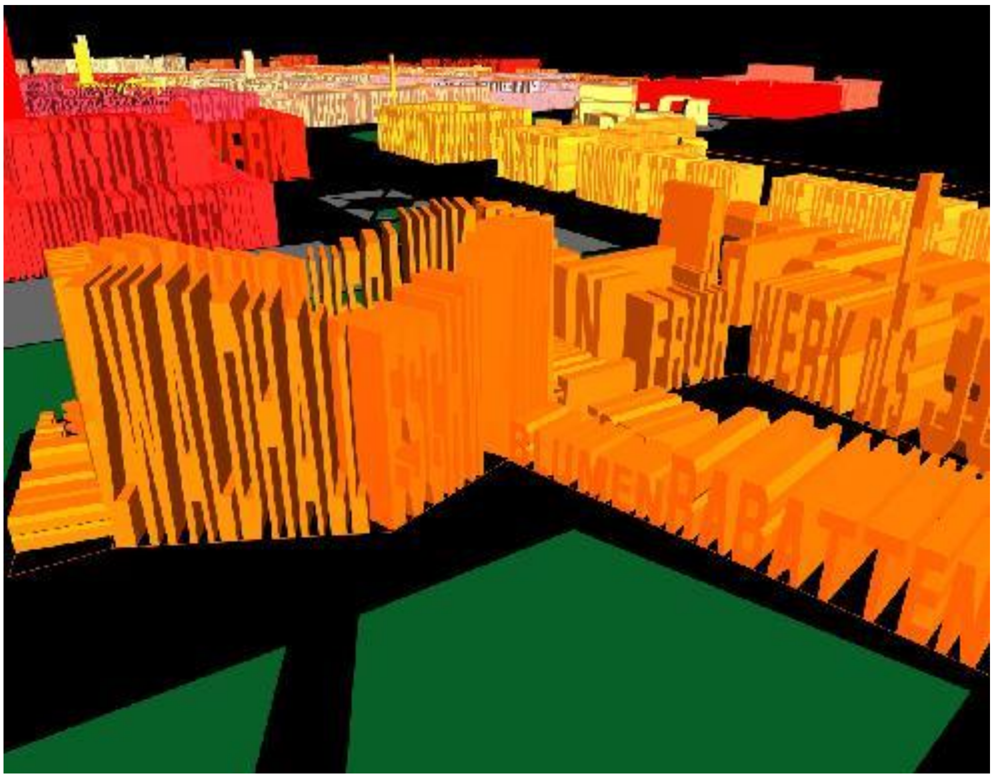


Figure 3. Legible City II Multimedia Artwork Implemented in Maverick

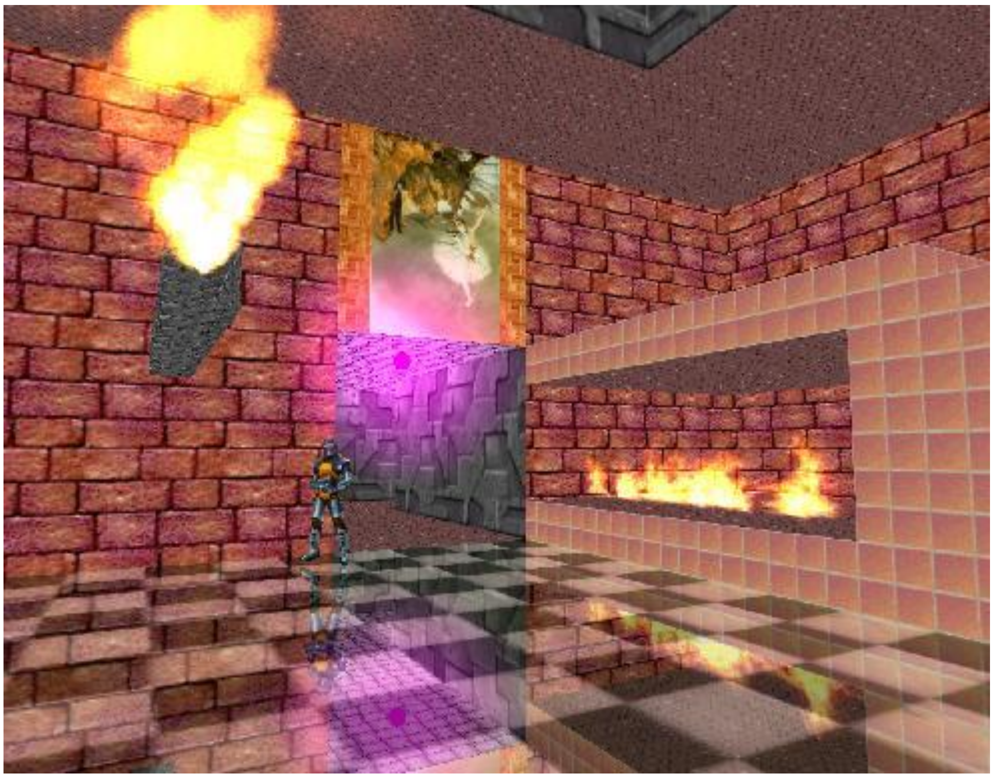


Figure 4. Special-Effects Techniques Common in Games

For the Oil Rig, the CAD application's native data structures (pipes, valves, walkways and so forth) are used in a similar way. When the application is called to render a tower, it can decide what would be an appropriate rendering for this frame—for example, it may decide to always render the tower, because it is an important landmark for navigation. Thus, the application can change representations dynamically to suit conditions. Similarly, applications with special lighting algorithms (see Figure 5), or atypical behaviours such as abstract information visualization (see Figure 6) use their own bespoke structures and algorithms to maintain real time performance.

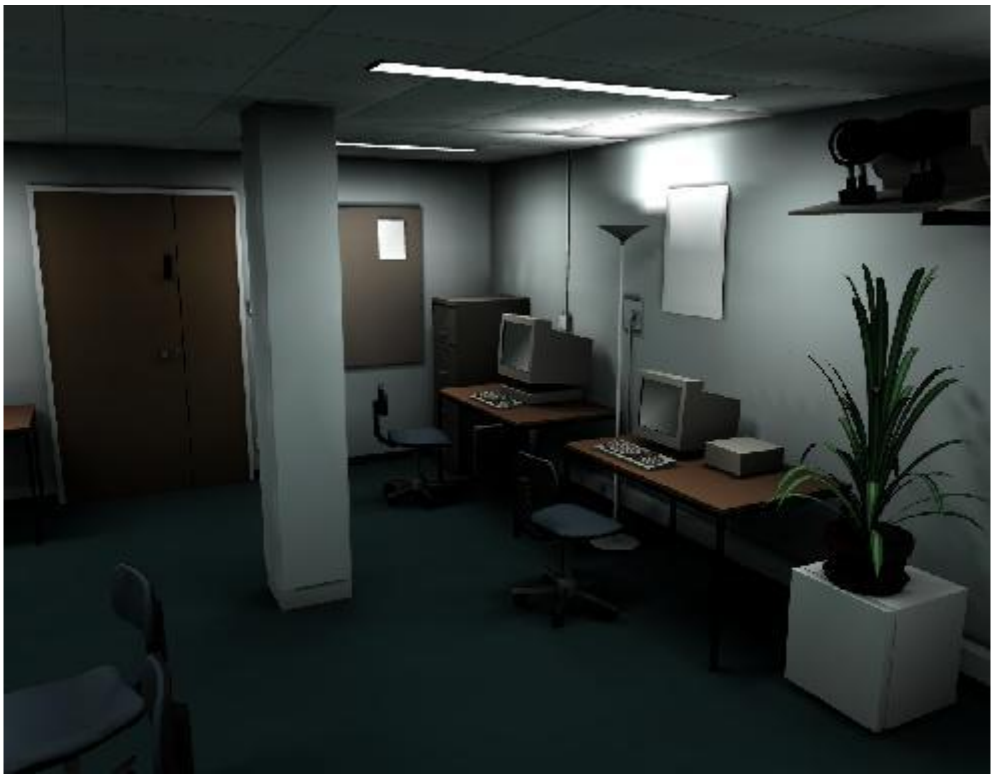
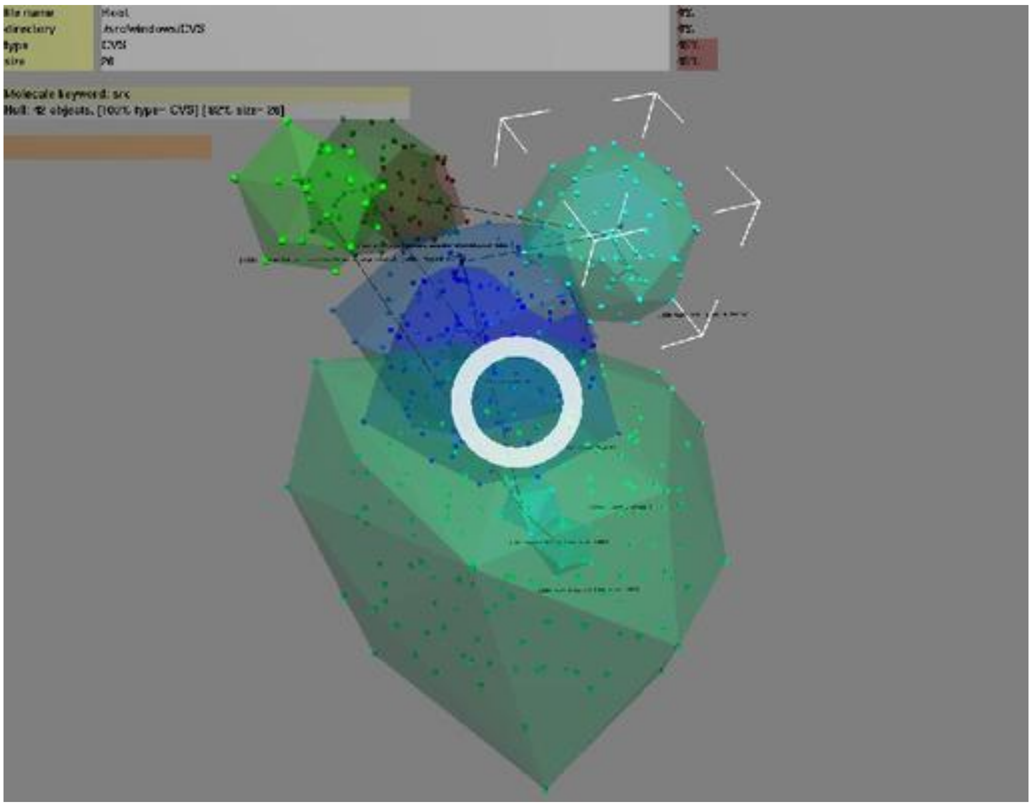


Figure 5. Radiosity Lighting Model


file name Root
directory /src/windows/CVS
type CVS
size 28

Molecule keyword: src
Hull: 12 objects. [100% type= CVS] [92% size= 28]

Figure 6. Data Visualization

Examples of work done with Maverik are given in the Maverik gallery on the web page. The important feature is that the video sequences (which are quite large files) are all captured directly from a 450MHz PIII Linux PC using a Voodoo2 3-D accelerator, and give a good impression of what performance can be attained with a cost-effective hardware setup.

Is This Cheating?

What we have gained is the flexibility to plug together quite different applications within a single VE framework, without compromising performance. That is a gain, but seems to be at the expense of having the application (and application programmer) do all the hard work. However, because Maverik is a *framework* into which things can be plugged with little performance loss, it is easy to provide commonly useful facilities and objects to get many applications off the ground. For example, Maverik is distributed with libraries of common geometric primitives, cones, cylinders, teapots, an animated avatar, sample graphics file parsers, navigation facilities, 3-D math functions, quarternion codes, stereo, 3-D peripheral drivers for head-mounted display and 3-D mice (see Figure 7) and so forth). If these are useful, then you use them as a good starting point. If they are not, then you provide your own objects and algorithms, perhaps refining the samples provided.

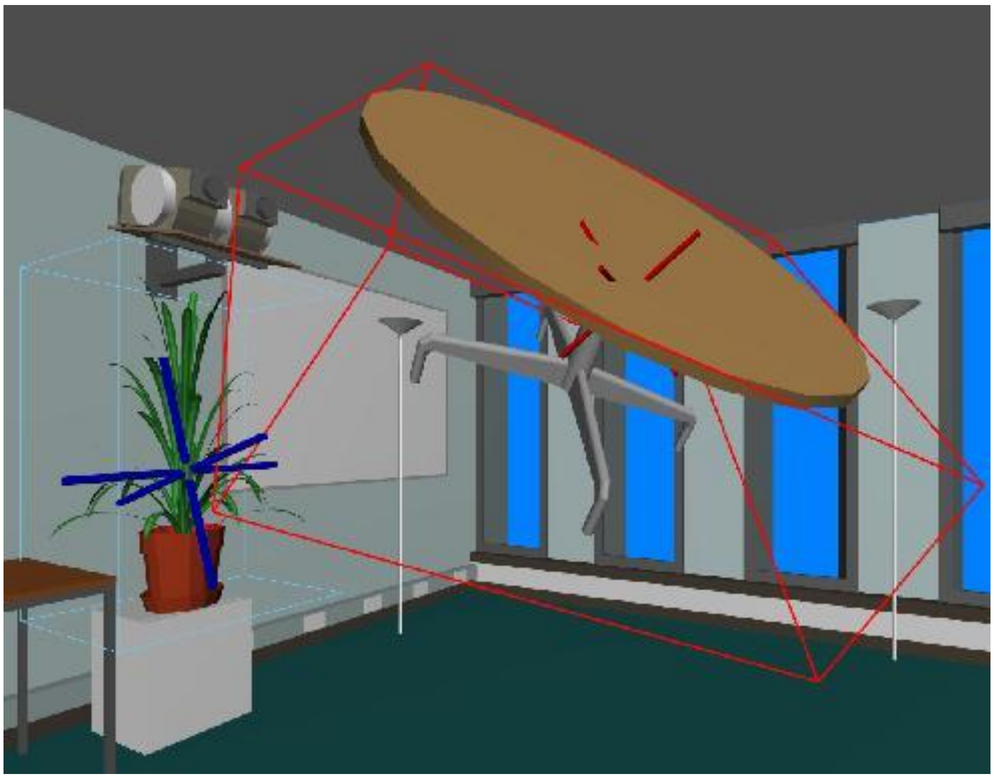


Figure 7. Immersive Editing

This makes it fairly straightforward to get started using the system. In practice, this gives three levels of increasingly sophisticated use.

1. Using the default objects and algorithms provided. Used this way, Maverik looks like a VR building package for programmers. The tutorial documentation for the system leads one through the building of an environment with behaviour, collision detection and customized navigation.
2. Defining your own classes of objects. This is where you benefit from application-specific optimizations that you could supply by providing your own rendering and associated callbacks. The tutorial gives an example of this, and the supplied demonstration applications make extensive use of the techniques. Unfortunately, the offshore platform example uses commercially sensitive data, so cannot be supplied with the distribution.
3. Extension and modification of the Maverik kernel. Rendering and navigation callbacks are one set of facilities that can be customized. For the adventurous, more of the Maverik core functionality is also up for grabs: for example, alternate culling strategies, spatial management structures and input device support. But this still takes place within the context of a consistent framework that seems to make it easy to plug different parts together.

The real test for Maverik as a research vehicle is how well it allows such pieces to fit into the overall puzzle. The hope is that it will make the task of building VEs “as hard as it should be”, avoiding the feeling that one's efforts are mostly spent “fighting the system”. So far, given a little familiarity with the approach, this looks very promising.

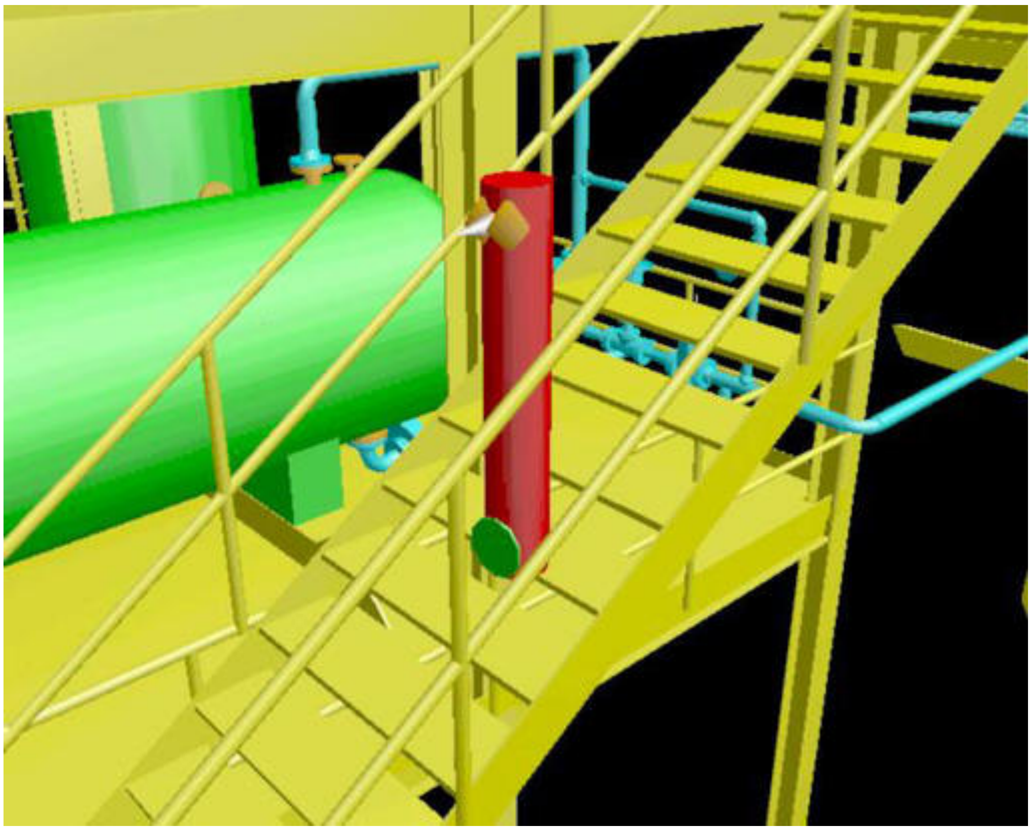


Figure 8. Navigation Algorithms

Recent Maverik developments include a novel force-field navigation algorithm to guide participants around obstacles, and an algorithm for probing geometry ahead of a user to test whether it can be climbed—for example, ladders and steps (see Figure 8). These algorithms integrate into Maverik, but at the time of this writing are released as separate beta sources for testing.

What You Can't Do with Maverik

Maverik has no support for audio or video within it. You will need to add your favourite mechanism for those yourself.

Maverik is a single-user VE micro-kernel. It does not include any assistance for running multiple VEs on different machines for sharing a world between people. To do that, you would need to synchronize the VEs yourself across the network. For just navigating around a shared VE, that's not so hard. Running a system with many users across wide area networks with multiple VEs, applications, interaction and behaviour is a more complex challenge. It's a challenge we are working on now with a complementary system that uses Maverik. We aim to release what we have on that under the GNU GPL license later this year.

We are very keen to know how well these ideas work, or don't work, for other people—particularly those with experience of graphics programming. That way, we get to understand more about this whole area. Feedback should be addressed to maverik@aig.cs.man.ac.uk.

Acknowledgements

Resources

email: ajw@cs.man.ac.uk

Adrian West (ajw@cs.man.ac.uk) lectures on computer science at the University of Manchester, U.K. He is part of the Advanced Interfaces Research Group, working on systems architectures for distributed virtual environments.

Archive Index Issue Table of Contents

Advanced search

The Puzzle of 3-D Graphics on Linux

John Matthews

Daryll Strauss

Issue #77, September 2000

New opportunities for game developers are provided by several tools.

Over the past year, there has been a drastic change in how Linux is viewed by gamers. No longer relegated to the role of just a dedicated server, gamers are using Linux in increasing numbers as their primary OS and as their gaming platform. And as if using Linux wasn't a challenge enough in itself, anyone trying to set up a Linux box to play a 3-D game will find the path fraught with a plethora of confusing acronyms and names. From OpenGL to DRI to DGA, each term refers to a particular part of the full scene of Linux graphics. Here we put those pieces together for our readers to give them a broader view of what Linux has to offer now and what it will offer, in the near future.

Figure 1. Heavy Gear II and Quake III Running Simultaneously

What is OpenGL?

We start by looking at OpenGL, a name that's gotten a lot of use lately, especially in reference to games and professional design applications. Originally called IRIS GL, OpenGL (Open Graphics Library) is a programming library that provides a rich array of graphics functions, both 2-D and 3-D, allowing the programmer to represent any object they design on the screen. It was developed by Silicon Graphics (SGI) and has become a standard graphics application programming interface (API) on many platforms including UNIX, Linux, Microsoft Windows and Apple Macintosh. While OpenGL is the standard for high-end graphics applications where programmers have found it powerful and easy to use, most people have heard of OpenGL in reference to the Quake series of games from id Software. It has been used as part of the graphics renderer since Quake 1, where it has shown itself to be a powerful game graphics API as well. The standard for OpenGL is an open one, independent of hardware platforms, windowing systems and operating systems. Thus the word “Open” as part of the name.

In order for OpenGL to be used in an OS, someone must create a library to implement the function calls OpenGL programs make. However, to call your implementation by the name “OpenGL”, you must actually obtain a license from SGI (or Microsoft). In fact, getting a license will earn you a package of code, called a Sample Implementation and written in C, from which you can build your OpenGL library on your platform. But since the standard for OpenGL is an open one, working from the SI isn't required. In fact, that's what Mesa is all about: it is an implementation created from scratch without a license from SGI and without any code from SGI. To use the OpenGL trademark, however, you must get a license from SGI and your implementation must be robust enough to pass a set of conformance tests developed by the OpenGL Architecture Review Board (ARB). The ARB historically consisted primarily of high-end hardware manufacturers and SGI itself, but now includes some game hardware manufacturers. It is worth noting that, while SGI created OpenGL, the future of the API is controlled by the ARB where SGI sits as one of many members.

Recently, SGI released a Sample Implementation (SI) that can be used to implement the API on any hardware platform with an appropriate compiler. This source code is very similar to the one sold to hardware vendors that implement OpenGL drivers for their video cards. While it has been said that the SI is now open-source software, this isn't entirely true. As of this writing, there are still some issues with the license attached to the SI that prevent it from being truly open.

Many people associate OpenGL with hardware acceleration. While it is true that OpenGL runs very quickly when it has hardware assistance, that acceleration is not required to run an OpenGL application. With a software OpenGL library, OpenGL applications can run and render the same image as they would with hardware acceleration, albeit significantly slower unless the program is very simple. When hardware acceleration is available (in the form of a 3-D graphics card and a driver written for that card), OpenGL applications can run very quickly and smoothly, since most of the intense operations have been offloaded from the CPU to the dedicated graphics board. However, not every accelerator has the ability to perform all the features of OpenGL. When a feature is not supported in hardware, the library may fall back to a software implementation which uses your CPU. For example, very few consumer-class video cards will do transform and lighting (T&L), so normal triangle setup is often done partially in hardware and the rest in software.

Since its initial creation, OpenGL has undergone several revisions and now stands at version 1.2.

In Linux, the implementation of OpenGL used most often is Mesa, created by Brian Paul.

Mesa: Open Source OpenGL

Mesa is an unlicensed implementation of the OpenGL standard that is available on several platforms, including Linux, Windows and Macintosh. Originally started in 1995, it has been under development ever since by a team of developers led by Brian Paul. Mesa is freely available and open source, so anyone can work with the source code and port it to another operating system.

For the most part, OpenGL applications can compile and run against a Mesa library just as they would against a licensed OpenGL library.

Provisions for hardware acceleration are incorporated into Mesa. For example, owners of a 3dfx card may choose to download and install the Glide SDK from 3dfx and then recompile Mesa from its source code. When Mesa configures itself for compilation, it should detect the installed Glide headers and libraries and consequently add the necessary code to allow the 3dfx card to accelerate many of the OpenGL functions (via Glide 2.*x*).

Like OpenGL, Mesa has undergone several revisions. As of Mesa 2.*x*, the OpenGL 1.1 standard has been supported. The later Mesa 3.*x* library is an implementation of the OpenGL 1.2 standard, and thus should be nearly as current as OpenGL itself. Mesa also includes support for GLUT and GLU.

So now we have OpenGL, a programming interface for creating 3-D graphics, and an open-source implementation called Mesa. The next part of the puzzle is the glue that joins OpenGL and the X Window system.

GLX: Using OpenGL with X Windows

As OpenGL is platform- and system-independent, it is also window-system-independent. Thus, it needs a window system binding to allow it to interact with the window system. This binding provides the functionality for actions like finding the location of a window on the screen or how to process input. In the case of UNIX and Linux systems, that's GLX, a library that allows OpenGL and X to operate together. (In the case of Microsoft Windows, it's called WGL.) That is, with GLX, it is possible to have OpenGL utilize an X window for its output. Even when you're using Mesa (full-screen or non-DRI; we'll explain DRI in just a moment), there's a fake GLX implementation to make the system think it is running under the normal window system binding. The GLX currently used in Linux is based on the source code released by SGI in February, 1999.

Some of you might even have run across the term Utah-GLX. What is Utah-GLX, and what part does it play in this puzzle?

Utah-GLX

Utah-GLX is a project to add OpenGL capabilities to some current video accelerators, like the Matrox G400/G200 cards and the ATI Rage Pro and Rage 128 cards, while still using XFree86 3.3.*x*. The Utah-GLX driver uses indirect rendering (and in some cases, a form of direct rendering) to provide this kind of acceleration. However, as will be explained shortly, this rendering incurs a performance penalty that prevents the hardware from reaching its full potential.

Among other accomplishments, the Utah-GLX project has led to the first hardware acceleration on the Linux PPC platform as well as the first hardware acceleration for a laptop. In both of these cases, the video card is the ATI Rage Pro.

Now that XFree86 4.0 has been released, the hope is that much of the work in the Utah-GLX project can be ported to the DRI. While there has been some talk of starting this move, as of this writing it hasn't happened yet.

This talk of *indirect* and *direct* rendering naturally leaves some unanswered questions. Let's take a closer look and see how the two differ and where they are used.

Indirect vs. Direct Rendering

The difference between indirect and direct rendering is the number of levels through which the data must pass. That is, how much the data must be massaged before it is actually put in the frame buffer of the video card to be displayed on a monitor. As one would expect, the fewer the levels, the faster the image, and thus the emphasis on direct rendering (via DRI) as part of XFree86 4.0.

When indirect rendering is used, data is copied from the application issuing the graphics output to the X server, and from there to the hardware. This incurs a performance penalty, since the output from the application must be packaged into a form for the X server and then, once X has done its job, the final output must be packaged appropriately and sent to the hardware. In a normal 2-D application, the speed of this process is adequate. However, in today's CPU and memory-intensive 3-D applications, that overhead is too unwieldy for adequate performance.

Direct rendering attempts to streamline this flow of data and allows the application to access the hardware more directly. That is, it allows an application to issue its drawing commands directly to the graphics hardware, with only the minimum amount of necessary intervention by the X server. This ability exists in XFree86 as the Direct Rendering Infrastructure (DRI), developed by Precision Insight.

Finally, we get to talk about this mysterious DRI that has come up in our discussion a couple of times. So without further ado, read on to learn more about DRI and what it means for Linux users.

DRI: Security, Stability, Speed

The Direct Rendering Infrastructure (DRI) is a technology developed by Precision Insight to add a direct rendering ability to an X server. In more common terms, DRI allows applications to safely and efficiently access the hardware directly while cooperating with the X server. Unlike Utah-GLX, it doesn't require privileged users or privileged programs. Furthermore, the work takes place in the application instead of in the X server.

More technically, one can think of DRI as three parts: a special X server, a direct rendering client and a kernel-level device driver. The special X server for Linux is XFree86 4.0, whose code contains special DRI modifications. The direct rendering client consists of several smaller components. The most important of these are an OpenGL implementation (in most cases Mesa), GLX as developed by PI from SGI's code, and driver libraries from PI that are Linux-specific (but hardware-independent). The kernel-level device driver is a module that abstracts direct memory access for hardware. It is designed in such a way that only the minimum of kernel modification should be needed in the future.

In short, DRI is hardware acceleration “done right”. Since it doesn't require special programs or privileges, there is an inherent level of security which current hardware-acceleration implementations may not have. Also, it is part of the XFree86 server and has undergone a significant amount of testing. With such a development system, the hope is that users can count on the usual stability associated with XFree86. Finally, the direct access to hardware gives Linux and XFree86 the chance to act as a solid, fast platform for professional graphics applications and demanding 3-D games.

Loose Odds and Ends

In addition to DRI, OpenGL, Mesa and the other terms, there are some smaller parts of the picture that you, faithful reader, are likely to have come across. Let's take a minute to go over each of them briefly.

- GLUT began as a library called aux developed by Mark Kilgard (formerly of SGI, now at NVIDIA) to make programming examples easier in his book *OpenGL Programming for the X Window System*. This library was eventually improved and has matured into GLUT.
- DGA is the *direct graphics architecture*. It is a very simple extension to the X server that allows applications to have direct access to the frame buffer. Users and applications need to have privileges to use DGA. With those permissions, they can get access to the frame buffer memory and draw, unaccelerated, on the screen. With the release of XFree86 4.0, DGA 2.0 will be included, bringing some hardware acceleration to basic 2-D drawing functions. In addition to the graphics abilities, DGA allows an application

to read peripherals like the mouse and keyboard more directly. Normally, signals from those input devices pass through the X server before they can be used by an application.

- VidMode—the vidmode extension allows an application to query the X server for the resolutions it supports and tell the X server to change to a specific resolution. This is often used in games that want to run at a specific size when your desktop is set to a different resolution.
- Glide is the 3-D API developed by 3dfx for their Voodoo line of cards to provide fast, full-screen hardware acceleration. Its functionality was designed to match that of the Voodoo hardware 3dfx had created. Many early 3-D accelerated games, like Tomb Raider, used Glide. With the widespread acceptance of OpenGL for both productivity and entertainment applications, the importance of Glide has diminished. Consequently, 3dfx opted to release the source code and specifications to the public, a move which allowed its spread to other platforms.

Putting it All Together

Let's review: OpenGL is a library that facilitates the creation of 3-D graphics. Under Linux, there is an open-source library called Mesa which provides OpenGL compatibility. We have GLX to provide a standard way for OpenGL applications to work with X. For people still using XFree86 3.3.*x* and certain video cards, Utah-GLX provides some OpenGL functionality similar to the type that XFree86 4.0 and DRI provide. With XFree86 4.0, however, users get maximized 2-D and 3-D performance, better integration and more secure operation.

So there you have it. We can see how all the parts fit together and their dependencies upon one another.

Resources

Matt Matthews, a PhD student in Applied Mathematics at NCSU, became a Linux user in the summer of 1999 when he and his advisor bought new machines for their research. In addition to his computational work, he enjoys playing games and testing out the newest 3D video cards under Linux. When time allows, he writes up those experiences for his favorite website, www.linuxgames.com. If not deep in his research or writing, you can find him out with his wife, Mandy, or tending to his ever-growing collection of video games.

Daryll Strauss has been working Unix systems of one variety or another for the last 15 years. Linux became he preferred choice in 1995. During that time he has done image processing for the aerospace industry, special effects for the film business (including work on two films that won the academy award for visual effects), and on supporting open source 3D software for Linux. He is currently employed by VA Linux Systems, formerly Precision Insight, as a Visual Mason and Evangelist. In those rare moments when he isn't working you might find him playing beach volleyball, watching films, playing golf badly, or learning to play the harmonica.

Archive Index Issue Table of Contents

Advanced search

Linux Job Scheduling

Michael A. Schwarz

Issue #77, September 2000

How I learned to stop worrying and love the Cron.

Today, in our ongoing series on learning to live with Linux's “inner dæmons”, we are going to look at two dæmons that schedule job execution on Linux. These dæmons are more or less exactly like those found on virtually every UNIX out there. (Linux has separate dæmons for **at** and **cron**. Old versions of Linux used a program called “atrun”, which was run in root's crontab once a minute to execute at requests. Some other Unix operating systems have atd functionality directly in crond. This qualifier brought to you by the bureau of auctorial honesty. This article will cover atd and crond as they are distributed with most currently sold distributions, including Debian 2.1, Red Hat, SuSE and Corel, among others.) My test cases were all carried out on a Red Hat 6.1 installation using version 3.1.7 of at. Debian and SuSE versions I currently have are at 3.1.8.

As for cron, most Linux distributions use “Vixie cron” which was originally written, as you might guess, by Paul Vixie. The distributions have each done their own fixes to address a security hole discovered in August 1999. Check your distribution's update page for the most recent version of cron, and make sure you have it installed.

What you think about at and cron will largely depend on what your background is. If you are familiar with only the DOS and Windows world, you should be fairly impressed with what atd and crond offer, even if you have made use of the System Agent, which has certain similarities to crond. If you are an old hand from the world of MIS where you had JCL and various batch environment control systems, you will probably find atd and crond lacking in some essential features. Even so, I hope you will come away from this introduction with a healthy appreciation for what these tools do offer, and perhaps a few ideas about how, even with their limitations, they significantly enhance Linux's capabilities.

People with a mainframe background are very familiar with the concept of job scheduling. They usually use this term interchangably with batch processing. Alas, job scheduling is *not* batch processing. Batch processing, to my mind at least, includes the concepts of job dependencies, batch process monitoring, checkpoint/restart and recoverability. Neither atd nor crond provides these facilities. If you come from the world of big iron, you may be feeling some disappointment. Don't. As you will see, atd and crond fit in well with the overall UNIX philosophy of simple tools that do one thing well.

If you are coming from a Windows/DOS perspective, you should be pleased by the multi-user nature of atd and crond. Unlike System Agent, you do not have to be logged in for your jobs to be carried out.

If you have a UNIX background, well, you are amongst old friends here.

For those totally unfamiliar with these concepts, what we are talking about is running programs. So what, you say? I log in and type commands and click on little icons. I run programs all day. What's the big deal?

What about having programs run at a certain time of the day, whether you are there or not? What about compiling the latest version of WINE on a busy Linux server when it won't slow down the branch office Intranet? What about that annoying log file the on-line order application spits out that is about to eat up all the free disk space on /usr/prod/orders?

This is where job scheduling comes into play.

There are two kinds of scheduled jobs. You can think of them as “one shot” and “repeating”. One-shot jobs are single executions of programs you want to have take place at some future time, whether or not you are logged in. Repeating jobs are programs you want to have run at certain times or dates, over and over again.

The command you use to schedule one-shot jobs is called “at”. The way to schedule repeating jobs is through a “crontab” (which is a portmanteau word made from CRON TABle, similar to INITtialization TABle and other *nix-y portmanteau words). Oddly enough, the command used to view, edit and store crontabs is called “crontab”.

Unlike some of the other dæmons we have covered in this series, these two have interactive user programs that control them. Because of this, we will cover the basics of using these two dæmons as a non-privileged user (I hope you aren't logging in to your Linux system as root!), then we will go over the dæmons and how they work, then we will cover some fine points of “non-user” or system-scheduled jobs, and finally some of the little “gotchas” that sometimes cause commands to behave differently than you expect when you run them through a scheduler.

Using at

The **at** command is used to schedule one or more programs for a single execution at some later time. There are actually four client commands:

1. at: Runs commands at specified time
2. atq: Lists pending commands
3. atrm: Cancels pending jobs
4. batch: Runs commands when system load permits

The Linux **at** command accepts a number of time specifications, considerably extending the POSIX.2 standard. These include:

```
HH:MM
```

Run at this hour and minute. If this is already passed, the next day is assumed. A 24-hour time is assumed, unless you suffix the time with “am” or “pm”.

```
now noon midnight teatime
```

You read that right. You can type “at teatime”, and Linux's **at** is civilized enough to know that this is 4 p.m. local time. The “noon” and “midnight” keywords have their normal meaning. The “now” keyword means what it says. It might seem like a dumb thing to have, since if you wanted to run something now, you would type it without the **at** command, but it has an application in “relative time” invocations. We'll see those after the date modifiers described below.

Date Modifiers

These time specifications may be optionally followed by a date specification. Date specifications come in a number of forms, including:

```
today tomorrow
```

These mean what you would expect. “at teatime tomorrow” will run the commands at 4 p.m. the following day. Note that if you specify a time already passed (as in “at noon today” when it is 3 p.m.), the job will be run at once. You do not get an error. At first you might think this a bad thing, but look at it this way. What if the system had been down since 10 a.m. and was only being restarted now at 3 p.m.? Would you want a critical job skipped, or would you want it to run as soon as possible? The at system takes the conservative view and assumes you will want the job run.

```
<month_name> <day> [<year>]
```

where month_name is “jan” or “feb”, etc., and day is a day number. The year is optional, and should be a four-digit year, of course.

```
MM/DD/YYYY YYYY-MM-DD
```

Don't listen to what the “man at” page tells you! At least in Red Hat 6.1, it is wrong! I suspect it is wrong in certain other releases as well, and I'm willing to bet this is because the documentation has not caught up with Y2K fixes to this subsystem. The at shipped with Red Hat 6.1 handles dates in the two formats above. It appears to handle 2-digit years correctly, turning values less than 50 into 20xx and those greater than 50 into 19xx. I did not test to find the exact pivot point, and I do not recommend that you bother to, either. If you use two-digit years at this point, be prepared to pay a price! Depending on your version of at to treat two-digit years a certain way is foolish. Use four-digit years. Haven't we learned our lesson? (If you worked with computers from 1995 to 1999, you felt the pain as work came to an almost complete halt while we pored over every system with microscopes, looking for date flaws in the designs of our systems. Don't make a Y2.1K problem! PLEASE!!!)

Relative Time

Another way you can modify a time specification is to apply a relative time to it. The format of a relative time specification is **+ <count> <time units>**, where “count” is simply a number and “time units” is one of “minutes”, “hours”, “days” or “weeks”.

So, you can say:

```
at 7pm + 2 weeks
```

and the programs will be scheduled for two weeks from today at 7 p.m. local time.

One of the most common forms is this:

```
at now + x units
```

to specify a program or programs to be run so many units from now. Something I often use this for is in shutting down my home machine's dial-up connection from work. I dial in before I leave for work, and then I kill it before my wife gets home (I'm too cheap to buy a second line). I use ssh to log in from work, and I like to close all my windows cleanly, so I frequently do something like this:

```
# ps fax | grep wvdial
  599 ?        S      0:00      \_ wvdial
  875 pts/2    S      0:00      \_ grep wvdial
# at now + 10 minutes
at> kill 599
at>
warning: commands will be executed using /bin/sh
job 9 at 2000-04-17 16:30
# exit
$ exit
```

I then have ten minutes to disconnect cleanly from my home system before my phone connection gets dropped.

Runtime Environment

Note that the plain old Bourne shell is used for all commands run by at. (Also note: I had to type ctrl-d, the *nix EOF character to close the interactive at session. More on this in the section on the at command line. This is just one factor affecting the behavior of at scheduled commands. Here are some other facts to bear in mind. The present working directory, environment variables (with three exceptions, see below), the current userid and the umask that were in effect when the at command was issued are retained and will be used when the commands are executed. The three environment variable exceptions are **TERM**, **DISPLAY** and "_" (which usually contains the last command executed in the shell). The output of the commands is mailed to the user who issued the at command. If the **at** command is issued in an **su** shell (meaning, if you "became" another user), the output mail will be sent to the login user, but the programs will run under the su user.

Permission

The ability to use **at** is controlled by two files: /etc/at.deny and /etc/at.allow.

The /etc/at.allow file is checked first. If it exists, only user names in this file are allowed to run at. If the /etc/at.allow file does not exist, then the /etc/at.deny file is checked. All user names *not* mentioned in that file may run at.

If neither file exists, only the superuser may run at.

Command Line

The at command runs either the commands passed on standard input (passed in through a pipe, or typed at the "at>" prompts as in the example above), or it runs the commands specified in the file named by the **-f** parameter.

The general form of the at command line is:

```
at [-V] [-q <queue>] [-f <file>] [-mld] <TIME>
```

where “queue” is a queue name. Queue names are letters, a-z or A-Z. See the section called “Queues” for more details.

“file” is the name of a file containing commands to run.

“TIME” is a time specification as discussed in detail above.

The remaining switches are -m (send mail to the user when the job is complete, even if no output was produced); -l (an alias for **atq**. See the atq section below); -d (an alias for **atrm**. See the atrm section below).

The atq Command

The atq command lists jobs queued by the current user (unless run as superuser, in which case pending jobs for all users are listed).

Here's a sample:

```
mars:20:~$ atq
5       2000-06-20 15:00 a
6       2000-07-04 15:00 a
10      2000-04-24 14:33 f
mars:21:~$
```

The first column is the job number, followed by the scheduled run time, followed by the queue. In this case, two jobs are in queue “a” and one in queue “f”. See the section on queues for more information.

You can use the **-q** switch to look at jobs only in a particular queue.

The atrm Command

The **atrm** command is used to delete jobs from the atq. For example, consider the queue in the atq example above. The following session illustrates the use of atrm:

```
mars:21:~$ atrm 6
mars:22:~$ atq
5       2000-06-20 15:00 a
10      2000-04-24 14:33 f
mars:23:~$
```

You may list any number of job numbers on the command line.

The batch command

The **batch** command is a variation of **at** that, rather than scheduling a job for a time in the future, submits a job now, but that job will not start until the system's load average falls below 0.8. What is load average? The simplest way to think of it is the number of processes that are waiting to run. Most of the time, programs are idle, waiting for hardware or for input, or waiting for the kernel to complete a request. When a program actually has something to do, it is in a runnable state. If the system is not busy, the kernel generally gives control to such a program right away. When some other program is in the middle of running, the program that has just become runnable must wait. The instantaneous system load is the number of runnable processes that are not running. The load average is an average of this instantaneous load over a short period of time. Thus, a system that is below 1.0 load average has some idle time. A system that is at and hovers near 1.0 is fully busy, and at theoretical maximum capacity. A system that is over 1.0 has no idle time, and processes are waiting for a chance to run. Note that this does not necessarily mean the system becomes perceptibly slower to users, but it does mean the maximum capacity of the system has been reached and programs are running slower than they might on a less busy system.

The batch command schedules a job for “right now”, but will delay the start of the job until there is idle time (load average less than 0.8) on the system. Note that this test is for starting the job. Once it is started, it will run to completion, no matter how busy the system becomes during the run.

Queues

Note that this section is quite Linux-specific. Other UNIX operating systems I have used have queues, but they are different from those documented here. Always consult local documentation. AIX doesn't work this way, for example.

Queues are a way of grouping jobs together in separate lists. They are named from a-z and A-Z. The at command by default puts jobs on queue “a”, whereas the **batch** puts jobs on queue “b” by default.

Queue names with “greater” values run at higher “niceness”. Nice values are a way that Linux (and other UNIX systems) set job priorities. The default nice level of a job is “0”, which means “normal”. Jobs can have nice values from -20 (highest possible priority) to +19 (lowest possible priority). Only the superuser can give jobs a negative nice value. We won't say anymore about nice here, as a discussion of the kernel scheduler is well beyond our scope. Just know that jobs in the “z” queue run at a lower priority (and thus slower and with less impact on other running jobs) than do jobs in the “a” queue.

Jobs that are running will be in the “=” queue, which is reserved for running jobs.

Queue names are case sensitive! Rembember, there are a-z queues and A-Z queues. The A-Z queues are special. If you use at to put a job on a queue with a capital letter, then the job is treated as if it were submitted to the **batch** command at the run time instead of the **at** command.

In other words, putting a job on an uppercase queue is like combining at and batch. When the job runs, it runs immediately if the load average is below 0.8, otherwise it waits until the load average falls below that point. In no case will the job start before its scheduled time.

Phew! All of that and we still haven't looked at the dæmon that takes care of all this! I hope you are beginning to see that “at”, while not a complete batch processing system, certainly provides a great deal of capability.

How They Work

The at and batch commands put jobs into the at queue. What is the at queue? Well, there is a directory, /var/spool/at, which is accessible only to the dæmon user and the superuser (*everything* is available to the superuser). For each job, there is a file in the directory. The file is a shell script that sets up the environment and umask, cd's to the working directory and then runs the programs specified to at/batch in succession.

The commands go into the shell script exactly as they were typed/piped to at. Each is run in turn. If you used **&**, **&&** or **;** to background jobs, or make jobs dependent on one another, these will be observed.

Important note! The shell /bin/sh is used to run these jobs. If you normally use some other shell, such as tcsh, be aware that you can't use the semantics of that shell because /bin/sh will be used instead.

The atd Dæmon

At this point, documenting the dæmon is rather anticlimactic. The atd dæmon examines the /var/spool/at directory. The names of the files actually encode their runtimes, queues and batch vs. at status. These files are shell scripts that set up the environment and run the job as described above. Output from the jobs is temporarily stored in /var/spool/at/spool until the jobs are completed, upon which the output is mailed to the invoking user.

At Summary

at is less widely known than cron, but is in many ways the more powerful tool.

Using crontab

Potentially every user on the system has a **crontab**, which is a portmanteau word made from CRON TABle. The command to create, examine and modify crontabs is called crontab.

There are four ways to invoke crontab.

```
crontab <file>
crontab -l
crontab -r
crontab -e
```

Generally, crontab works on your own crontab. All four forms accept the -u option followed by a user name. In most cases, you will be able to view and edit other users' crontabs only if you are the superuser. You might want to check your system security if you are able to edit another user's crontabs. You probably have some problems!

The first form stores the named file as the crontab, replacing any current crontab. The second form dumps the current crontab to stdout. The third form removes the current crontab. The fourth form opens the current crontab in the editor specified by the VISUAL or EDITOR environment variable.

If you want to experiment with your crontab, it's a good idea to do a

```
crontab -l working-crontab
```

to save your current crontab if any, then use

```
crontab -e
```

to modify your crontab in your favorite editor. you can always use

```
crontab -r working-crontab
```

to put everything back the way it was.

At this point, you may be wondering what a crontab looks like and what it does.

Basic crontab Format

A crontab is a list of program command lines along with a specification of when to run that command line. It is a whitespace-delimited file with a newline between commands. Blank lines and lines beginning with a pound character (#) are ignored.

The fields are:

```
minute  hour   day of month    month    day of week    command
```

Any of the time fields may be an asterisk (*), which means “every”. Thus, an entry of:

```
* * * * * fetchmail
```

Will run fetchmail once a minute, every minute of every hour, every day.

Ranges of numbers are allowed. So:

```
* 8-17 * * 1-5 fetchmail
```

will run fetchmail once a minute, between 8 a.m. and 5 p.m., Monday through Friday (0 or 7 represents Sunday).

Lists are allowed. Thus:

```
0,20,40 * * * 1-5 fetchmail
```

will run fetchmail at the hour, at 20 past, and again at 40 past the hour every hour of the day, Monday through Friday.

Step values are allowed after asterisks and ranges. They are of the form <range>/<step>. So,

```
*/5 8-17/2 * * * cp /var/log/* /log/backup
```

will run that cp command (just in case you had started thinking you could run only fetchmail) every five minutes in the 8 a.m., 10 a.m., noon, 2 p.m. and 4 p.m. hours of every day.

Finally, names may be used for months (jan-dec, case insensitive) and days of the week (sun-sat, case insensitive). The Red Hat man pages claim that you can't use names in ranges, but I gave it a try myself and it appeared to work correctly.

Runtime Environment—Advanced crontab Format

This is the area that confuses users of cron the most. They specify commands they run every day from their interactive shells, and then they put them in their crontab and they don't work or they behave differently than they expected.

For example, if you write a program called “fardels” and put it in &HOME/bin, then add $HOME/bin to your PATH, cron might send you mail like this:

```
/bin/sh: fardels: command not found
```

The PATH cron uses is not necessarily the same as the one your interactive shell uses.

It is necessary to understand that the environment in which cron jobs run is *not* the environment in which they operate every day.

First of all, none of their normal environment variables are initialized as they are in their login shells. The following environment variables are set up by the cron dæmon:

```
SHELL=/bin/sh
LOGNAME  set from /etc/passwd entry for the crontab's UID.
HOME  set from /etc/passwd entry for the crontab's UID.
```

We've been holding out on you. There's another kind of entry allowed in your crontab file. Lines of the form i**name=value** are allowed to set environment variables that will be set when jobs are run out of the crontab. You may set any environment variable except **LOGNAME**.

An important one to note is **MAILTO**. If **MAILTO** is undefined, the output of jobs will be mailed to the user who owns the crontab. If **MAILTO** is defined but empty, mailed output is suppressed. Otherwise, you may specify an e-mail address to which to send the output of cron jobs.

Finally, any percent sign in the command portion of a job entry is treated as a newline. Any data which follows the first percent sign is passed to the job as standard input, so you can use this to invoke an interactive program on a scheduled basis.

Permissions

The ability to have and use a crontab is controlled in a manner very similar to the **at** subsystem. Two files, /etc/cron.allow and /etc/cron.deny, determine who can use crontab. Just as in the case of at, the cron.allow is checked first. If it exists, only the users listed there may have cron jobs. If it does not exist, the cron.deny file is read. All users except those listed there may have cron jobs.

If neither file exists (and this is quite unlike “at”), all users may have crontabs.

The cron Dæmon

There is hardly anything to document here. The cron dæmon (which is called either cron or crond) takes no arguments and does not respond to any signals in a special way. It examines the /var/spool/cron directory at start-up for files with names matching user names in /etc/passwd. These files are read into memory. Once per minute, cron wakes up and walks through its list of jobs, executing any that are scheduled for that minute.

Each minute, it also checks to see if the /var/spool/cron directory has changed since it was last read, and it rereads any modifications, thus updating the schedule automatically.

System crontab

I've led you through a merry dance so far. I've got you thinking that only users have crontabs, and that all scheduled jobs run as the crontab's owning user. That's almost true. Cron also has a way to specify crontabs at a “system” level. In addition to checking /var/spool/cron, the cron dæmon also looks for an /etc/crontab and an /etc/cron.d directory.

The /etc/crontab file and the files in /etc/cron.d are “system crontabs”. These have a slightly different format from that discussed so far.

The key difference is the insertion of a field between the “day of week” field and the command field. This field is “run as user” field. Thus:

```
02 4 * * * root run-parts /etc/cron.daily
```

will run “run-parts /etc/cron.daily” as root at 2 minutes past 4 a.m. every single day.

Final Notes

There you have it. While Linux does not ship with a mature and complete batch process management tool, still the combination of at and cron permit considerable flexibility and power.

Bear in mind that we have covered the Linux versions of these tools as shipped with most current distributions. While just about every UNIX system on the market has these tools, some things vary.

Expect at queues to be different. Not all crons support names or ranges. Most do not support lists of ranges or the increment feature. No other cron with which I am familiar supports setting environment variables in the crontab. I don't think any other **at** supports “teatime” as a time specification.

This boils down to a basic piece of advice. Always check the local documentation. If in doubt, experiment.

Resources

email: mschwarz@sherbtel.net

Michael Schwarz (mschwarz@sherbtel.net) is a consultant with Interim Technology Consulting in Minneapolis, Minnesota. He has 15 years of experience writing UNIX software and heads up the open-source SASi project. He has been using Linux since he downloaded the TAMU release in 1994, and keeps the SASi project at http://alienmystery.planetmercury.net/.

Archive Index Issue Table of Contents

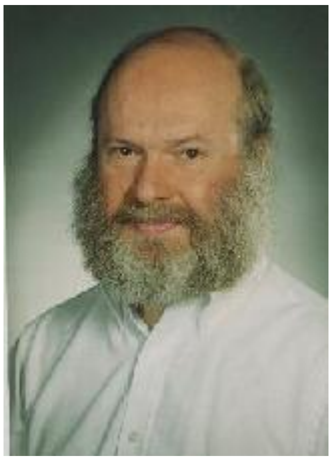
Advanced search

The Long View on Linux

Doc Searls

Issue #77, September 2000

Part One of Doc Searls' talk with the father of *Linux Journal*, Phil Hughes, about UNIX, GUI computing and the joys of an (almost) purely Linux publishing house.



I met Phil in 1990, not long after I met my wife, Joyce. It was kind of a package deal. Joyce collected interesting friends, and Phil was near the top of her list. She had met him earlier in Seattle while she was dating another UNIX geek. Both shared the curse of knowing they were smarter than pretty much everybody else (although Joyce is a bit more modest about it), along with a raft of common interests including travel, business, food and constant repartee spiced with affectionate insults.

Phil Hughes has been around computing longer than most humans have been on earth. He also knows a significant trend when he sees one, which is why he turned his attention to Linux before most of the world even noticed the Net.

In '93 and '94, he flattered me by including my e-mail address in a list that explored the idea of doing a free-software magazine. I followed as best I could (for a geek who was definitely not in the same caste as the other listees). But we were all surprised when Phil suddenly decided to do a magazine on Linux. What the hell was Linux? Phil knew.

In '94, when the 1.0 kernel came out, so did *Linux Journal*. I was amazed to see how steadily it grew, and how (as usual) Phil seemed to be right about Linux's inevitable triumph in the marketplace. Even though it was free, business would find it more useful than most of the costly stuff it competed with, he said, just like it did with the Net. And, as with the Net, commercial development was inevitable.

In early '99, Phil showed me StarOffice running on a Linux laptop with KDE. It looked like such a Windows killer that the possibilities blew my mind. Then he asked me to join *Linux Journal* to expand coverage of business issues, so I did. (This was right around the same time I started co-writing *The Cluetrain Manifesto*, which was kind of a double-whammy career move for me.)

Last August, Red Hat went public. Cobalt, VA Linux and Andover followed. Billions of dollars in new Linux wealth was created in less than two seasons. And then, it melted like snow at the end of winter. The mainstream press, especially the old PC rags, began tarring Linux as last year's fad. But, as Phil is quick to point out, the sum of Linux companies together adds up to a lot more value now than they did a year before. Again, the long view. Since we're both 52, it's one I understand.

Doc Searls: You were an old hand at UNIX when Linus was still a kid messing around with a Commodore VIC-20. What can you tell us about UNIX that seems to be forgotten as the Linux industry moves into enterprises where the most familiar computing worlds are Microsoft NT and Windows?

Phil: My first UNIX exposure was in 1980, when I convinced the company I worked for that a UNIX system and the C language would solve all our problems. This was with an engineering-based company that manufactured semiconductor test equipment.

What we needed was a software development environment where a handful of engineers and programmers were writing code to control the equipment we were designing. UNIX was the right choice, because we needed a cooperative environment where multiple users could share files; we needed compilers and assemblers; and we needed the building blocks to put together some support tools, such as programs that could reformat and download code into the machines we were building.

Someone, possibly Dennis Ritchie, described UNIX as a toolbox full of tools. When you were presented with a problem, you would divide it up into a set of smaller problems and then see if there were tools available to solve each smaller problem. It wasn't uncommon to write a little C code, maybe an awk or sed script, and a few calls to utilities such as sort and uniq and glue the whole thing together with a shell script.

This is much different from today's GUI approach, where you find one huge monolithic program and try to coax it into doing what you want. Maybe you can —but if you can't, you are at a dead end.

Doc: This is the stuff I learned (which might be an exaggeration, but let's pretend) in the excellent “Intro to Linux” class I took at SGI (http://www.sgi.com/support/) a few weeks ago. What amazed me was that the toolbox nature of UNIX made both creating tools and solving problems extremely easy. Sure, you can do sophisticated searches with the GUI Find command in KDE, GNOME and the PC operating systems, but the speed, flexibility and innate sophistication just aren't there. And they're slow.

Phil: This is always an issue when we hire new people. They can be productive sooner if we hand them StarOffice, but investing in them and teaching them shell commands, vi, troff and such makes them much more productive in the long run. The best example is in editorial, where everyone uses vi to prepare articles for layout. It would take so much longer if they used a word processor instead of vi. People like Darcy Whitman (Managing Editor of *Linux Journal*) who have been with us a long time are proof of this.

Doc: So GUI computing is secondary at *Linux Journal*. For the most part, the company runs in command-line mode.

Phil: Yes, but I'm not saying GUI computing is bad, just that it isn't always the solution. My big concern is that it is changing the way people think. Rather than looking at a problem and logically addressing it, I see people deciding that a spreadsheet is the solution to every problem because all they understand is spreadsheets.

Or, to go back to a common analogy, everything looks like a nail if the only tool you have is a hammer. You can keep adding features to your hammer, but you still have a hammer.

Doc: I see it as something like auto mechanics. You can do so much more if you know how the computer really works. And too few people do, just as too few people know how a car works. I think a UNIX jock looks at the OS in the same way a mechanically inclined driver looks at a car. If problems develop, chances are they're exposed and fixable.

Phil: It's exactly the same thing. You can even continue down this same line with the car. You don't have to understand physics to drive a car, but if you do, you will be better prepared to deal with situations such as when you are driving on slick pavement.

Doc: One of the virtues of a Linux box running a GUI such as KDE or GNOME is that you can go into terminal mode and work in the command line. You can't do that with Windows. And you can't do that with a Macintosh, although the next version of the Mac GUI will sit on open-source BSD with a mach kernel and command-line access. If you want, you can open a shell, get in there and do real computing stuff. I'll be interested in seeing if that makes any difference in the marketplace. Apple is already quietly running some of its heavy web servers on the new OS. This may be the result of Steve Jobs spending ten years in the UNIX world at NeXT.

Phil: You can get a shell on a Windows box, but there are many fewer capabilities and a lot less sophistication than with Linux. I seldom use Windows, but every time I do, the first thing I do is bring up a shell in case I want to do anything real, which I define as removing a file, moving a file, copying a file...you get the idea. Note that the most common thing I end up doing is copying files to floppy disks so I can take them to a Linux system and use tools such as sed and awk to work on them. vi isn't a problem; I always put vi on Windows boxes.

Doc: I remember *Linux Journal* growing out of an e-mail conversation—which I was improbably part of—about starting a free-software magazine. A couple of questions here: 1) who else was in that original group? and 2) what exactly happened?

Phil: In 1993, I had an idea to do a free-software magazine. While SSC was doing pocket references for UNIX and UNIX-related programs, I decided I wanted to do this magazine independent of SSC. There were six or seven of us involved in talking about the original idea. Early players included Arnold Robbins, author of some of SSC's products; Laurie Tucker, a friend in Atlanta; Gerry Hammons, a longtime friend and co-worker from years before; David Keppel, a UW computer science grad student; and Melinda McBride, another co-worker from a previous job.

I had set up a mailing list so we could all keep in touch. One day, I realized that to do a good free-software magazine, you would have to be like *Consumer Reports* and have no advertising. I did a little arithmetic, and realized we were about $9,999,900 US short of the $10 million I estimated it would take to start the magazine. I posted to our list what I initially thought was a joke: why don't we just cover Linux?

Everyone thought it was a great idea. So, I decided to think seriously about it, and agreed. It had balance: it wouldn't cost much to do, but there wasn't much market, either. Not sure if that was good balance, but it was balance.

I posted some questions on comp.os.linux, the only Linux Usenet newsgroup at the time, and received very favorable responses. Thus, it looked like we had an audience, so I set the wheels in motion.

Things looked pretty good, and we were almost ready to dive in full-force when some disasters struck. The most significant was that my friend Gerry died. He was doing some programming for the infrastructure we needed, plus he was going to invest. It was obvious that we needed to put the project on hold, and I made another Usenet post to let people know there was a very large bump in the road.

At the time, Bob Young had started a publication called *New York UNIX*. Someone saw the post about our problems, and gave it to Bob. Bob contacted me, and suggested that he could be publisher if I could take on the task of editor. We decided to go for it.

After two issues, it was clear to both Bob and me that this wasn't the right relationship. We split the responsibilities, with him assuming the debt for printing already done, and I took the subscription obligation—the 926 readers who had paid for 10 to 12 more issues of the magazine. I rolled *Linux Journal* into SSC, and we ran with it.

Doc: So it's true you gave Bob Young his start with Linux?

Phil: Yes.

Doc: That checks out. When I talked to Bob a couple weeks ago, he gave you all kinds of credit for getting his Linux career started. When I told him “Phil says he taught you how to spell Linux”, he said, “It's true! I owe a lot to Phil. He even gave me one of my favorite metaphors: that closed source is like a car with the hood welded shut.”

Phil: While he gives me credit for the hood metaphor, I really don't remember using it. But, I do like it.

Doc: One of the most interesting things to me about *Linux Journal* is how it seems to be written, to a large extent, like Linux itself. Many of the features and columns come from readers—members of the Linux community. And most of these are people solving real problems. I get a sense with both Linux and *Linux Journal* that we're all working together to raise a barn.

Phil: It has exactly that feeling. In the early days, many authors were surprised that we actually paid them to write for us. Some just donated the money to the Free Software Foundation or other projects.

Doc: Y'know, the e-world is full of all these new acronyms around commerce: B2B for business-to-business, B2C for business-to-consumer, B2E for business-to-enterprise. So I have an acronym for *Linux Journal*: G2G for geek-to-geek. Because that's the model that is making the new world, and the great irony is that the New Economy folks haven't got a clue about it. Which makes it that much more subversive.

Phil: This isn't new. Remember, UNIX was born because 30 years ago, a couple of geeks wanted to play a computer game. I wonder if 30 years from now, most people will not even know that Linux was a student project. Actually, I wonder how many don't know that now.

Doc: How do you see the free/open movement today? How has it changed? Is it for the better?

Phil: There are two things here: open and free. We had open and free software on mainframes in the '60s and '70s. If you bought a mainframe, it came with an operating system complete with source code. When you were shelling out millions of dollars for the computer, or more commonly, leasing a multi-million-dollar computer system, why not give you the code? It wasn't like you would go to Radio Shack, buy another computer and copy the OS.

A combination of the price drop in hardware, generic hardware and Amdahl Corporation forced a change. First, Gene Amdahl, who had designed IBM's mainframes, started his own company to make machines like the IBM mainframes. By “like”, I mean they had the same instruction set and could, therefore, run the same operating system. IBM now had to unbundle and charge for copies of the OS so they could make money off the sales to Amdahl users.

Doc: I've never heard this analysis before. It's really fascinating. What you're saying is that, historically at least, software really did want to be free. And Bill Gates invented an industry that didn't need to be there. This is what Neal Stephenson suggests in his great little book, In the Beginning Was the Command Line.

Phil: Yes. The OS was something the computer manufacturer had to include with their hardware. Without it, they couldn't sell the hardware.

Doc: You think IBM is going back in that direction with Linux? They're communicating rather clearly that they don't much give a shit anymore about selling OSes, maybe because that was never the idea in the first place. They sell iron. Why not sell iron that's easier to deploy because it runs a universal OS?

Phil: Yes, I do. IBM abandoned their web server in favor of Apache. I think they know people don't want to buy an OS—they want to buy a solution. Actually, I think they knew this when the PC came out. I feel that Microsoft managed to confuse the issue, and IBM fell for it for a while.

The first cheap and generic computer was the IBM PC. Other manufacturers jumped on the bandwagon to make inexpensive clones. That meant the OS cost, again, could not be bundled with the hardware cost. Besides the end of "free" in terms of price, the openness of the past was gone because there was direct competition.

The GPL is an attempt to force freedom back into the mix. It is only one example, with the BSD license being another. Each has its advantages and disadvantages; my point is that we used to have this freedom, but because hardware changed, we now had to do something different in order to get free —both in terms of price and freedom—back into computing.

Explaining the BSD license to a businessperson is fairly easy. They will remain skeptical, but it is easy to tell them you can get something for free, do whatever you want with it and then sell it.

Doc: Because you have to credit only the originator.

Phil: Exactly. You can build a proprietary product, and don't have to pass along your additions or improvements. The GPL is a lot harder. They don't understand why it makes sense to make the changes and then give them away. Or that they are expected to give away whatever changes they make. In other words, not to act like they own it exclusively.

Context is the issue. It's the same as IBM giving you the OS with the hardware; you need to show that businessperson how giving away the OS will sell something else. Sorta like giving away a keychain at the car dealer, and then hoping you buy a car to go with it.

As each new vendor enters the Linux space, you need to educate them. You need to show them success stories. Eventually, they will get it.

Doc: I remember when the Net first became obvious to business, right after Netscape released the next incarnation of Mosaic. Nobody knew what to do with it, because no company owned it. The thing was as free and open as air.

Now, it seems like every business on earth has .com after their name, and the Net has utterly changed the context of their work. Are we seeing the same thing happening with Linux? It seems to me that the Net and Linux are two aspects of the same thing—a new world that works for three reasons: 1) nobody owns it, 2) everybody can use it, and 3) anybody can improve on it. This turns out to be great for business, as long as the business doesn't try to throttle this public goose that lays golden eggs.

Phil: There is certainly a similar evolution. That is, Linux was ignored by many for a long time, even though it really was viable. A few years ago, SCO did a fax spam about how you could get a $50 trade-in on your Linux distribution to "upgrade" to SCO UNIX. Well, anyone using Linux at the time was aware that Linux was a better product than SCO UNIX. It's pretty obvious this effort didn't work, as we can tell by SCO's market share.

So, yeah, Linux is a tool that can make money for a lot of different people, because it is open and free. Many companies have played along very well to help the goose continue to produce the eggs. The one company I believe got off track for a while was Red Hat. They seemed to want to make their name mean Linux. It wasn't that they did not continue to give back to the Linux community; the problem was their marketing wasn't showing Red Hat consumers that there was a lot more to the Linux community than Red Hat. Sorta like Microsoft trying to make people think that they were the only game in town.

At this point, I think the situation has corrected itself. You see books on SuSE, Caldera, Debian and other flavors of Linux in the stores. That means the distributions can compete on how well they fit the needs of each consumer, rather than on a visibility issue.

Is this change for the better? If you believe Linux should grow in market share, yes. We used to be a bunch of geeks doing what we want. Today, Linux is business, and to keep it growing, we need to keep getting more businesspeople on board. It is just that along the way, there will be various frustrations. The most common is when a vendor doesn't understand the point of open, and finds a way to get around licensing such as the GPL.

Doc: I've found it very hard to talk rationally about Microsoft inside the Linux world. Too many distinctions get collapsed. For example, the literal distinctions between open and closed source get collapsed with moral distinctions between right and wrong, and with qualitative distinctions between good and bad software. Eric Raymond is fond of saying he just likes software that doesn't suck. But a lot of the world likes closed-source software for which there are no alternatives, whether that software sucks or not. And most Microsoft users, on the whole, don't think Office sucks.

Phil: I think two things happened here. First, for many people, the first word processor they used was Microsoft Word. They learned it, and see anything different as something new they have to learn. So, whether MS Word is sucky or not, they know how to use it. When I was learning vi, I had been using ned, a screen-oriented editor on UNIX, but then ned wasn't available for the new computer SSC had bought. I hated vi and loved ned. What happened, however, was in about two weeks, while I still hated vi, I realized I was editing faster than had ever been possible with ned.

The second is the charm of the GUI. Microsoft had the “in” that allowed it to produce the best GUI-based editor, because they had the inside track on the GUI. For the casual user—someone who might write a letter a day—Word was adequate, and easier to remember how to use. Even I can use Word; it's just that I can get more work done in vi.

The person I started SSC with, Irene Pasternack, left to “get a life” that included raising a family—something you don't want to do when you are starting a company. Irene has consulted at Microsoft for close to 15 years doing tech writing. A year or so after she went to work for Microsoft, her boss told her that she did eight times as much work as his other employees and asked if she had any idea why. She said, “they use Microsoft Word, I use vi.”

Doc: A friend of mine used to program for a familiar dot-com company that runs on Windows NT. He said he constantly had to deal with memory leaks in that OS, and added, “we're putting up 60-story buildings and we don't know if there's rebar in the concrete.” Then he added, “Actually, we know there isn't rebar in the concrete, and we're going ahead anyway, knowing we just have to fix problems constantly.” Open-source software doesn't put geeks out of business; it just gives them more room to do their business. And gradually, Linux is proving that. Which is why I think that in five or ten years, the software and construction businesses will look very similar, with many more builders, architects and designers. Plus many more suppliers. The main difference will be that no one pre-fab supplier will control the business anymore.

Phil: The UNIX community grew up as computer capabilities grew up. The 30-year history of UNIX means there were people making those buildings without foundations, because there wasn't any room for the foundation. But they recognized the problem, and when room became available, they added as much foundation as would fit. Thus, the best possible building was produced at each step along the way, and we built a workforce of people who understood how to grow with industry changes.

On the other hand, the Microsoft community knew the foundation was missing and there was nothing they could do about it, so they realized the whole building was disposable. Of course, being disposable means you can sell the new product next year. I think I am beginning to realize that the mess being created was not a plot—just circumstance.

Doc: You've probably watched the climbing Linux adoption curve closer than anybody. Why is Linux finding universal adoption where other free UNIX systems (such as the BSDs) did not?

Phil: As much as Richard Stallman and others would like to credit this adoption with the difference between the GPL and the other software licenses, I just don't think it is the case. Linux got out there, and it worked. While the various BSD camps were fighting over which was the one true BSD, people were using Linux and seeing that it was a solution.

Once Linux had the inertia (and *Linux Journal* certainly helped build that inertia), Linux would continue to roll along. I also have to give a lot of credit to Linus' management skills. Getting hundreds, possibly thousands, of programmers around the world all to cooperate on the development was an amazing effort. Getting them all to do it for free was way beyond amazing.

Even before I met Linus, I was impressed by his maturity and sobriety. The guy just seemed to have this highly informed and calm perspective, like a judge. Not your average 20-something dude; not your average geek, either. Made me wonder what would have happened, say, if Bill Joy had stayed with Berkeley UNIX. Instead, Bill is doing fun stuff, but it's all stuff that's owned by Sun. Java. Gini. Solaris. Sun is a good company, but in their own way, they're as closed as Microsoft.

Linus has introduced a new model to the computer industry. Or, maybe I should say that Richard Stallman proposed a new model, and Linus was the right man to successfully implement it. In any case, it has worked, and that may help convince companies such as Sun to at least move in the direction of more openness.

Doc: When did you meet Linus?

Phil: I first met Linus in 1994. It was at a Linux-related party in suburban Washington, DC. When I got there, Linus was there along with a few others. They were talking about some technical Linux-related issues. What I saw was a person who could participate in the conversation, rather than act as the boss. At this point, I understood why Linux was moving forward so rapidly.

Doc: Linux has spread like wildfire, and *Linux Journal* has grown right along with it. How have both Linux and *Linux Journal* changed since both came out at v.1.0? And are you happy with all those changes?

Phil: The short answer is that today we are both better products. But, more importantly, our focuses have changed. Linux went from a geek OS to a major contender in the business and commercial market. As that was happening, *LJ* went from being a developer magazine to offering a lot more that would appeal to the commercial user.

Over the years, we have received letters from readers who credit *LJ* with convincing their bosses that Linux was real. At times, I felt *LJ* was portraying a more professional image than Linux itself; at other times, I felt it was the other way around.

I can't say I am happy with all the changes, but I also realize they needed to happen. While it used to be more fun to play with Linux than to publish a magazine, I also realize that for Linux to move forward, it is necessary for it to become more commercial. We would not be seeing drivers for all the new hardware if we didn't have the kind of growth we have.

Doc: In the last year, Linux has become a very hot topic. It seems to me this was for two reasons. One was the popularity of the OS itself. The other was the popularity of new Linux companies in the stock market. These are two very separate concerns that were covered by the mainstream press as if they were one—at least while Linux stocks were hot. Now we're seeing the topics start to separate a bit, now that dot-coms of all kinds (including Linux stocks) are dropping from the sky like a plague of frogs. Linux stocks were especially hard hit. It hardly seems fair.

Phil: The growth in the popularity of the OS has been amazing. Here, I think the U.S. Department of Justice and Microsoft both get some credit. The DOJ antitrust suit against Microsoft brought the concept of an operating system into the minds of much of the public. Prior to the DOJ action, most people only understood “computer” and “program” together, where “program” was commonly spelled Microsoft Word. They bought a computer, and added Microsoft Word or Microsoft Office in order to make it do things. That's all very different today.

The Microsoft credit has to do with NT/Windows 2000. A lot of companies were betting on NT or Windows 2000 offering the server side of their computing. The harder people look at these products now, the more they seem to realize there are alternatives such as Linux, and quite bluntly, Linux has proven to be a better choice.

Windows 2000 is finally out there, but people are sick of waiting. If Windows 2000 works, fine. But if not, they are likely to pick an alternative, and Linux seems to be at the top of that list.

Doc: Here at *Linux Journal* we use Debian, which is the largest noncommercial distribution. Are we doing it for agnostic reasons (not wanting to favor any particular advertiser), because it's better, or both?

Phil: Note that we used to use Slackware. I like the idea that Debian seemed to be the best fit for us, because we can then be agnostic, but we chose it mostly because it was a better solution. Debian has always had a good dependency system, plus it was designed so you could upgrade without a reboot. While Debian development might fall behind the commercial distributions, I am not a fan of upgrades without a reason, so it has tended to keep up with our needs.

Doc: What do you run personally?

Phil: On my personal systems, I run a whole bunch of different distributions for various reasons. For example, I have SuSE on one laptop and Caldera on another.

Linux, the kernel and hundreds of utility programs are virtually the same in all the distributions. Sure, SuSE will release a new distribution this week with a newer kernel than Red Hat. And next week, Caldera will be ahead, and so on. But they are mostly the same.

What they add to all this GPLed software is a combination of additional programs, such as an evaluation copy of some commercial software, and an installation method. All the distributions I have seen differentiate between GPLed software and commercial software, so I don't see this as a serious issue. As for the installation methods, most distributions have GPLed this code as well.

Doc: Let's talk about Linux on the desktop. Today, there still isn't a single non-proprietary desktop application suite to compete with Microsoft Office. Enterprise-customer types tell me Linux won't make it on the desktop until it has a truly competitive (and compatible) office suite. What will break the logjam here?

Phil: StarOffice, while not open source, is free and is forcing the issue of compatibility. Sun, the owner of StarOffice, needs a generic and compatible office suite in order to restrict Microsoft's ability to rewrite the standard every year or two. If StarOffice is successful in preventing the standards rewrite, then I expect other packages will appear. The good news is that StarOffice is available for MS Windows; and, as I understand it, e-machines is shipping it on their boxes. This should help market penetration.

Doc: If you see Sun as the only player with the clout to really compete with Microsoft in office suites, what do you make of Applix's and Corel's chances? And do you think it's possible that some group of open-source developers, KDE for example, will come up with a practical open-source office suite?

Phil: Corel doesn't have a complete suite yet, and I don't think they will until after the issue has been forced anyway. Applix has had an office suite out there for a long time, but I don't think they have the clout needed. Sun wins on image.

Linux will creep into the office desktop on a company-by-company basis. Take a bank, for example. While there are lots of computers in banks, they don't do a lot of different tasks. Armed with a word processor, a spreadsheet and some program to access their proprietary software running on a server, you could quickly convert a bank over to Linux desktops. This will happen because of cost issues.

Doc: This is what lots of people want to see with Linux appliances. Should be interesting to see how that goes.

Phil: Besides cost, the ease with which you can add things to Linux and the fact that Linux for an appliance can be made really small are two more pluses in favor of Linux in this market. As much as BeOS is a great OS, I think it is going to be hard for Be, Inc. in this market because of Linux.

Doc: Do you give Be much of a chance in any case? How about if they open source the OS?

Phil: I think they bailed out of the desktop at the wrong time. For some high-end IAs, they may have a market, but I don't see them getting any significant percentage. Qt without X (this is new) makes Linux more viable here.

Doc: Let's talk about loves & hates. Why do you hate Perl and love Python? Why do you hate Emacs and love vi? These are often religious issues, but I'm interested in what makes for wise and forward choices.

Phil: They generally are religious issues. In the days of less-powerful computers, you could make the argument that vi is small and Emacs is a pig, but today it doesn't matter. I learned vi in 1983. I don't even think about which keys to hit to do a task—it is just magic now. I have no reason to change.

A newcomer should try both. Some people, for example, really can't handle a moded editor like vi. Others see the advantage of fewer keystrokes. Yes, we have an office full of vi users, but I expect some places have offices full of Emacs users.

Perl vs. Python is different. Historically, Perl evolved from UNIX commands such as sed and awk, whereas Python was designed. Python was designed to be an object-oriented language, whereas OOP features were tacked on to Perl. If you are an old UNIX programmer, then Perl may make sense to you; but for a non-UNIX person, I can't see it.

Doc Searls (info@linuxjournal.com) is Senior Editor of *Linux Journal* and co-author of *The Cluetrain Manifesto*.

Interview with Andrew Leyden—CEO of PenguinRadio

Jason Schumaker

Issue #77, September 2000

One of the coolest Internet appliances being developed comes from PenguinRadio.

One of the coolest Internet appliances being developed comes from PenguinRadio. The company was founded by Andrew Leyden and is working toward providing the world with an Internet-based car radio that will allow access to thousands of stations—from anywhere in the world. Imagine listening to your favorite radio station from Madrid (if you have one) while driving away from the latest black tie affair at the Gates mansion, outside of Seattle. This will be possible, eventually, thanks to PenguinRadio.

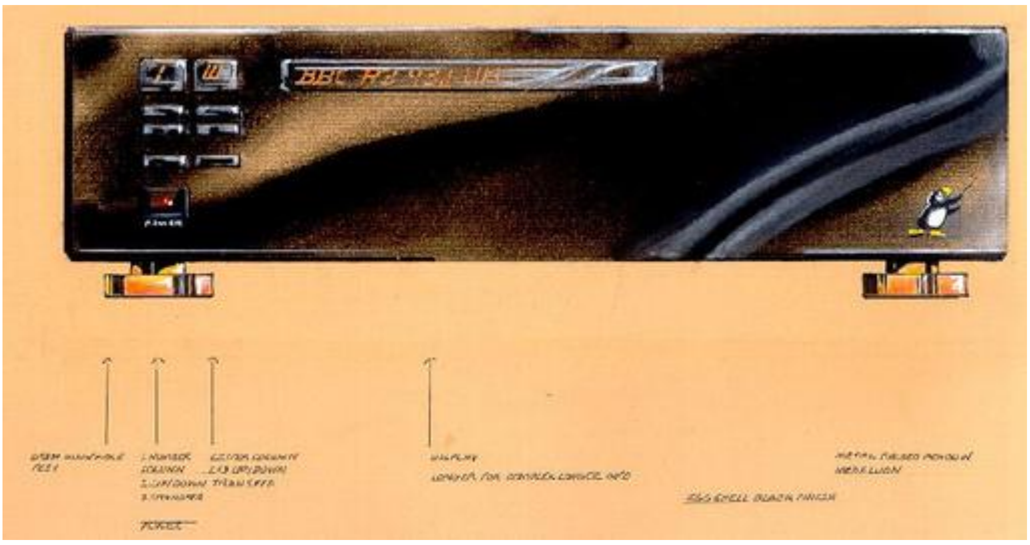


Figure 1. Prototype of PenguinRadio

While they are still in the development and venture capital-raising stages, the prospects are exciting. By partnering with wireless communications developer Ineva.com, PenguinRadio may be able to differentiate itself from Kerbango, its nearest competitor. Ellipso, Inc., called a “sister company” by Ineva, will provide the satellite technology that makes listening to any station, anywhere, possible. Ellipso, as mentioned on their web site, has three patents covering their “elliptical orbit and deployment characteristics”. They have found a way to link satellites high above the earth in order to provide uninterrupted service nearly anywhere in the world.

I talked with PenguinRadio CEO Andrew Leyden about the latest developments at PenguinRadio.

Jason: Why Linux?

Andrew: There was never a debate as to the operating system for the PenguinRadio. Not only did Linux meet our needs technically (scalable, stable, inexpensive) but it also fit our ideals philosophically. We believe very strongly in working with others to develop new products that are going to bring the Internet (and Linux) to more people.

Jason: How has Linux worked so far? What problems have you encountered?

Andrew: When we first started developing our box, we were occasionally stumped by hardware conflicts and a lack of drivers for certain parts of the radio, but the Linux community has proven to be a great resource for us in helping us develop this product. We've e-mailed and chatted with people all over the world who have offered suggestions, given guidance and pointed us to the right way to make a device like this become a reality.

One problem that has yet to be overcome is the view (in some quarters) that Linux is little better than shareware. We've demonstrated this not to be the case, but some of those with money and software necessary to make a box like this work are not totally committed to the idea of Linux. They'd like to see machines work on more established systems backed up by “professional” (i.e., paid) support.

Jason: You said that when first developing with Linux “we were occasionally stumped by hardware conflicts...” Could you provide an example?

Andrew: Some of the less expensive hardware we were experimenting with had poorly supported sound chips, making it difficult to produce glitch-free audio output at the $200 price point.

Jason: What operating system do you use at work/home?

Andrew: We've got a mishmash of machines around our offices and home. We use Macs for a lot of our web design work, Windows machines for some of our office functions, and Linux boxes to keep them all running straight. Our chief designer actually has a Linux box set up to maintain a web cam in his house so he can check up on his dogs while he's at work.

Jason: When is the product expected to be available to the public?

Andrew: Our goal remains sometime this summer or fall, but in order to make the device easy enough for my mother to use (i.e., she knows how to dial the phone) I have to spend more time making it easier to use.

People expect a radio to work every time they turn it on. Computers can crash and have blue screens of death, but a radio is something that you turn on and it works. If it doesn't, you throw it out and get a new one. We need to meet this expectation of a 100% fool proof device, which is a bit difficult.

Jason: What is the projected price range?

Andrew: We feel we can get these devices out in the $200-$250 U.S. price range.

This is what we consider the “one spouse” price point, i.e., one spouse can purchase without the permission of the other. Your mileage may vary on this...

Jason: What station would you listen to from, say, Cairns, Australia?

Andrew: We currently have about 50 Australian radio stations in our database, including a few from around Queensland, and are working to improve this number on a daily basis.

There's actually a very strong interest in a product like this in island nations like New Zealand, Australia and the U.K., where traditional radio has very real geographic limitations.

Jason: Could you please provide a bit of background information about yourself and your company?

Andrew: Most recently I served as Counsel to the Commerce Committee of the House of Representatives in Washington, D.C. This is the committee of jurisdiction over the telecommunications network, the Internet and all commerce in the United States. I conducted a number of oversight matters into technology, and it was while working with the committee that I came up with the idea of the PenguinRadio.

I was spending some time writing a memo, playing a streaming media feed in the background when I ran into a problem many people have confronted: the computer could not handle the decoding and the other task at the same time. There were blips and beeps and delays. Initially I said, “buy a new computer that is more powerful”, but then it hit me, “no, go the other way. Make a streaming media device.” I did a bit of research into the sector and decided this would be a great device to have in my own house, regardless of whether or not I sold a million units. I left the committee about a year ago, raised an angel round from some interesting investors and recently closed on my first VC round from the Internet Partnership Group.

Jason: Will you be releasing any of your work to the Open Source community?

Andrew: We haven't figured out what is going to be released and what is to stay with us, but I'm pushing hard to be generous to the Open Source community as they have been so helpful with us. We already allow other web pages access to our portal of thousands of radio stations and plan to work with other device manufacturers to allow them access to the work we've been conducting.

Jason: What sort of help do you need from the Open Source community, in terms of development work and so on?

Andrew: Comments, suggestions, rants, raves. Whatever they feel might be helpful, or any problems they've encountered in working on embedded systems are greatly appreciated.

Jason: Which Linux kernel are you using?

Andrew: We've been experimenting mostly with the 2.2 series.

At this stage, we want to use a fairly modern kernel and we're not as concerned about the kernel size, since memory is cheap and getting cheaper all the time.

Jason: Have you released the source code to your Linux port?

Andrew: We don't mean to sound evasive, but the streaming audio landscape is evolving and changing so quickly that we're not ready to release anything at this point. Our work is mostly in writing customized applications, so there's nothing really tricky or unusual with the distribution that we'll use.

Jason: Is there a demo of the radio available, or are you still in the development stage?

Andrew: We have built some prototypes, but there's not an official demo unit.

Jason: Who would you identify as your main competition?

Andrew: As far as the hardware arena is concerned, the company formerly known as Kerbango comes to mind. But that's just one dimension of PenguinRadio. We've partnered with Ineva.com to deliver Internet radio just about anywhere a satellite signal can be received. PenguinRadio is also geared towards the wireless community, as we deliver the sounds of the Internet via our stripped-down portal, phoneradio.com.

Jason: Thank you for your time, and best of luck.

Jason Schumaker (info@linuxjournal.com) has worked for *Linux Journal* for nearly two years. He is Assistant Editor and a staff writer and spends much of his time outside work injuring himself while playing various sports. He's a jock.

Compaq's Approach to Linux in Your Hand

LJ Staff

Issue #77, September 2000

Digital, Compaq and evolution bring us another embedded Linux solution.

Digital, Compaq and evolution bring us another embedded Linux solution.

While the Palm has always been the top dog of palmtops for Linux users and Yopy (see “Linley on Linux”, *LJ*, September 2000) is the first kid on the block with Linux inside, there is Linux brewing at Compaq as well. iPaq, the handheld pocket PC from Compaq, should have the attention of most Linux lovers. While the shipping version runs Windows Pocket PC, getting a Linux version of this 200MHz handheld with 32MB of flash RAM is as easy as surfing to the Compaq web site. And what's more, the iPaq owes its very existence to open source and Linux in general, and a research project called “Itsy” in specific.

As Dick Greeley, program manager for the Open Handheld program and a member of Compaq's Corporate Research team, said in a recent interview, Compaq's study of handheld computing devices—seeing what it was that made them so popular and what could be done to improve on some of the product's deficiencies—had it origins not in Compaq, but in Digital Equipment Corporation, which Compaq acquired in 1998. Getting Digital's significant research labs as part of the bargain, Compaq became the home for Digital's “Itsy” project. The “Itsy” project was an attempt to marshal many of the technological innovations in computing power, screen quality and memory capacity for the burgeoning handheld computer market. It is this project, started four years ago, that eventually produced the iPaq.

“Our goal in research,” said Dick Greeley, “is to look into the future, to look beyond the time horizon of the existing business units, so to that extent, we're looking out further than a lot of these groups do.” What he and his researchers saw was that issues of power management and user interface were key to developing handhelds that would be popular and able to exploit the developments that were leading to the incredible shrinking machine. “While all these devices could shrink,” Greeley noted, “our fingers can't, our eyes and ears can't...so we needed to start thinking about how the user interface of the future is going to work with these devices...what can and can't be done and how will people really want to relate to them.”

Building a better user interface for handhelds led the Itsy team to a number of interesting innovations, one of which was the addition of a 3-D accelerometer which enabled a user to measure when the Itsy unit was tilted or moved. What good was this? Greeley calls it a “rock and scroll” interface. “The idea is that as you flip the unit around in your hand, it would be basically programmable and would do different things for you,” he explained, using the example of being able to turn pages on a particular application simply by flipping the wrist. So giddy had the Itsy team become with the success of their early prototype device —which was about the size of a deck of playing cards—that they even loaded the popular action-shooter game, DOOM, onto it. Recalled Greeley, “I think my favorite aspect was how you cocked the shotgun, by flipping the unit forward and backward.”

Why did the Itsy team choose Linux as the development operating system? Dick Greeley offers a number of reasons. For one, like many developers, they wanted something they could play with—from the source-code level up—on their own, without any proprietary strings attached. Second, the fact of the matter was that the Itsy project was not a secret; numerous developers from the academic and research communities were also involved to one degree or another. This involvement also led to the founding of the www.handhelds.org web site and the Open Handhelds program, which has as its goal the continued support of Linux on the iPaq, making the device available to the research/ development community.

While the Itsy team choose Linux as their operating system, Greeley admits that, within Compaq, his research team is pretty “operating system agnostic”. He adds, “I think what's more critical than necessarily the OS choice is the notion of going open source, to be able to share and interchange the fruits of many people's efforts and to have access to a wide variety of applications that are out there, which makes innovation really possible.”

The transition from Itsy to iPaq took place when the newly Compaq'd Itsy research team went looking around the company for a new home for the project. Greeley says they found that home in Compaq's iAppliances Group, who had been working on an iPaq desktop PC and a new StrongARM-based handheld pocket PC. The iAppliances pocket device did not have the Itsy's 3-D accelerometer, but it did have the memory and expansion capabilities the Itsy researchers wanted. As Greeley said, “We proposed to them: Hey, let's launch this, what we'll call the Open Handheld program...We'll take our Linux port that we had originally done for Itsy, get it up to the latest revision of the Linux kernel, and at the same time, adapt it to work with the iPaq. This would be a great way to get people innovating on our platform.”

As for what might be called the Itsy/iPaq premium, Dick Greeley points to a number of things he believes help set his handheld apart from the rest of the pack. “First of all,” he says, “is the processing power. We've got a much faster processor than everybody else.” Second, he points to the iPaq screen, which he suggests is at least as good, if not better, than others on the market. “You can look at the screen in bright daylight and still see it,” he insists. “You could find your keys in the dark using the screen like a flashlight,” Greeley adds by way of example. The iPaq screen also has both a variable brightness control as well as an ambient light sensor that automatically adjusts to the environment. The unit also gets a maximum of 12-14 hours on its battery, but one of the key features as far as Greeley is concerned is the fact that the operating system (Microsoft Pocket PC standard or Linux, available from the Compaq web site) is stored in flash memory as opposed to being burned into ROM. This gives the user increased ability to load Linux on the iPaq by loading into flash instead of burning a new ROM. Says Greeley, “That proved to be an important thing for us and it's actually made our whole program possible.”

Only recently started, it is probably impossible to gauge how much Linux developer interest will flow toward the Itsy/iPaq project. Greeley himself notes that, with regard to the Linux port to the iPaq, “we don't have all the bells and whistles yet,” meaning there is no browser and the wireless Ethernet is enroute, but has not yet arrived. The Linux Itsy does feature the X Window System, however, and Greeley expects progress to come quickly. “We've started shipping the new iPaq in the last several weeks and there's been very good demand for them...One of the things we've heard a lot of the last week and a half is a lot of Linux people saying, “Finally! I've got my thing I can play with now! Thank you for following through!”--even though the product is not exactly the same as the Itsy.

Archive Index Issue Table of Contents

Advanced search

Linux in Embedded Industrial Applications: A Case Study

Luca Fini

Issue #77, September 2000

In order to allow the acquisition of diagnostic data from an existing industrial plant which was not designed for the purpose, a protocol converter gateway has been built by using an industrial PC running Linux.

A big turbocompressor in an oil extraction plant is controlled by a 15-year-old Digital Control System (DCS), which provides all the supervision functions, plus the logging of significant events and data onto a printer. The system has been running satisfactorily for years, but now the management would like to integrate the machine into the global plant SCADA system, where data from various machines are gathered and processed in order to provide integrated management of the entire plant.

The PLC-based architecture of the existing DCS is hardly expandable and doesn't allow, from either the hardware or software point of view, for the addition of the data transmission functions which would be necessary for the project. A redesign of the DCS system to add the required functionalities was considered an excessively high-impact solution for the purpose.

The solution proposed was thus to add to the existing DCS a sort of “protocol converter gateway”, i.e., a box which captures the printer data stream, analyzes the lines of text to extract data values and makes them available to the SCADA (see sidebar) system. The latter will periodically issue data queries to receive the data values from the gateway. For this part of the connection, the standard protocol Modbus RTU was selected because of its simplicity and because it was already widely used throughout the plant.

Definitions of Terms

System Architecture

From the hardware point of view, what is needed is a box with two serial ports: one to receive the printer data stream from the DCS and the other to receive data queries from the SCADA system. The box must run some program which decodes the printer data stream and replies to the SCADA queries.

The main challenge in this kind of application is represented by the particular environment: the plant is located in a geographically remote area and the site has all the problems which are usually found in industrial environments—dust, electrical noise, wide ambient temperature ranges, and so on. Industrial PCs are easily available on the market, but one must take into account a few peculiarities which are usually not found in other PCs:

- There is no fan on the CPU. Spinning parts are the weak points in a PC and the miniaturized CPU fan is particularly unreliable with time. The main fan cannot be easily avoided, and is accompanied by suitable filters. We can do without the CPU fan, provided that the clock speed is reduced to limit the heat dissipation needs.
- There are no disks. The hard disk is substituted by a solid state "Flash EPROM disk". The floppy disk drive may be included, but it will be used only for software uploading. This means that the available disk space is very limited, usually not more than 64MB, but in our case, just 8MB.
- Standard RS-232 lines are not suitable for serial data transmission in "noisy" environments. The more reliable RS-485 standard is usually adopted.
- The PC will usually run without keyboard or display. They can be reconnected for maintenance needs.

As a result of some of the above constraints and due to working conditions which are typical of embedded applications, the software must also meet a number of requirements:

- It must work with a small disk space.
- It must require limited CPU power.
- It must boot with no need of human intervention.
- It must restart smoothly after power off, both programmed and due to failures.
- It must cope with untrained personnel.

From the point of view of the software development, it was clear that the software which analyzes the print stream had to be written from scratch, while existing libraries could be used to write the code needed to implement the Modbus protocol.

Choosing an Operating System

Why not a real-time O.S.? As it should be clear by the description of the problem given, we are not challenged by hard, real-time requirements. The limited speed of the two I/O channels puts an upper limit to the data throughput and no problem should arise from unpredictable delays in servicing the two serial lines. This makes it possible to use a comfortable standard O.S. and rules out any special purpose real-time operating system as a cost-effective solution.

Why not DOS? The problem at hand is essentially asynchronous in nature. The system must be able both to swallow the data stream as produced by the DCS printer output and to reply to queries from the SCADA system, two process which are not synchronized in any way. DOS is thus clearly not a viable solution. Although it has the advantage of being very lightweight and having a small footprint, it would require quite a lot of low-level programming to solve the problem.

Why not Windows 95/98/NT? Microsoft Windows (especially Windows NT) is gaining consideration in industrial applications due to the great number of third-party packages and tools suited to any need, and because it is backed by a major vendor. However, the hardware characteristics of the target computer makes it challenging, if not impossible, to tailor a version of any Windows brand to the limited file space and CPU power available. Windows CE was not stable or reliable enough to be worth consideration. Moreover, the peculiar (weak) multitasking capabilities of Windows would likely make the programming of concurrent applications slightly more difficult.

Why Linux? Managers are usually worried when they have to make choices which are not “industrially sound”. They will have no problems in justifying the choice of a costly product provided that it is backed by a major vendor and has a large installed base. This is not (yet) the case of Linux, so to foster this kind of choice the technical arguments must be particularly strong. In this case, I believe that the advantages provided by Linux are superior to many objections.

The choice of Linux has many pros:

- Linux can be tailored to any particular set of needs down to the level of kernel configuration.
- Documentation on how to tailor the system is thorough and easily available.
- Linux can run from a RAM disk, so the Flash EPROM disk is used only to bootstrap. In this way the disk can even be write-protected.

- Development may be made in the same environment where the target system will run.

It also obviously has some cons:

- Finding drivers for non-standard devices may be difficult or even impossible.
- Support from vendors may be more difficult.

The Gory Details

In this project we needed two non-standard devices, i.e., the Flash EPROM disk and the RS-485 serial interface. This didn't cause any problem: the Flash EPROM disk was used as a standard DOS formatted device and the serial interface replaces the COM ports and is managed by the standard serial driver.

Software Architecture

The protocol converter gateway software has a very simple structure. It consists of two independent processes: a data swallower and a Modbus server. The data swallower receives printing lines from the serial port, analyzes the strings, extracts relevant data and writes the values into a shared memory buffer. The Modbus server receives queries from the SCADA system according to the Modbus protocol specification and sends back the requested values, reading them from the shared memory buffer.

Two interactive programs have been added for debugging purposes: a memory dumper which prints out values read from the shared memory buffer, and a program to write values into the shared memory. Both programs can, obviously, run concurrently with the gateway processes.

All the code is written in C and the shared memory management was implemented by using the standard System V interprocess communication API which allows the creation and management of shared memory segments and provides *semaphores* for synchronizing the access to them.

Due to the very simple structure of the problem, synchronization was easily implemented by locking access to the entire shared buffer when doing a memory access. This simple-minded approach is quite suitable in this case because all the accesses to memory are performed in chunks and the low speed of the I/O operations, with respect to memory accesses, ensures that any process will wait for a memory lock release for relatively short times.

As a tool for the constant verification of the proper functioning of the gateway, a location of the shared-memory buffer has been reserved for a counter which is incremented anytime the data swallower process ends a reading cycle. The SCADA system periodically reads the variable and may thus raise an alarm if the variable is not incremented after a given period of time.

System Configuration

Following directions found in the Linux BootDisk-HOWTO, a small Linux system was built, starting from a standard Red Hat 6.1 installation. This is actually a trial-and-error process in that one has to find out exactly which files are needed for one's purposes.

Even though our Flash EPROM disk provided a comfortable 8MB of disk space, all the software must be transferred onto the target computer by floppy disks, thus it is desirable that the resulting system is as small as possible.

The tailored system includes the kernel, a number of standard Linux commands (we were fairly prodigal in adding commands: better to have all the needed tools at hand when doing maintenance in the future) and all the related libraries. It also includes the loadable modules which are needed to manage DOS format volumes. This may be useful to mount and access either DOS format floppy disks or the Flash EPROM disk.

Needless to say, the four programs developed for the gateway, plus a few ASCII configuration files used at startup by the two running processes, were also included.

Due to the Linux bootstrap procedure requirements, the above components were stored into two files: the compressed kernel image (450KB) and the compressed root image (2500KB). Just a tiny bit more than what would fit onto two floppy disks: we actually needed three floppies for the full distribution. In the BootDisk-HOWTO one can find a number of hints related to shrinking the size of the root image, but we felt satisfied with the size reached and did not want to work more on this aspect.

Bootstrapping the System

The Flash EPROM disk selected for the project (M-Systems DiskOnChip) is provided with a Linux driver and can be used as a Linux bootstrap disk. This can be done by including the DiskOnChip driver into the kernel but also requires some fiddling with a DiskOnChip configuration utility and a special version of LILO to make it bootstrappable.

After a few tests we preferred a different solution: DiskOnChip was configured as a plain DOS bootstrappable disk. This has the advantage of avoiding both rebuilding the Linux kernel and reconfiguring DiskOnChip (it is the device's shipping configuration) and moreover it was considered a more stable solution with respect to future releases of the device. The Linux image files are stored on the DOS file system and Linux is booted by the LOADLIN utility. This adds around 160KB of DOS files to the software.

The system's power-on sequence is thus:

1. Boot DOS.
2. Run LOADLIN from AUTOEXEC.BAT to boot Linux. The boot sequence creates the RAM disk containing the Linux file system and expands into it the compressed root image.
3. Start the protocol converter processes.

The protocol converter processes are started at boot time because they are inserted into the **inittab** table. This also provides automatic restart in case of a crash of any of the two processes.

After booting, if the keyboard and display are connected, the usual Linux login prompt is displayed and a root login can be done. This allows us to perform maintenance operations and notably to launch Linux commands or use either of the two interactive monitoring programs described above. If needed (e.g., in order to make modifications to the configuration files without going through the entire process of making a new root image), the Flash EPROM disk can be mounted as a DOS volume.

Conclusions

We have shown how the use of Linux can solve a typical problem of data acquisition in an industrial environment. Actually, we were able to build a system which can do something useful with 32MB of RAM and as little as 8MB of disk space, but the most noteworthy characteristic of the system is its robustness.

Disk access is limited to read-only at boot time; the run-time file system is supported by RAM disk. This means that restarting after a power failure will never require a file system check, which could otherwise prevent the boot process from proceeding. Moreover, because all the system files stay on a read-only device, it is most unlikely that they are inadvertently tampered with by anybody. Finally, in case of a program crash, Linux provides the capability to restart the processes.

The native support of the compressed kernel and root images was also very valuable because it allowed us to keep the entire system very small.

email: lfini@arcetri.astro.it

Luca Fini (lfini@arcetri.astro.it) has been with the Osservatorio di Arcetri for 20 years where he has worked as software developer and system manager of a LAN with more than 100 computers running UNIX, Linux, Windows 95/98/NT, MacOS, and more. He also deals with control system design and development for both astronomical instrumentation and, occasionally as an independent consultant, in the industrial field.

Archive Index Issue Table of Contents

Advanced search

The Bullet Points: Linux 2.4 - Part Deux

Joe Pranevich

Issue #77, September 2000

Introducing the new DevFS: Linux 2.4 will change the way you access your devices.

Every so often, something happens that is so breathtaking, so absolutely amazing, that it changes the world. Linux 2.4 still isn't one of those things. Linus' rapid-release plan didn't actually rapidly produce a new kernel, not at the rate many developers (myself included) expected. This article is an addendum to “The Bullet Points” (January 2000) and a followup to “Linux 2.4 Spotlight: ISA Plug-and-Play” (February 2000). For a more complete rundown of Linux 2.4, see “Linux 2.4 Scorecard” and “Wonderful World of Linux 2.4”, available on Linux Today (www.linuxtoday.com).

Linux 2.2 took just over a month during its “pre-” stage, ditto with 2.0. While the 2.4 kernel is in the final development cycle, this pre-series has been extended several months and it's still difficult to predict how long it will take for the official release. The 2.2 kernel had some “brown bag” problems that I'm sure Linus would not be interested in repeating. According to the recent 2.4 bug list, there were 12 very important bugs that need fixing in the current pre-releases, 26 less important ones, several more minor minor ones, lots of things to merge and verify and other similar things. This list grows smaller daily, but it's still a big list.

Alan Cox's recent work on the “ac” series of kernels has helped to stabilize the code, but a number of items remain on the “must fix” list. I expect 2.4 to be released in a couple of months. The actual date of release, however will be entirely up to Linus.

And now, in no particular order, more of the new features of Linux 2.4.

DevFS: The Device Filesystem

Nearly every variant of UNIX accesses block and character devices through a common model: special device files in the /dev directory. Unfortunately, different UNIX variants name these devices in drastically different ways. BSD variations, for instance, refer to hard disks as /dev/wd* instead of /dev/hd*. (The characters “wd” stand for Western Digital, the manufacturer of IDE hard disks.) Unlike Linux, some UNIX variants use the /dev directory for network devices, too (e.g., eth0).

Although the naming of devices is different across UNIX variants, the method of how these devices communicate with the kernel is generally the same. Under Linux 2.2, special files (device nodes) for each accessible device are placed in the /dev directory. The device nodes are just two values, a major and minor number. The major number generally correspond to a driver or subsystem in the Linux kernel. The minor numbers generally corresponds to a sub-function or sub-device (e.g., a specific partition on a hard disk). Communication is actually done with these numbers. The names of the device nodes are standardized, but could be called anything an administrator wanted. (All Linux distributions today use the same standard naming scheme provided as a document with the kernel.)

There are several shortcomings of the 2.2 /dev model. First off, it uses a finite numeric name-space. Inevitably, we run out of numbers naming new devices in the standard. Second, modern devices (such as USB) don't lay well against a major/minor arrangement. Even complicated partitioning schemes reveal limitations in the current infrastructure. As it's impossible to know in advance which devices a user will install on their system, distributions create hundreds (sometimes more than a thousand) of device nodes in the /dev directory. Only a handful are used on a given machine; the rest remain to satisfy all possible configurations of all compatible hardware devices. It's a bit crazy, when you think about it. To better meet the demands of a more plug-and-play world, Linux 2.4 introduces the Device Filesystem (DevFS): in an elegant reworking of /dev, only configured devices are listed.

DevFS is a kernel-based file system. Like /proc, DevFS is seen in the filesystem tree (as /dev) but never gets “synced” to a physical device; /dev stays in RAM. Whenever a driver is loaded into the kernel and the device is detected, appropriate entries are added to the /dev tree.

Besides cleaning out the file system tree, DevFS will be faster, too. Here's how it works. Communication is direct; when you open() a file, you are communicating directly with the driver. Under 2.2, when a normal device node is open()ed, the kernel looks up the major device number in a table and calls a function. The actual driver then works out the specific device using the minor number. DevFS removes a layer of indirection. Now programs talk to drivers, not nodes.

With the introduction of DevFS and deprecation of device nodes, Linus has decided 2.2 device nodes are too limiting and going the way of the dodo. Nearly all DevFS node names are different than their major/minor counterparts, usually very different. Device classes are now categorized in subdirectories (structurally similar to /proc/scsi/*). For compatibility with 2.2 systems, administrators may still use old-style nodes (with the devfsd d<\#230>mon), but they are deprecated.

Unfortunately, there are some problems with DevFS as well. The permissions are more difficult to manage in the new system. Device permissions are allocated by the drivers, rather than by nodes in the file system. Changes to permissions can be made, but they are forgotten each time the module is loaded (or on reboot). Drivers (part of the kernel) name themselves, hence controlling the naming conventions and /dev layout, hence setting policy which is considered a bad thing. You can move (or rename) DevFS files after boot, or change their permissions, but you start with the kernel's policy. (Scripts are available to record the state of /dev at shutdown.) Presently, it's not clear which routines the distributions will use, or if they will even use DevFS. DevFS is still “experimental”, after all, though that may change before release.

While setting policy in the kernel is considered a bad thing, the many wins of the DevFS model have convinced Linus that the new way is the right way. Work is being done to make DevFS more “friendly” to administrators stung by its downsides—some of this is likely to make its way into the kernel before Linux 2.4 officially ships.

Logical Volume Manager (LVM)

Another semi-revolutionary change made to Linux 2.4 involves the way it handles disks and partitions. Previous versions of Linux were somewhat limited in the way disks and partitions were used. Like most other operating systems, Linux sat directly on the standard hardware partition scheme. Using fdisk or a similar tool, you would create partitions that would be formatted as “Linux swap” or “ext2”. These partitions are difficult to change once in place, however. Resizing or moving a partition was almost impossible, even for power users.

Previous versions of Linux include the “multiple device” (or “md”) driver, which allows Linux administrators to concatenate partitions and do more complicated maneuvers to create software RAID arrays. Using this driver and some footwork, it is possible to extend the native partition scheme somewhat, but not to the level of flexibility required by many applications.

The Logical Volume Manager has the power to make the Linux world a more flexible place. (Just as with the md driver, other OSes generally won't be able to understand Linux's LVM scheme.) Instead of dividing disks into static partitions, the LVM allows a Linux user to concatenate several physical disk devices into a single “volume group”. These groups can then be partitioned into multiple “logical volumes”. The LVM allows volumes to be resized (with certain constraints) and moved. More disks can be added to volume groups on the fly, allowing for massive storage capacity—the sky is almost literally the limit. The LVM subsystem is not new to the commercial UNIX world; the code is modeled partly from its implementation in Tru64 UNIX, HPUX and other commercial UNIX systems.

As an added bonus of this effort, code to resize ext2 partitions has been released to the public. While this code lives out in user space, far away from the kernel, it is a very important component of the LVM subsystem. Even when used without the LVM subsystem, it will no doubt be used in the next generation of installation programs and fdisk-like utilities.

Bigger and Better

Linux 2.4 has been through several major and minor stumbles on its way to becoming suitable for very large environments. The hardware is largely to blame for the difficulty. i386 hardware (being 32-bit and somewhat archaic) cannot easily support huge files and other requirements of a 64-bit operating system. This is a limitation of the hardware, and not a limitation with Linux itself. With the growing popularity of NT on i386 hardware, there has been a push to get 64-bit Linux on i386 too—a push to raise the bar where Linux can, to overcome the limitations of the platform without sacrificing the cleanliness and speed of the current implementation. Two major improvements in this area really stand out: more users per system and very large files.

First, one feature literally demanded by enterprise is 32-bit user and group IDs (UID and GID). In the Linux and UNIX worlds, every user is given a unique number. Unfortunately, the numeric system is finite and a limit of 65,000 users is constricting in some applications (i.e., a high-volume web hosting site like Geocities or Tripod or an ISP will have scaling issues when they gain 65,000 users). Linux 2.4 bumps the limit up to about 4.3 billion. For comparison, the population of the world is just over 6 billion. [Almost enough for everybody to have an account on every single computer! —Ed.]

Along with this trend of bigger and better, Linux 2.4 raises another ceiling: maximum file size. Previous incarnations of Linux would choke on files larger than 2GB, despite the fact that the underlying file system could theoretically handle it. Although many people may not immediately see the benefit of having incredibly large files, this feature is especially useful for managing information or media—imagine tarring all your MP3s into one convenient file, or putting them in a database.

It should be stressed that while Linux 2.4 will allow for bigger and better applications, one of the main concerns of the development team has always been to optimize for the common case: real users. To this end, the code has been carefully designed in such a way as to affect real users the absolute least possible. While these features will be assets to the users who need them, they will not be a stumbling block for users who don't.

New Ports: ia64, SuperH, S/390

One benefit of the open-source methodology of development is that it is perfectly reasonable for one to "fork" a copy of the kernel and do their own development on the fork. This method is often employed by kernel hackers who want to plod off and do their own thing for a while without having to pay careful attention to the other developments. As it turns out, however, these changes must eventually be integrated into the "official" kernel to be properly respected and supported. Although forks can be of any type, one common area for Linux to fork-and-merge is its support for processor families.

With the latest revision of the Linux kernel, several new processor families are given their special place within the kernel sources. ia64, one of the most talked-about additions to the kernel in some time, is the future Intel processor that is supposed to be the 64-bit replacement for the i386 line. In a sense, it's sort of like the PPC to Motorola's m68k. Hardware for this platform is virtually nonexistent and is not expected to reach consumers in quantity for years. However, Linux will be there when the first motherboards and processors roll off the assembly line. In many ways, this is a demonstration of Linux's larger role in the operating system market, as all previous Linux ports were done after hardware and an alternative OS were already available.

SuperH is the embedded processor used in Pocket PC (a.k.a. Windows CE) machines and another addition to the "supported by Linux" clan. There's an irony to supporting Linux on WinCE hardware that many users just can't help but chuckle over. Again, this port is still in an early stage, but the developers are chugging along.

About as far on the other end of the computing spectrum as you can get, the S/390 will get a port, too. The S/390 is the latest generation of IBM's mainframe line and probably the largest variety of hardware that Linux is known to run on. Much of the port was done by IBM; this in itself is another first for the Linux community.

While it is true that Linux supports all of these new processors in the official kernel source distribution, these ports are not necessarily ready for prime time. Additionally, pre-compiled distributions for any of these processors may be a long way away from the “normal user” stage.

Firewall/NAT Rewrite (iptables)

Although I probably shouldn't admit it, I know a number of system administrators who never bothered upgrading their kernel to Linux 2.2. Even though Linux 2.2 included a completely rewritten network layer, the cost of bothering to update their old (2.0 era) scripts to the 2.2 command set was daunting to some. That being said, Linux 2.4 has rewritten the entire networking layer (again!) and introduced an entirely new interface: iptables. But what about those people who don't want to upgrade again? This time around, Linux 2.4 includes compatibility modules for both the 2.0 and the 2.2-era tools. With compatibility tools lowering the cost of entry, it is hoped that this release of the Linux kernel will be more readily implemented than the previous release.

The Linux 2.4 networking layer wasn't rewritten again for nothing. Network Address Translation (NAT) and Firewall operations have been made more flexible in their operation and split off into separate modules. With these modules, a Linux 2.4 system becomes nearly as powerful and flexible as modern-day commercial routing hardware. Of course, to use the really nifty features of the new kernel, you have to be using the “real” iptables interface and not either of the compatibility interfaces provided.

While the new flexibility may be enough to convince hard-core network people to upgrade, the 2.4 Linux kernel also includes more general fixes and speedups for the networking layer. David Miller and the rest of the networking gurus have been hard at work making sure Linux 2.4 talks more efficiently to other operating systems. The networking layer and the TCP/IP stack have been rewritten to be more scalable on multi-processor machines. Network device drivers are now written to make them more stable and to eliminate some possible race conditions in the infrastructure, too. These changes further build on the great work that was done with Linux networking during the Linux 2.2 development cycle.

Direct Rendering Manager

Linux 2.2 included the first official support for frame-buffer graphics devices in the kernel; Linux 2.4 also recognizes a new interface for kernel-level control of graphic hardware. With the introduction of Direct Rendering Manager (DRM) comes a system to keep multiple demanding video processes in check. Rather than being a complete video driver in itself (such things are better left in user space), Linux 2.4 makes user-space video more stable (and secure) by providing a kernel interface which controls and synchronizes access to graphical devices. Supported programs, such as Xfree4.x, will talk to this interface whenever a hardware resource is needed. The kernel will know when multiple programs are attempting simultaneous access to video structures, and will save state or do whatever is necessary to make sure they don't conflict. Since supported programs will be unable to send conflicting requests to the graphical hardware, these conflicts will not be able to cause a crash. This new feature is largely geared to advanced accelerated hardware, but lower-end hardware may benefit from the new resource allocation routines as well.

Firewire/I2O

One major area of improvement in Linux 2.4 is the number of device types it supports. I already wrote about Linux 2.4's support for USB, ISA Plug-and-Play and PC Card devices in my previous articles. This picture would not be complete, however, without mention of support for Firewire and I2O (Intelligent Input/Output) devices, two relatively new additions to the PC hardware market.

Firewire, IEEE 1394, is a high-speed external bus system that is similar in concept to USB. (You may also hear it called by Sony's name: i.Link.) Unlike USB, Firewire supports multiple computers on the same bus and at higher-speed transfers than USB. Due to the high bandwidth available, Firewire has proven most useful for digital (video) cameras and similar devices which require a lot of data to be transferred quickly. It should be noted that, although the underlying bus is supported under Linux, not all hardware chip sets and devices are supported yet. This support will improve over time and as more hardware becomes available.

I2O is a new type of I/O subsystem that features operating-system independence in addition to high-speed data transfers. This means that, in theory, one driver is guaranteed to work with all devices of a specific type, regardless of vendor or how the device actually works internally. Unfortunately for us, there are relatively few I2O devices made so far, and the kernel support is still somewhat incomplete.

Although Firewire and I2O are relatively new to the Linux sphere and relatively little hardware actually exists for these bus types, the open-source snowball is rolling and support for these device types will improve as these devices become more common.

Linux 2.4 Scorecard

In Conclusion

Linux 2.4 is shaping up to be the best version of Linux yet... oh, wait, I already used the line for Linux 2.2. With many new features for desktop and enterprise, the new kernel has something for everyone. Bring the kids!

> Joseph Pranevich (jpranevich@lycos.com) is an avid Linux geek and, while not working for Lycos, enjoys writing (all kinds) and working with a number of open-source projects.

email: jpranevich@lycos.com

Archive Index Issue Table of Contents

Advanced search

Content Management

Reuven M. Lerner

Issue #77, September 2000

Keep track of updates to your web site documents with this Mason application.

Over the last few months, we have looked at Mason, the web development system written by Jonathan Swartz that combines Apache, **mod_perl** and HTML/Perl templates. Normally, Mason is associated with web applications, especially those that use back-end databases and produce dynamic content. Indeed, I have found Mason to be an extremely useful tool in my consulting practice, making it possible to create web sites quickly and easily.

But Mason can be used for more than simple server-side web applications. One of the most interesting Mason applications is a set of components known as the “Mason content manager” or “Mason-CM”. Mason-CM provides basic content management functionality, beginning with its ability to handle the staging of files to a production server, and continuing with built-in support for spell checking, RCS version control and editing of Mason components.

While Mason-CM is written in Mason and thus requires Apache, mod_perl and the appropriate Perl modules, it can work just fine with a static site using nothing but HTML and graphic images. In fact, I recommend that even site owners uninterested in Mason and mod_perl take a look at the Mason content manager, because it provides so many useful features in a simple, free package.

Why Content Management?

As web sites have become increasingly complex, so too have the organizations surrounding them. Whereas it used to be common for a professional site to be a one-person operation, it is now usual for even a small or medium-sized site to have at least three people: a writer/editor, a designer and a programmer. Even such a small team will eventually find multiple members trying to edit the same file simultaneously. This problem was solved years ago by version control systems, such as RCS and CVS—but the tools were designed by and for programmers, and can often be daunting for a designer or editor to use.

Moreover, the Web presents a number of challenges that are different from development in the traditional software world. For example, software is traditionally written, compiled, tested and debugged, at which point the cycle begins again until the software is released. The Web, however, works differently. As soon as an HTML file in the web document hierarchy is edited, it is immediately available to everyone on the Internet. This is good news when someone finds a mistake needing to be fixed quickly, and also means that sites can update their content on a regular basis without having to go through long processes. However, it also means that the results of editing a file—whether they are improvements or mistakes—are immediately available to anyone who happens to enter the right URL at the wrong time.

For all of these reasons, most medium- and large-scale web sites now run two web servers. The first, known as a “staging server”, is where the developers, editors and designers can write, edit and test their changes to the site. Only after files have been fully tested are they sent to the second web server, known as the “production server”.

Of course, these two servers need not be on separate computers. They merely need to be separate in some way, to keep outsiders from seeing the content as it is tested and to ensure that the staging and production servers have parallel directory structures.

Mason-CM is a set of Mason components that makes it relatively easy for anyone to set up a content management system on their own server. The user interface is not beautiful, and I had some minor problems getting it installed on my system. However, it does the job more than adequately, and makes it possible for multiple people to work on a project without stepping on each other's toes.

Installing Mason-CM

Before you can install Mason-CM, you will need to have a working copy of Apache with **mod_perl** and Mason installed. (If you need help installing Mason, see the last three months of “At the Forge”.) In addition, you will need to download and install several Perl modules from CPAN, including MLDBM, Image::Size, URI::Escape and File::PathConvert. Make sure to import these in your Apache configuration file (httpd.conf) with the **PerlModule** directive, so as to increase the amount of memory shared among the various Apache child processes.

Mason-CM comes as a gzipped tar file from the Mason home page (see Resources). The archive should be unpacked into a directory under your Mason component root. I chose to put it in /usr/local/apache/mason/cm, where /usr/local/apache/mason is my Mason component root directory. If you have unpacked the archive correctly, the /cm directory should contain a **README** and an **INSTALL** file. I will cover all of the steps needed for installation, but it is probably a good idea to read through those files just in case.

Because Mason-CM must ensure that each file is being edited by only one user at a time, and because a content-management system should be available to authorized users only, you must restrict access to the /cm directory using HTTP authentication. Mason-CM will refuse to run unless it is in a directory that has been password protected.

The easiest way to password-protect a directory is to create a .htaccess file in it. The .htaccess file overrides the default Apache configuration settings for its directory, as well as any subdirectories underneath it. For example, here is the .htaccess file for my content management system:

```
AuthName "Content Management System"
AuthType Basic
AuthUserFile /usr/local/apache/conf/staging-passwords
require valid-user
```

The **AuthName** directive defines the string that will be displayed to users when prompted for a password. (Without such a string, it might be hard for users to remember which system is requesting a user name and password.) The **AuthUserFile** directive should point to a file containing user names and encrypted passwords.

The password file should be outside of the web document root directory, so that users cannot retrieve it using their web browsers. To create or edit the password file, you will need to use the “htpasswd” program, which is installed by default in /usr/local/apache/bin.

The “require valid-user” tells Apache that without a user name and password that match the entries in AuthUserFile, the user will be denied access to the directory.

If the .htaccess file does not have any effect, check the AllowOverride directive in httpd.conf. This directive indicates which Apache configuration options can be overridden by an .htaccess file. By default, Apache servers are configured not to let .htaccess files modify directives of type AuthConfig. You can change this by putting the following section in httpd.conf:

```
<Directory /usr/local/apache/mason/cm>
AllowOverride AuthConfig
</Directory>
```

Finally, I found that there was a small bug in my installation of Mason-CM. Many of the components lack a .html suffix, making it difficult or impossible for Apache to identify the content as “text/html”. So even though Mason-CM components produced output in HTML-formatted text, the contents were interpreted by my browser as if they were in unformatted ASCII. To give Apache some help, I explicitly set the content type using the mod_perl “content_type” method. The following autohandler, placed inside of the /cm directory, automatically sets the content type to “text/html” for each document within the directory:

```
<% $m->call_next %>
<%init<
$r->content_type("text/html");
</%init>
```

Because my **mason.pl** configuration file is set to ignore non-text files, I can be sure that the above will not accidentally force JPEG and PNG images to be rendered with a type of “text/html”.

Configuration

Mason-CM is now in place. However, we need Apache to load a number of Perl modules in order for it to work. In your Mason configuration file (which I call “mason.pl”, but the Mason documentation calls it “handler.pl”), insert the following block of Perl code:

```
{
   package HTML::Mason::Commands;
   use Fcntl;
   use MLDBM;<\n>
   use Image::Size;
   use URI::Escape;
   use File::PathConvert;
   use File::Copy;
   use File::Find;
   use IO::Handle;
   use IPC::Open2;
}
```

Now that we have told Apache and Mason where to find Mason-CM, it is time to configure Mason-CM itself. Nearly all of the configuration is performed by modifying the cmConfig component, located in the /cm directory. As of this writing, cmConfig is written using the old-style Mason interface, which may seem a bit foreign to those of us who started using Mason with version 0.80. For example, the initialization block is called **<%perl_init>** rather than simply **<%init>**, and one component invokes another with **mc_comp** rather than **$m->comp**. Nevertheless, the component should be relatively easy to recognize and understand by anyone with even a minimum amount of experience with Mason.

The two main variables that must be set at the top of cmConfig are **$CM_HOME** and **$CM_DATA**. (These are defined on line 25 of the default version of cmConfig, at the top of the **<%perl_init>** section.) The first refers to the directory in which Mason-CM is installed. The second refers to the directory in which Mason-CM can store information on the files it manages, such as locking and version control information. On my system, I defined them as follows:

```
   my $CM_HOME = '/usr/local/apache/mason/cm';
   my $CM_DATA = '/usr/local/apache/cmdata';
```

Both of these directories must exist in order for Mason-CM to work. While **$CM_HOME** should already be defined (since cmContent is supposed to be inside of **$CM_HOME**), you may need to create the **$CM_DATA** directory. Note that this directory is different from the Mason data directory, which I typically put in /usr/local/apache/masondata.

Following the definitions of $CM_HOME and $CM_DATA is a large hash, called **%cm_config**. The keys in %cm_config describe different configuration options, and the values are the settings for those options. In most cases, the default options are probably adequate; we will discuss only those options that you must or should change.

The “admin” key refers to the e-mail address of the Mason-CM administrator. The administrator is responsible for undeleting files, unlocking locked files and generally managing the content management system. By default, this is set to be cm-admin, but you can change the value to something else.

Defining Branches

The value associated with the “branches” key is an array reference describing the various “branches” on the version control system. While all of the branches must be under **$CM_HOME**, this makes it possible to differentiate between subsites. For example, a newspaper might have separate branches for the news, sports and business sections. Each branch is identified by a unique name, followed by a hash reference identifying different characteristics associated with the branch. For example, here is what the “branches” key looks like for a web site with a single branch, called “Primary”:

```
   branches => [
         Primary => {
             path =>'/usr/local/apache/htdocs/staging/content',
             trg_from => 'staging',
             trg_to => 'production',
             components => 0
         }
      ]
```

The above branch will be displayed in the Mason-CM “branch selector” as “Primary”, and controls all of the documents under /usr/local/apache/htdocs/

staging/content. Make sure the named directory does not end with a “/”, or Mason-CM will fail with a security violation.

The “trg_from” and “trg_to” keys are used in a simple substitution, indicating that to move documents from the staging server to the production server, we replace the string “staging” with the string “production”. (Mason-CM calls the staging process “triggering”.) Thus content is initially placed in /usr/local/apache/htdocs/staging/content, and is staged to the directory /usr/local/apache/htdocs/production/content. Finally, we indicate that this branch contains static HTML (rather than Mason components) by setting the “components” key to 0.

A more complicated site might set branches to the following value:

```
branches => [
        News => {
            path => '/usr/local/apache/htdocs/staging/news,
            trg_from => 'staging',
            trg_to => 'production',
            components => 1,
            hidden => 1
        },
        Business => {
            path => '/usr/local/apache/htdocs/staging/business,
            trg_from => 'staging',
            trg_to => 'production',
            components => 1,
            obj_dir => '/usr/local/apache/staging/obj',
            hidden => 1
}
]
```

The above Mason-CM configuration has two branches, known as “News” and “Business”. Because “branches” is an array reference rather than a hash reference, its elements are kept in their original order. This means the branch selector will display the branches in the order they are entered into branches above. Changing the order in which branches are displayed is as easy as modifying the order of elements in the branches array reference.

If we set up the branches using the above system, we can then modify our Apache configuration such that it takes any URL beginning with /news and rewrites it as /production/news:

```
Alias /news /usr/local/apache/htdocs/production/news
```

Now the staging server is hidden from view via a web browser. We can, however, configure our web server such that all requests to port 8080, or any other port we choose, are directed toward the staging server.

The “hidden” tag indicates whether the branch will be displayed by default in the branch selector. Normally, all branches are displayed by default, and are available to all users. And any user can customize the list of branches using the “my.CM” link in the upper right-hand corner of the Mason-CM index page, adding and removing branches from his or her menu. However, making a branch hidden by default gives new users a relatively clean view of the content management system.

Unlike the “Primary” branch, “News” and “Business” are defined to contain Mason components. Staging a component is different from staging a static page of HTML, in that Mason-CM will try to compile the component and test it for errors before actually allowing it on the production server. In this way, a broken component will not cause the production web site to fail, but rather only the staging server. If we want to store the compiled Mason components in a specific directory, we can specify that with the “obj_dir” key.

cmConfig can be modified in many other ways to customize your Mason-CM. However, once you have configured **$CM_HOME**, **$CM_DATA** and branches, you can begin to use Mason-CM.

The Index Page

To access the main Mason-CM interface, point your web browser at **$CM_HOME**. On my system, I opened the URL http://localhost/mason/cm/.

Because this directory is password-protected, I had to enter a user name and password. Following a successful login, I was presented with the main Mason-CM index screen. To a large degree, the index page is a web-based file browser, allowing you to navigate through the directories and subdirectories in the various defined branches, open files for reading and writing, and search for a file by file name or content.

The index screen is easily identified by the picture of a juggler. While you can replace this with any image you prefer (setting the “juggler_src” key in cmConfig), the image seems rather appropriate for those of us who work on web sites! Clicking on this image from anywhere in Mason-CM brings you back to this main index page.

Along the right-hand side of the index page is the branch selector, listing the branches that were defined in cmConfig. Clicking on a link within the branch selector allows you to handle staging for that particular branch. The current branch appears in a slightly different background color from the other branches, so your current location should always be fairly obvious.

The current directory is identified in the middle of the screen, with the “current directory” headline (and a blue default background). Each component of the current directory path is a hyperlink to that path, making it possible to navigate using the mouse. To switch into a subdirectory, merely click on its name.

Alternately, you can create a new subdirectory by using the text field in the middle of the screen.

Above the “current directory” line is a search system. I am obviously not the only person who has ever reverted to **grep** and **find** after failing to remember where a particular file is stored on a web site. Mason-CM puts both of these programs into an easy-to-understand package, allowing even non-UNIX users to search for files within the current branch. The search supports Perl regular expressions, meaning you can look for files by name or by content in a variety of ways. Be careful about what you search for, however; Mason-CM will happily search through hundreds of files for a complex regular expression, even if the execution will take a long time.

Beneath the “current directory” line is a list of files available within the current directory. Each file is identified by name, by its last modification date, by the person who performed the last modification and by the file's current status. The status is one of “staging” (meaning it exists on the staging server only), “prod” (the file is identical on the staging and production servers) and “modified” (the file exists on both servers, but has been changed on the saving server).

You can also create a new file, using a text field and the “create” button, just before the list of existing files. Do not confuse the subdirectory create button with the file create button; I modified the button definitions in the “dirTable” and “fileTable” components, so that they say “create subdirectory” and “create file”, respectively.

Viewing and Editing Files

To view the current version of a file, click on its name in the file table. The HTML source will be displayed at the top of the browser window. At the bottom of the page, you can ask to see a rendering of the HTML, to wrap text after 80 columns (rather than displaying the text verbatim), or to display line numbers along with the HTML source.

You can also edit files from within Mason-CM, using a primitive but functional text editor. To edit a file, click on the “edit” link next to the file name. This will bring up a <textarea> widget containing the file's contents. You can modify the contents of the file by typing into the <textarea> field, and can even copy or rename the file using the text field at the top of the page.

This editor is nearly as primitive as things can get, with a barely functioning set of Emacs key bindings for cursor movement. However, the fact that it lets you make modifications easily and quickly is certainly an advantage. And it is integrated into the rest of Mason-CM, in a format that most designers and editors can understand comfortably.

From the editing screen, you can choose from a number of options:

- The “save” button updates the file, and returns to the editing screen.
- The “save and exit” button saves the file to disk, and returns to the main Mason-CM index page.
- The “save and render” button displays the HTML output produced by the file, and can be used to preview the way a particular page or Mason component will work.
- Finally, the redraw button at the bottom of the screen makes it possible to resize the <textarea> widget, adjusting its height and width.

Mason-CM uses a locking mechanism to ensure that only one user can edit a file at a given time. If you are editing a file, it is noted inside a red box at the top of the index page. That box lists the files on which you're currently working, offering links to the file editor and to the “unlock” page.

If you try to edit a file that someone else is editing, Mason-CM will refuse to display the editing screen. Once the file is unlocked another user will be able to modify it again.

Staging Files

Once a file appears to work correctly on the staging server, you must move it to the production server. To do this, select one or more files within a directory with the checkboxes on the left-hand side of the file table. Then, click on the “trigger” button at the bottom of the page. The files will be copied over to the production server, instantly making them the “current” copies of the web site.

You can trigger all the files in a directory by clicking on the “check all” checkbox at the bottom of the page, next to the trigger button. This is particularly useful when you have created a new directory and want to use all of the items at once.

If someone happens to modify the production version of a file, it is possible to “reverse-trigger” the file. This copies the file from the production server onto the staging server. This is a potentially dangerous operation, and should not be treated lightly; as a result, Mason-CM asks for explicit confirmation before allowing such an operation.

Spell Checking

Once you have the basic Mason-CM functionality working, you may want to try some of the optional features that it includes. Perhaps the most interesting feature is the spell checker, which is a Mason component that uses ispell to check the spelling of the document. The Mason spell checker ignores HTML tags, so you need not worry about having to add “href” to the dictionary.

To enable spell checking, uncomment the “ispell”, “main_dict” and “supp_dict” keys in the %cm_config, defined in cmConfig. They are commented out by default; on my Linux system, I was able to uncomment them without making any modifications:

```
   ispell       => '/usr/bin/ispell',
   main_dict => '/usr/lib/ispell/english.hash',
   supp_dict => "$CM_DATA/suppDict",
```

Once you have defined these keys, the Mason-CM editor will automatically include a “spell check” checkbox. This means that every time you click on “save”, “save and render” or “save and exit,” the document will be spell-checked. If a word in the document is misspelled, a small JavaScript program allows you to choose an alternate, ignore the misspelling, or add the word to a dictionary. Everyone on a Mason-CM system shares a dictionary, meaning that if one user adds a word to the dictionary, everyone else will gain from it. (This also means that if one user accidentally adds a misspelled word to the dictionary, everyone will suffer, so be careful!)

Version Control

Mason-CM also supports the use of RCS for version control. This requires mason.pl to import the RCS module along with Image::Size, URI::Escape, and File::PathConvert. Following that, define (or uncomment) the following lines from cmConfig:

```
   rcs_bin => "/usr/bin",
   rcs_files => "$CM_DATA/archive",
```

Once Mason-CM sees that these values are defined, it adds a “version label” text field to the top of the editing page. If you enter a version label when the document is saved to disk, then RCS will automatically be used to keep both the older version of the file and the newer one.

Moreover, activating version control means that the file list will include a “versions” label. Clicking on this brings up a list of the document version history, and provides a nice interface to **diff** and the ability to check out older versions. Version control is almost a necessity when working on larger web sites, since bugs can creep in almost anytime, and it's often more important to use a stable, older version than an unstable, newer version with more features.

In addition to spell checking and RCS, Mason-CM includes a number of other features: users can upload files via HTTP and FTP, and administrators can restrict user access on a per-directory basis. Because Mason is written in a straightforward dialect of Perl, it shouldn't be difficult to add other features, such as the ability to stage to other computers (rather than other directories) and HTML validation before staging.

Conclusion

Mason may be a powerful tool for creating web sites, but Mason-CM displays the versatility of this tool. Mason-CM demonstrates that Mason components may be used to create a tool that doesn't directly affect the content produced on the Web. I am very impressed with the variety of tools Mason-CM offers, and while I won't be giving up GNU Emacs as my editor of choice in the near future, I do expect to use Mason-CM on a number of my clients' sites—both those that use Mason for content generation, and those that use simpler, less-advanced tools.

Resources

Reuven M. Lerner (reuven@lerner.co.il) owns a consulting firm specializing in web and Internet technologies, based in Modi'in, Israel. As you read this, he should (finally) be done writing Core Perl, to be published by Prentice-Hall. You can reach him via e-mail at reuven@lerner.co.il, or at the ATF home page, http://www.lerner.co.il/atf/.

The Ghost of Fun Times Past

Marcel Gagné

Issue #77, September 2000

Marcel writes about text-based games.

Bonjour, mes amis! Welcome back to Chez Marcel, where we dabble in fine Linux cuisine with a French touch and, of course, provide our wonderful clients with healthy servings of the proud and noble fruit of the grape vine.

Allow me to show you to your tables. Yes, I know that François normally seats you, but he is currently indisposed. What do I mean by that? Let me pour you a little wine and I shall explain. I have a wonderful Châteauneuf-du-Pape that I guarantee you will absolutely love. It is wonderful, is it not?

François? Ah, well, at the moment he is in the wine cellar. The air is cool and dry...something else, too: the faint odor of magic. To his right, a corridor leads into the dark. Turning to his left, François sees a wooden doorway with an old, large padlock. Directly in front of him are several rows of wine racks with several hundred fine vintages. A few steps behind him are the stairs that led him down into the cellar. What do you think he should do, mes amis? Should he walk into the dark corridor, or see what is behind the locked door?

The classic UNIX/Linux adventure game is simply called “adventure”, a search for fame, power and wealth as you enter “Colossal cave”. Something must be deep beneath the stream that runs next to the old farmhouse. You can tell, because all the water from the stream pours into the ground through a two-inch slit in the rockbed. Here's a quick reminder for those who might have forgotten.

> You are standing at the end of a road before a small brick building. Around you is a forest. A small stream flows out of the building and down a gully.go stream
>
> You are in a valley in the forest beside a stream tumbling along a rocky bed.

Go on. Admit your guilty pleasure. There are times, mes amis, when I long for the days of simple command-line entertainment: back to the days when an entire word-processing package would fit on a 128K diskette. I won't bother you with the details of my 12-mile walk to school through flurry of snow (uphill in both directions), but I will take you on a tour down memory lane, courtesy of your Linux system.

When I upgraded my system to Red Hat 6.1 (and even 6.2), I was devastated that my old friend, the “bsd-games” package, was no longer included on the CD (I understand some of the games had licensing questions, but certainly not all). KDE and GNOME came with a handful of great games, and the classic X games such as “xbill” were there as well. Unfortunately, the command-line classics were conspicuously absent. If any of the fine folks at Red Hat (or Caldera, or Corel) are reading, I might humbly suggest that they consider putting the text games back when they next produce a release. There are some wonderful pastimes which take up very little in space or system resources. People resources are another matter. I should point out, however, that a text adventure game looks a lot more like work than Doom, but don't quote me on that.

One of the classics that is still around, and included with most Linux distributions, is the venerable **fortune** program (as in fortune cookie). Displaying a fortune is as simple as typing in the path name to the command itself, usually **/usr/games/fortune**. Speaking for myself, I have spent a great deal of time with this little diversion. A common practice is to have the program execute each time you log in, thus giving you a different fortune each time. This is done by adding the command to your **.bash_profile** file. You can even specify the type of fortune. Have a look at the fortunes directory by doing an **ls /usr/share/games/fortunes**. You'll notice names like startrek, kids, zippy and linuxcookie (which sounds interesting). Let's try to call up a linuxcookie-specific fortune.

```
$ /usr/games/fortune linuxcookie
I've run DOOM more in the last few days
than I have the last few months. I
just love debugging ;-)
(Linus Torvalds)
```

Interesting. You can also specify long or short fortunes by adding the flags **-l** or **-s**. For instance, to generate Star Trek one-liners, try **/usr/games/fortune -s startrek**. Speaking of Star Trek, the famous text-mode Star Trek game is part of this package. Remember, though, you cannot fire phasers with your shields up. To start, just type: **/usr/games/trek**.

Before I go further, I should clarify something. As I mentioned, a number of the classics were now missing from my system. Merci Dieu for the Internet, non?

The bsd-games collection is still available, even if it is not on my Red Hat disc. When you get hungry, as I did, for the days of old (or new) when games were lighter and healthier but no less filling, visit the Metalab Linux archive at UNC Chapel-Hill (you'll find the link in the Resources section at the end). This excellent group of individuals “never sleeps” (or so they claim), making sure Linux users never run out of things to do. In fact, if you haven't found what you are looking for elsewhere, check it out.

Here's something for the Linux distributors out there who may wish to take up my earlier suggestion. The team at Metalab has taken the time to separate out the games that were at issue from a licensing standpoint. From their site, I picked up the latest bsd-games bundle, **bsd-games-2.11.tar.gz**. Building the games is easy. Extract the files into a work directory, then run the configure script:

```
tar -xzvf bsd-games-2.11.tar.gz
cd bsd-games-2.11
 ./configure
```

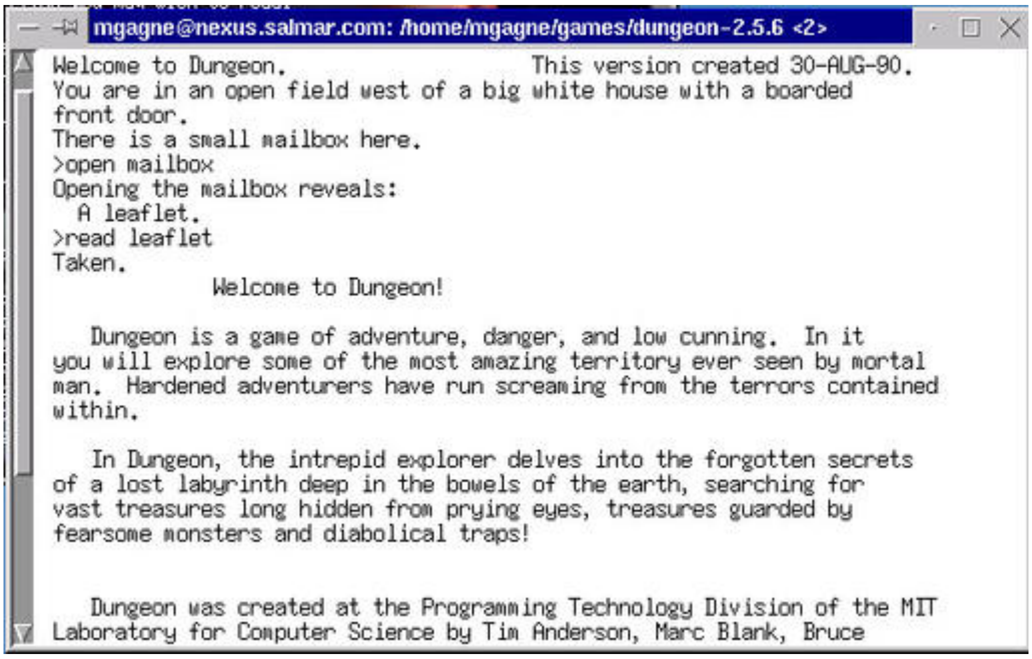
It is now question-and-answer time. The *configure* script will ask you a number of questions about where you want to store all these things. It does give you the opportunity to exclude certain games from the build. For instance, if you already have the fortune program installed, you don't want to overwrite it. Next, type **make**, followed by **make install**. I must admit I found some of the builds a little buggy, but I could build individual games by specifying the game name suffixed with an **_all**. To finish and install the games, you do the same thing, but with an **_install** suffix, like this:

```
make monop_all make monop_install
```

Any guesses on what that game might be? Consider it an adventure in program building. Speaking of which...I started this discussion by alluding to Adventure, the D&D style of quest game where you find yourself in a strange place with no instructions. This package contains **adventure**, the Colossal Cave challenge, as well as **battlestar**, another adventure, this time aboard a “battlestar”. If you really like creepy caves and feel the thrill of the hunt calling, you might consider hunting the Wumpus with **wump**. If you feel particularly destructive, you might even learn a lesson from **wargames**.

One game I highly recommend can be found in Metalab's **textrpg** directory. It is called simply **dungeon**. This game was originally written at MIT and is the predecessor of the now-legendary **Zork** games from **Infocom** (a company whose brilliant games stole a great deal of Chef Marcel's early years). The current version of dungeon is dungeon-2.5.6.tar.gz and comes in a pre-compiled ELF format. Simply extract the file into a directory and run it.

```
tar -xzvf dungeon-2.5.6.tar.gz
cd dungeon-2.5.6
 ./dungeon
```


mgagne@nexus.salmar.com: /home/mgagne/games/dungeon-2.5.6 <2>

```
Welcome to Dungeon.                    This version created 30-AUG-90.
You are in an open field west of a big white house with a boarded
front door.
There is a small mailbox here.
>open mailbox
Opening the mailbox reveals:
  A leaflet.
>read leaflet
Taken.
            Welcome to Dungeon!

   Dungeon is a game of adventure, danger, and low cunning.  In it
you will explore some of the most amazing territory ever seen by mortal
man.  Hardened adventurers have run screaming from the terrors contained
within.

   In Dungeon, the intrepid explorer delves into the forgotten secrets
of a lost labyrinth deep in the bowels of the earth, searching for
vast treasures long hidden from prying eyes, treasures guarded by
fearsome monsters and diabolical traps!


   Dungeon was created at the Programming Technology Division of the MIT
Laboratory for Computer Science by Tim Anderson, Marc Blank, Bruce
```

Figure 1. Are you ready to enter the dungeon?

These games challenge your mind as well. How about some speed **arithmetic** or a **quiz**? I may be French, but every once in a while, I feel the need to brush up on my Shakespeare. **quiz** lets me do that, as well as reminding me of Morse code, flowers and their meaning, and of course, **Middle-Earth** capitals. Type in the command **/usr/games/quiz** and you will get a menu of choices.

```
Subjects:
African capital
American capital
area-code state-region city
arithmetic answer
Asian capital
baby adult
Shakespeare-lines next work character
Chinese-year next
individuals collective
function ed-command
symbol number weight element
European capital
flowers meaning
```

This is just a sample of the selection. To start a quiz on Shakespeare-lines, you could choose to identify the character based on spoken line, or identify the work itself. Type **/usr/games/quiz Shakespeare-lines character** and you are on your way. Here's a sample for you to savor:

```
$ /usr/games/quiz Shakespeare-lines character
The quality of mercy is not strain'd?
Portia
Right!
My only love sprung from my only hate!?
Juliet
Right!
The better part of valour is discretion;?
```

Before the restaurant opened today, François and I were chatting on this subject and decided that there is something, how do you say, inherently geeky about deriving enjoyment from text-only games and diversions. This brings us to Chris Gushue's **Geekcode generator**. While this isn't a game per se, it is a way to proclaim your geekness to others who understand where you are and what space you occupy in this busy, multimedia world. Here is a sample geekcode block. Whose it is must remain a mystery.

```
-----BEGIN GEEK CODE BLOCK-----
Version: 3.12
GL d+ s+: a+ C++ UL++++ P+++ L++++ E--- W+++ N+ o-- K- w---
O M-- V PS++ PE Y++ PGP++ t++ 5+ X+ R tv-- b+++ DI++ D
G++ e* h--- r+++ y+++
------END GEEK CODE BLOCK------
```

Ah, with code like that, it can only be a sign that the day is getting on and that Chez Marcel must soon close. In this world of fast-paced multimedia games, it can be extremely satisfying to bury yourself in a text-based adventure, and letting your mind work in ways we don't do as often. After a few minutes (hours) of playing in this old-fashioned way, you may even find that the images you supply on the canvas of your imagination are far superior to those of today's noisier offerings. I submit to you, mes amis, that as a fine wine can improve with age, so can the experience of text over graphics.

Speaking of wine, let me refill your glass one more time. I fear I will be closing up the restaurant on my own tonight, since François has not returned. I do hope my adventurous waiter has not fallen into one of the open pits in the cellar. Do not worry about him, mes amis. I assure you, he is quite resourceful. Until next time, I invite you to explore and return safely to Chez Marcel. Your table will be waiting.

A votre santé! Bon appétit!

Resources

email: mggagne@salmar.com

Marcel Gagné (mggagne@salmar.com) lives in Mississauga, Ontario. In real life, he is president of Salmar Consulting Inc., a systems integration and network

consulting firm. He is also a pilot, writes science fiction and fantasy and edits TransVersions, a science fiction, fantasy and horror magazine. He loves Linux and all flavors of UNIX and will even admit it in public. You can discover many things from his web site at http://www.salmar.com/.

Archive Index Issue Table of Contents

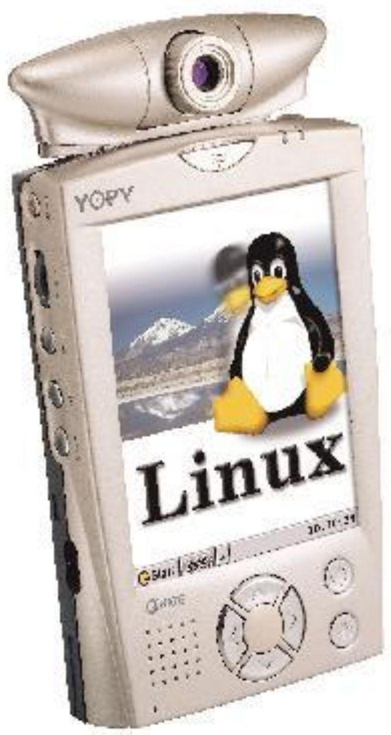
Advanced search

Yopy Puts Linux in Hand

Linley Gwennap

Issue #77, September 2000

While Linux shows some promise for handhelds, displacing current leaders Palm OS and Windows CE will be challenging.



Samsung subsidiary Gmate has announced Yopy, the first handheld computer that runs Linux. Gmate hopes that its support for Linux will lead to a flood of third-party software for Yopy, rivaling the software infrastructure that has emerged for Palm's handheld devices. While Linux shows some promise for handhelds, displacing current leaders Palm OS and Windows CE will be challenging. The prize is a market that will ship more than 10 million units this year and is growing by more than 30% per year.

A Powerful Handheld Computer

Yopy is a true handheld device, measuring just over 5 inches by 3 inches and weighing just under eight ounces. It bears a striking resemblance to the latest crop of Windows CE devices known as Pocket PCs, particularly to Compaq's iPaq H3600 device.

Yopy is nearly the same size as the H3600, although it is 25% thicker and 25% heavier. Both devices include a color TFT display and have a similar array of buttons. Inside, both rely on a 206MHz StrongArm CPU with 32MB of battery-backed DRAM for storing user data. These similarities are probably not coincidental—both devices appear to be based on the same Intel reference design.

32MB of flash ROM are required by Yopy to store its Mobile Linux operating system and the core set of applications. The Pocket PC version of Windows CE is apparently lighter, as the H3600 requires only 16MB. The extra flash ROM adds $30 to Yopy's manufacturing cost, more than offsetting the $10-$20 license fee for Windows CE. (Mobile Linux is free, of course.) Yopy takes advantage of its relatively speedy CPU to offer a range of media functions such as voice recording, MP3 playback, video playback and games. It also includes a personal information manager (calendar, address book, etc.), e-mail, a web browser and other utilities. To take full advantage of the browser, users must add a modem using the Compact Flash expansion slot.

The H3600 and other Pocket PCs offer all of these features as well and, like Yopy, sell for about $500. Thus, any advantage for Yopy must come from Linux itself, either in ease of use or application availability. We were not able to test a Yopy unit, so we can't comment on its usability, an area where Pocket PC has gotten lukewarm reviews. When comparing application availability, one must remember that standard Linux and Windows applications must be modified to fit on a 240 x 320-pixel display. Microsoft has already modified key programs such as Word, Excel and Outlook; over time, the Linux community will develop a variety of applications for Yopy.

Yopy offers nearly 100 times more CPU performance than the popular Palm devices and 16 times more data memory than the Palm V. The Palm products use a smaller, lower-resolution display that, except for the new Palm IIIc, can't display colors. But the Palm V is smaller, half as thick and less than half the weight of Yopy. The Palm III and Palm V are also less than half the price.

Can Linux Win?

Most handheld computers sold today are from Palm or Handspring, a Palm-compatible vendor. These devices have gained popularity due to their relatively low cost and, more important, their ease of use. Over the past four years, independent programmers have developed thousands of Palm applications, many of which are freely distributed.

Yet Palm remains vulnerable. The only new features that Palm has added to its units since their 1996 debut are wireless connectivity and a color screen, and these features are currently available only in expensive, unpopular models. A true second-generation handheld could generate many new sales and tip the market away from Palm.

Linux and Windows CE already deliver next-generation features such as multimedia capability and expandability, and they are good platforms for adding voice recognition and wireless Internet access. Linux has the advantage of being freely available, including source code. Its larger footprint adds cost but, as the price of flash memory falls, this penalty will be reduced. Additional work on the code base could also bring Linux's code size in line with that of Windows CE.

Although Gmate www.gmate.co.kr is the first to deliver a Linux-based handheld, other vendors, both large and small, are investigating this opportunity to eliminate a Microsoft license fee. Therefore, more Yopy-like products are likely to appear. If the Linux community can rally around these tiny computers to develop an easy-to-use operating system and key applications, Linux could flourish in the palm of your hand.

email: linleyg@linleygroup.com

Linley Gwennap (linleyg@linleygroup.com) is the founder and principal analyst of The Linley Group (http://www.linleygroup.com/), a technology analysis firm in Mt. View, California. He is a former Editor-in-Chief of Microprocessor Report.

Linux at the University

Kevin K. Gifford

Issue #77, September 2000

In outer space, on the ground, and in the classroom: some exciting real-world applications developed with Linux by students and researchers at the University of Colorado in Boulder.

The Linux operating system (OS) coupled with powerful and efficient available software development tools, has been the platform of choice for several research and commercial projects at the University of Colorado located in Boulder. The projects include several payloads that have flown seventeen sortie missions on the NASA Space Shuttle. The payloads were built by BioServe Space Technologies, which is a NASA Center for the Commercialization of Space (CSC), and is associated with the Aerospace Department within the College of Engineering. BioServe's payloads have logged over a full year of combined operation in micro-gravity, including two extended stays on the Russian Space Station Mir. Student projects include an unmanned ground vehicle (UGV), which navigates autonomously based upon its assessment of dynamic environmental conditions.

Linux has proven to be an invaluable teaching instrument from a pedagogical standpoint at our university. Several classes are taught in the Aerospace Engineering Sciences and Computer Science departments that take advantage of the sophisticated and robust features of the Linux OS, not only as a development platform but as a method to teach and *implement* advanced topics in hardware/software integration, control theory, operating systems research and systems administration. This article details several of the projects from a scientific and engineering perspective and provides an overview of common software methodologies used in the development of the complex control schemes necessary for reliable and robust operation.

Linux in Outer Space

BioServe Space Technologies has built and flown several payloads that use the Linux operating system to interface with the control hardware, provide support for operator input and output devices, and provide the software development environment. The workhorse of the BioServe payload fleet is the Commercial Generic Bioprocessing Apparatus, CGBA, shown in its version 1 configuration in Figure 1. The operator data entry and visual feedback devices are located on the front panel. The keypad and mini-VGA screen are attached to an embedded control computer running Linux. The CGBA payload operates as an Isothermal Containment Module (ICM) with the capability to precisely control chamber temperature from 4 to 40 degrees C. Inside are the biological and crystal growth experiments loaded before launch. CGBA is a “single locker” payload (approximately 21"x17"x10") for insertion into either the Shuttle mid-deck or into large payload modules that fly in the Shuttle cargo bay.

The Fluid Processing Apparatus (FPA) shown in Figure 2 is the fundamental experimental unit of the CGBA payload. It is essentially a “micro-gravity test tube” which allows controlled, sequential on-orbit mixing of 2 or 3 fluids. The fluids are contained inside a glass barrel with an internal diameter of 13.5mm. Up to eight of the FPAs are loaded into a motorized Group Activation Pack (GAP), shown in Figure 3. Control software initiates the mixing of liquids containing different cell culture mediums at specified times once the payload has attained micro-gravity. A second motor initiation, which depresses a plunger in a manner similar to a syringe, combines the cell culture mixture with a fixative reagent, which squelches further biological reaction. The flight samples are then analyzed in comparison with identical samples in ground-based experiments to determine the effects of micro-gravity upon fundamental biological processes. Pharmaceutical and crystallization investigations are common candidates for these types of experiments; a small increase in the efficiency of a reaction can lead to a significant increase in the desired products, and thus to a company's profit margin.

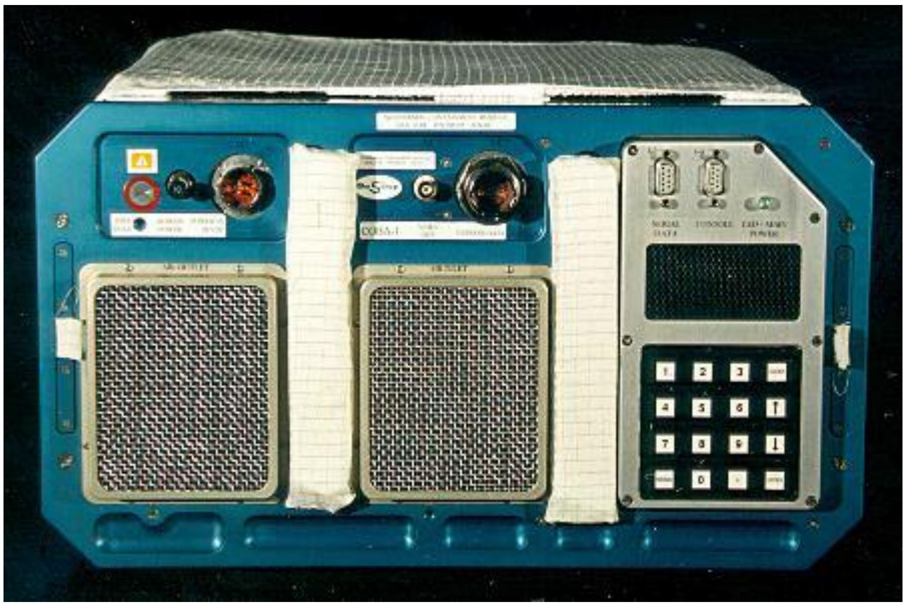


Figure 1. Commercial Generic Bioprocessing Apparatus

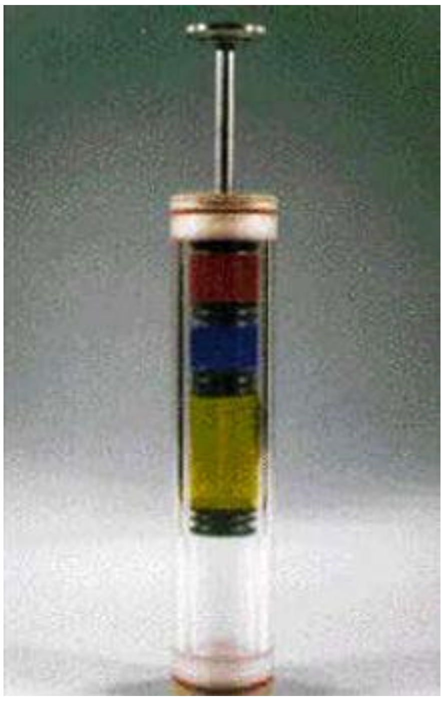


Figure 2. Fluid Processing Apparatus

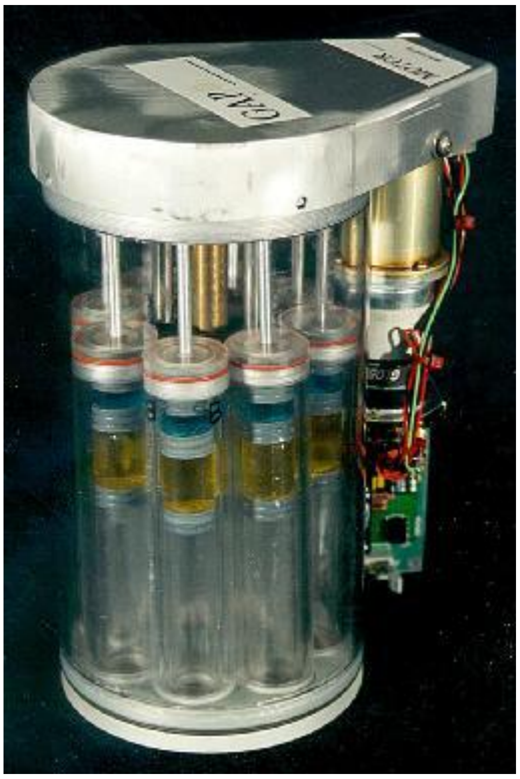


Figure 3. Group Activation Pack

A second version of the CGBA payload highlights a configurable internal palette, upon which several different habitat chambers can be built. This payload contains multiple video cameras to provide time-sequenced image recording of the habitat contents. BioServe has used this payload configuration, in conjunction with SpaceHab, Inc., to provide exciting space-based educational outreach to primary and secondary school students across the world. SpaceHab's Space Technology and Research for Students (S*T*A*R*S) program supports a wide array of potential experiments for investigation.

In July of 1999, two experiments flown in the payload onboard Shuttle mission STS-93 involved students from high schools in the United States and students from a secondary school in Chile. As shown in Figure 4, this payload had three center habitats which contained ladybugs and aphids while the larger chamber in the front left housed butterflies. The first experiment, of interest to students in the U.S., involved the incubation and development of pupae larvae from a crystalith (cocoon) to the maturation stage of a butterfly. The second experiment, designed entirely by a group of teenage girls in Chile, investigated the effects of micro-gravity upon the predatory relationship of ladybugs and their favorite prey, the aphid, which is notorious for its devastation of food crops. Figure 5 shows aphids, the small black objects on immature wheat plants before the introduction of predatory ladybugs into the habitat chamber. The same habitat after the release of the ladybugs is pictured in Figure 6, showing a (expected) decrease in aphid population. Note the ladybug in the middle right of the image as it attempts to catch an aphid in microgravity. The results of this experiment hope to gage the potential for using natural predator/prey relationships as a means to minimize detrimental insect destruction of both terrestrial and space-based food crops.

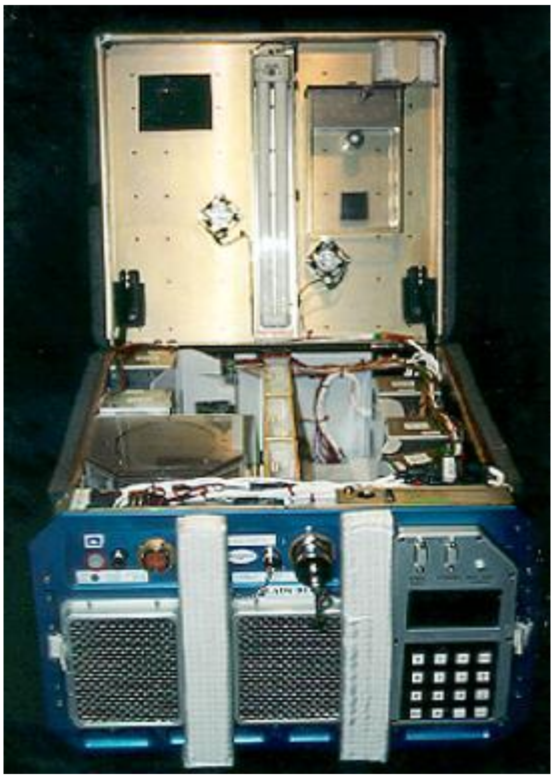


Figure 4. The CGBA-02 Payload for S*T*A*R*S

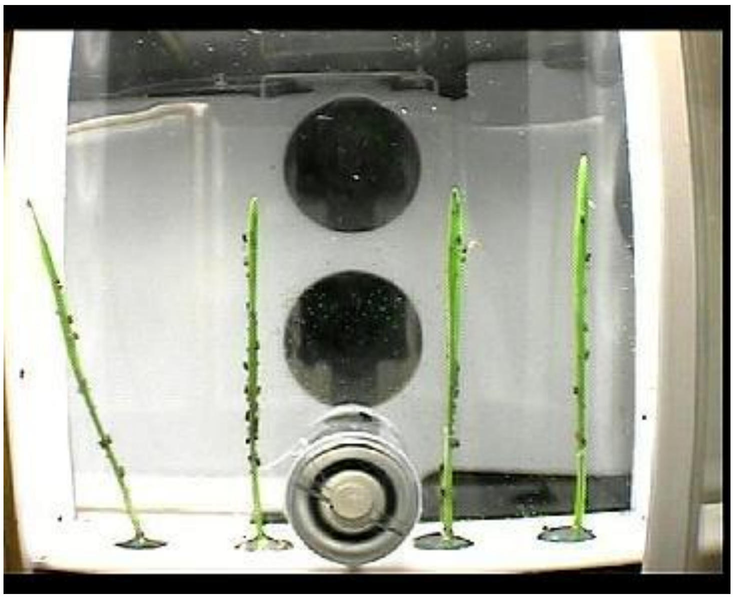


Figure 5. Habitat with Aphids on Wheat

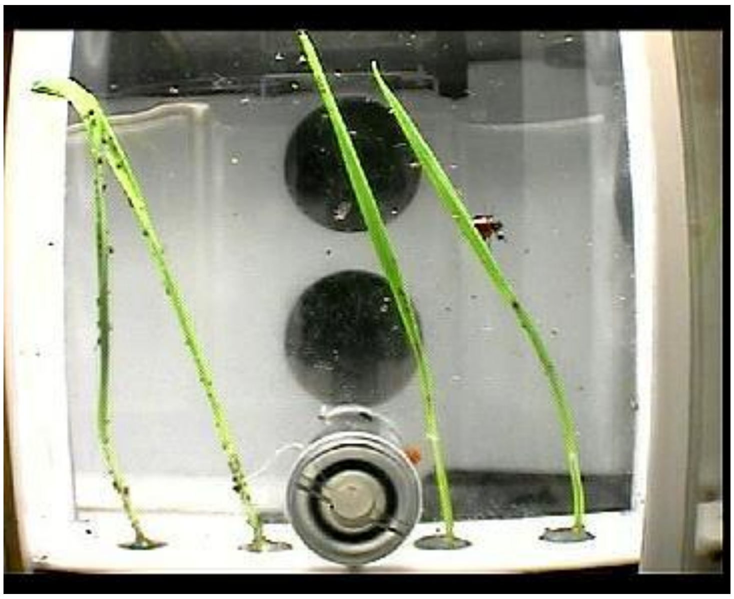


Figure 6. Same Habitat after Introduction of Ladybugs

The second S*T*A*R*S flight, to be flown aboard Shuttle flight STS-107 in 2001, will support an international array of educational science experiments. Participation in this program is literally worldwide. The educational departments of numerous countries submitted proposals for selection, from which six experiments have been chosen. Australia will fly an AstroSpider experiment, which is a collaboration between Glen Waverely Secondary College and researchers from the Royal Melbourne Institute of Technology. They will investigate how the mechanical properties of the spider's silk, which can be 100 times stronger than steel, differ when it is produced in microgravity. Jingshan High School in China has proposed an experiment which will investigate one of their country's icons, the silkworm. Their experiment will attempt to reveal the effects of microgravity on silk production and the metamorphosis process as the silkworms transform from worm-like larvae, spin cocoons and then emerge as adults. Two of the new S*T*A*R*S experiments will be a joint effort involving investigators in Japan and the United States. Both groups will each fly an enclosed, aquatic ecosystem that contains plants and tiny aquatic animals. They hope to investigate the navigation and locomotive processes of these animals as they swim in microgravity. Israeli science students from Motzkin Ort Junior High School are designing their experiment to investigate the formation of crystals in microgravity. They hope to see differences in the shape, structure, strength and size of space grown crystals over similarly grown, earth-based crystals. The final participant in this flight is Fowler High School from Syracuse, New York. In conjunction with Syracuse University, they are designing a zero gravity ant farm. They hope to chart differences in the species' well-characterized social activities brought on by the disorientation of space flight. Recently, a third S*T*A*R*S mission has been scheduled for the year 2002 onboard the International Space Station, which will allow longer-term scientific investigations to be conducted.

The CGBA payload, in its version 3 configuration (see Figure 7), allows for individual temperature control of each GAP container by coupling it to a heat sink (water loop) with thermoelectric modules and insulating each GAP from one another. The water loop is distributed on the top and bottom of this locker to virtually eliminate thermal gradients and to provide efficient temperature control of the individual sample container. These individual temperature controlled containers allow for time sequenced experiment initiation and termination. The National Institute of Health will fly an experiment investigating the development of fruit fly maturation, from larvae to adults, using this configuration. The fruit fly maturation process can be effectively controlled by warming Petri dishes contained in the individually temperature controlled GAPs at different times using pre-programmed temperature profiles. The fruit fly has a scientifically well-documented development process. Scientists have been able to genetically splice fruit fly neurons with fluorescent markers and are interested in the development of neural based muscular activity. The research is aimed at investigating the difference in neural pathway development between space-based samples as compared to similar ground-based control samples. The scientific goal is an increased understanding of how gravity affects neural development.

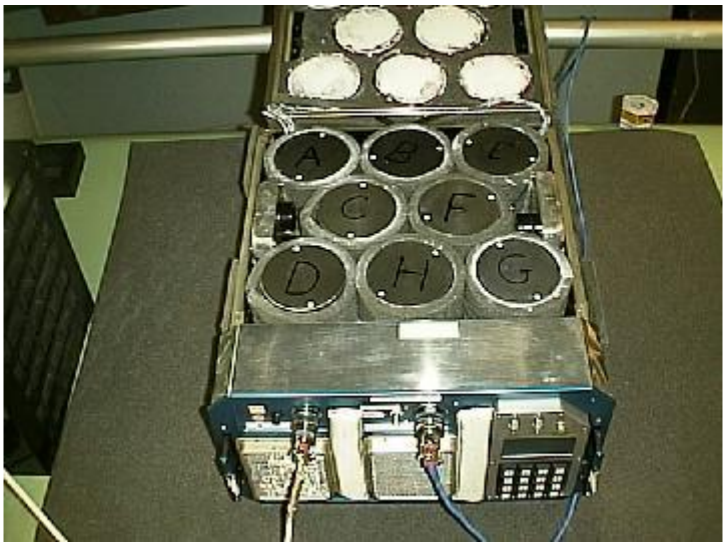


Figure 7. CGBA-03 Payload with Temperature Controlled Containers

The Plant Generic BioProcessing Apparatus (PGBA) (see Figures 8 and 9) is twice the volume of the “single locker” CGBA payloads described previously. It is designed to precisely control the environmental conditions of temperature, humidity and oxygen/carbon dioxide levels within the plant growth chamber for fundamental space-based research in the plant sciences arena. Plant science is fundamental for many large commercial entities including the pharmaceutical and timber industries. Both of these commercially important industries are, in part, based upon plant-produced compounds. For example, the timber industry funds significant research programs investigating the production of lignum which is responsible for providing structural strength for both plants and trees. From a commercial perspective, lignum is undesirable for the formation of paper products but a desirable constituent for strong timber products. In micro-gravity, a plant's need for structural strength is greatly diminished. Current research focuses upon the change in lignum production in micro-gravity based plants and trees to determine potential methods for regulating the compound's terrestrial in situ production.

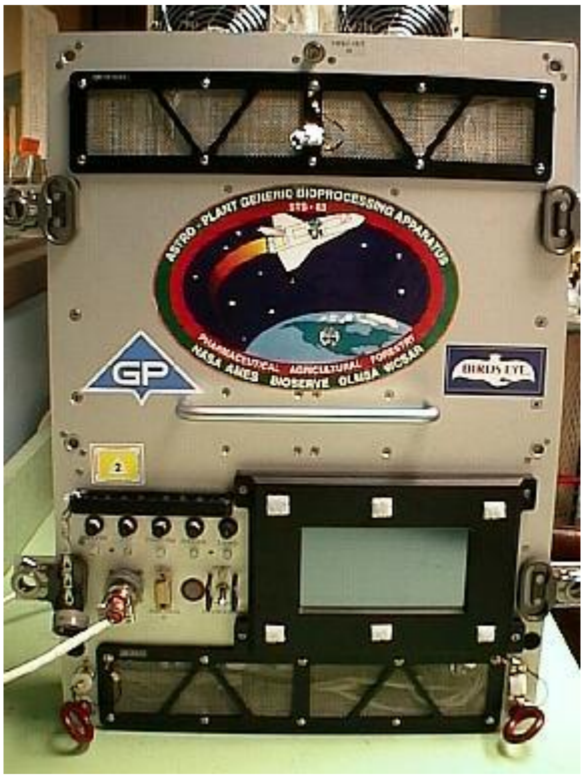

ASTRO-PLANT GENERIC BIOPROCESSING APPARATUS
STS-63
PHARMACEUTICAL AGRICULTURAL FORESTRY
NASA AMES BIOSERVE OLNSA WCSAR
GP
BIRDS EYE

Figure 8. Front Panel (PGBA) with Touchscreen in Lower Left

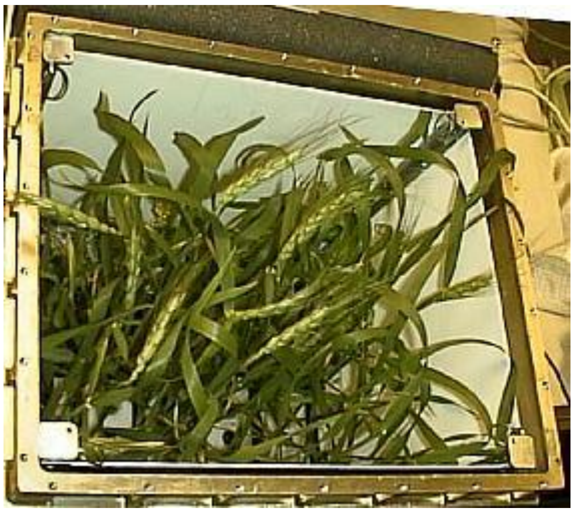


Figure 9. Wheat Plants in Environmentally Controlled PGBA

The Fluid Generic Bio-Processing Apparatus (FGBA) (see Figure 10) is interesting from both the research and commercial perspectives. FGBA was developed in conjunction with the Coca-Cola Company and was flown onboard STS-77 in 1996. FGBA was used to provide basic research about human taste perception, which changes in micro-gravity. The payload also investigated fundamental relationships pertaining to two-phase fluid flow and dispensing in microgravity, as the beverages were mixed and dispensed using pressurized carbon dioxide in a manner similar to earth-based soda fountains. From the commercial perspective, this payload demonstrated the potential for the funding of space-borne projects based not only upon their fundamental research potential, but also on their capability to bring commercial advertising dollars to the cash-strapped International Space Station.

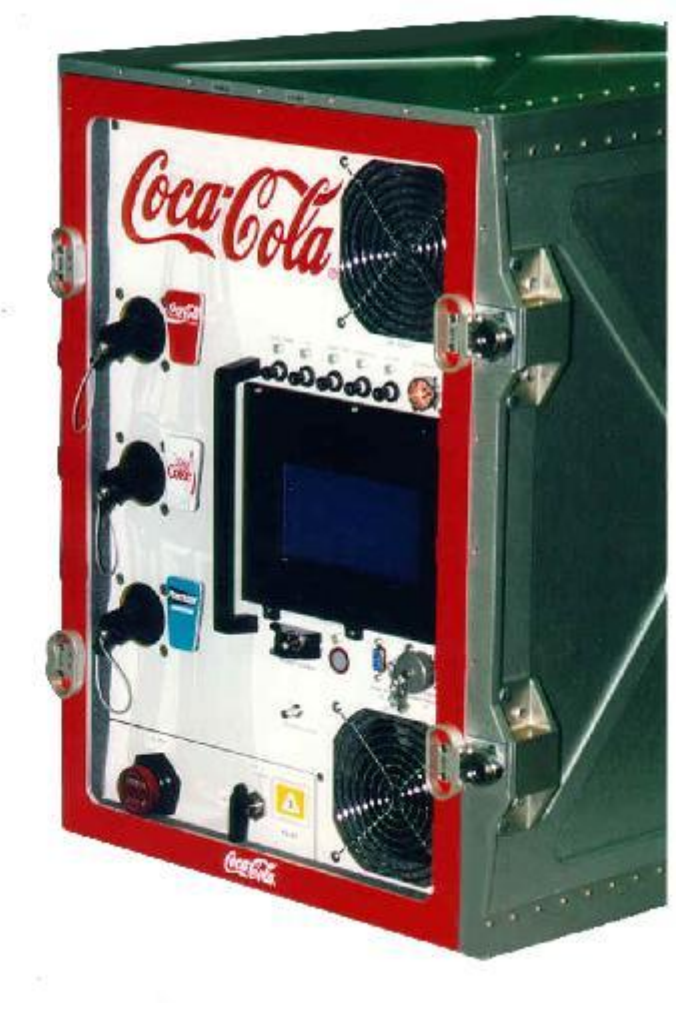


Figure 10. (FGBA) Developed with the Coca-Cola Company

In all of the payloads, an accelerometer-based system is used to detect launch, thus allowing experiment initiation (motor-activation, temperature change, lighting conditions) immediately upon entering orbit. Additionally, automatic experiment termination can be programmed to occur at any time during the mission, including just prior to reentry, based on pre-planned (or updated) Shuttle end-of-mission time. These two capabilities combined allow an early-as-possible experiment initiation and a late-as-possible termination, since those are periods when astronaut crew availability for payload tasks is typically at a minimum.

Design Issues

Linux was chosen as the operating system for the payloads onboard the Shuttle (and soon to be International Space Station) missions for several reasons. First and foremost is its high reliability. In several years of developing different payloads, we have never encountered a problem that was operating system related. Other reasons for use of the Linux OS include:

- Ease of custom kernel configuration, which enables a small kernel footprint optimized for the task at hand;
- The ability to write kernel-space custom device drivers for interfacing to required peripheral hardware such as analog and digital I/O boards or framegrabbers;
- The ability to optimize the required application packages resident on payload non-volatile storage media;
- The availability of a powerful software development environment;
- Availability of the standard UNIX-like conventions of interprocess communication (IPC) methodologies, shell scripting and multi-threaded control processes.

With the recent advances and capabilities of real-time variants of the Linux OS, critical real-time process scheduling is available (if needed) for incorporation into the software/hardware control architecture.

Because of the limited volume of these complex payloads, it is necessary to minimize the volume of every hardware item, including the communications and process control computer. In our payloads, we use a combination of small single board computers (SBC) and PC-104 form factor boards to supplement the capability of the SBC. In particular, we have had great success and outstanding reliability from Ampro, Diamond Systems, SanDisk and Ajeco products.

The computer hardware architecture we have designed and developed is generic in that the same architecture is used for all of the different payloads. The single board computer provides the CPU, FPU, video interfaces, IDE and SCSI device interfaces, along with the serial and Ethernet communications interfaces (see Figures 11 and 12).

The inside of the control computer assembly, with a single board computer and stack-on PC-104 module, is pictured in Figure 11. The unit is 6 feet wide and 2.5 feet deep. Figure 12 shows the outside front of the control computer assembly with the keypad and mini-VGA attached. Analog and digital IO capabilities, necessary for interfacing to control hardware and sensors, are implemented via an additional PC-104 module. Imaging of the experiments within the payloads is accomplished via multiple cameras interfaced to a PC-104 frame-grabber. This generic hardware architecture allows for the integration of a complex array of sensors and control actuators. Additionally, distributed processing is implemented by attaching micro-controllers over a serial RS485 multi-drop communications bus, which facilitates the addition of new control hardware in a modular manner.

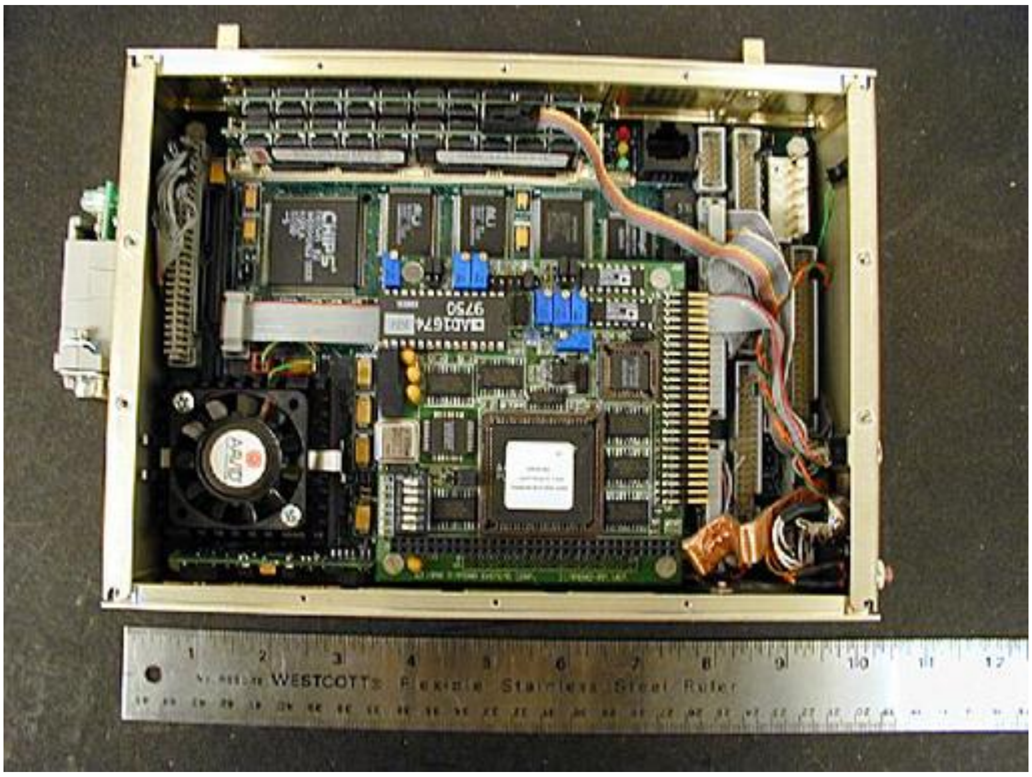


Figure 11. Inside of the Control Computer Assembly

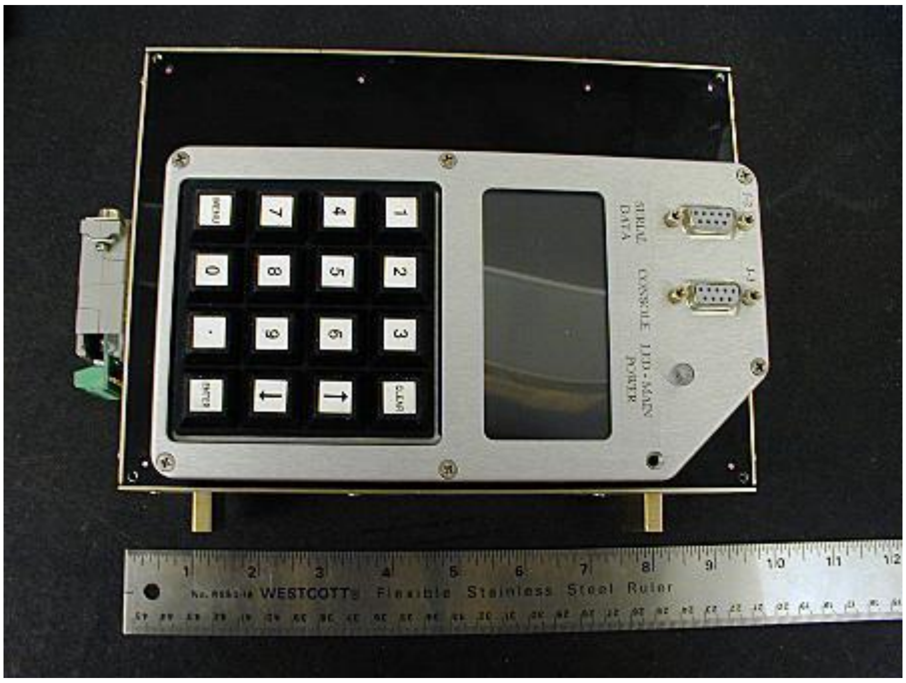


Figure 12. Keypad and Mini-VGA Screen

One tremendous advantage provided by the true multi-processing Linux environment is the relative ease of software development as compared to monolithic DOS systems. A similar generic, multi-threaded software control architecture is used for process control and communications for all of the projects described in this article and shown in Figure 13.

The software architecture follows the multi-threaded peer model. At start-up, the thread responsible for initialization of the entire system first spawns the processes required for payload-to-groundside communications. Next, shared resources are requested and initialized. Communication and synchronization control variables are initialized, and then a separate thread is created corresponding to the different services required from the payload. For our payloads, these services are categorized as either primary or secondary services. Primary services include those tasks which are required for the fundamental control of the payloads. Primary services for the different payloads tend to be identical in functionality, but the methods of hardware implementation can differ greatly. Secondary services are those specific to a particular payload.

From the development standpoint, the use of threads in a proven multi-process architecture facilitates multi-programmer development, module reuse and a much faster test-and-debug cycle, since processing of individual threads is easily enabled or disabled. This allows for complete testing of a single subsystem, say the chamber temperature control, in a stand-alone manner. Once completely tested, the subsystem is incorporated into the overall payload control architecture, thus minimizing adverse effects of integration.

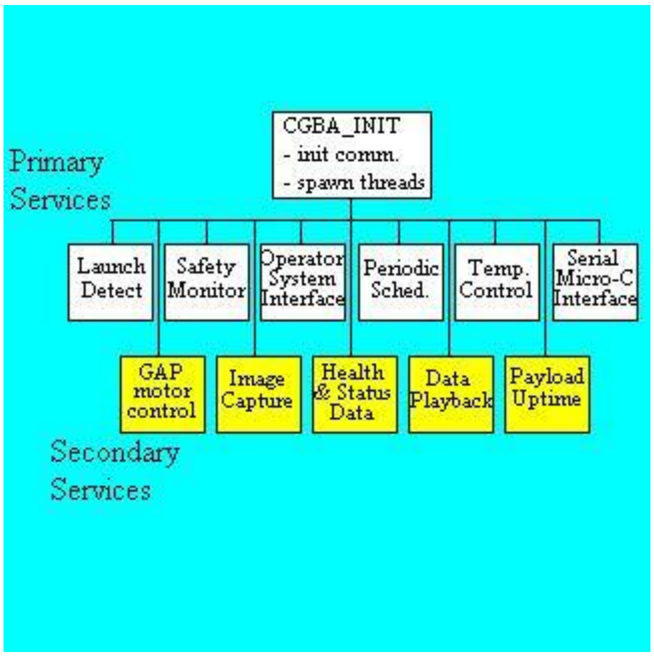


Figure 13. Simple Chart of Software Architecture.

The communications system is implemented in a modular manner as well. Data transmission is abstracted at the programming interface level as different messages (e.g., analog data, health and status messages, images, etc.). These different length messages are termed “channels”. A thread that wishes to transmit a message to ground stations, queues (writes) a dynamic message of pre-defined length to a specified channel (see Figure 14). The channeling system prepends channel routing information and a packet checksum. Multiple channel messages are combined into a secondary packet to maximize usage of available bandwidth. This secondary packet is termed a “RIC” packet; it serves as an interface abstraction to the Rack Interface Computer which is used onboard the International Space Station to communicate to the different payloads. RIC packet contents are also sanity checked via an additional checksum. Finally, the RIC packet is encapsulated into a TCP/IP packet for delivery to the payload.

Communication between the RIC and the payloads is TCP/IP based for reliable communications. Unfortunately, telemetry data transmitted from the payload to the RIC ends up being distributed on the ground from the Marshall Space Flight Center to the remote users via UDP, which is unreliable. In addition, the physical communications link in the form of the TDRSS satellite network is subject to outages as well. To minimize the potential impact of dropping packets and loss of signal caused by holes in the available TDRSS satellite network coverage, a “playback” system has been implemented that retransmits critical information to improve the probability that a complete set of telemetry data files is received properly by the ground stations.

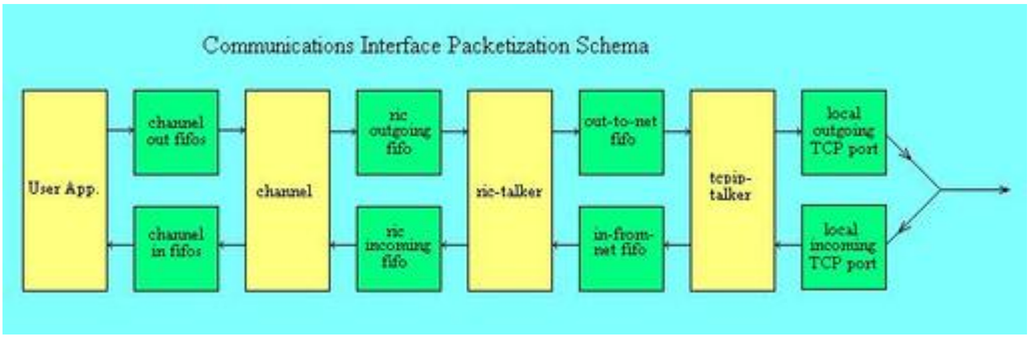


Figure 14. Interface for Payload to Groundside Communications

Finally, one of the channels has been configured to serve as a remote shell on the payload. Input to and output from the bash shell is first redirected to a channel and then packetized as described above. In this manner a true remote shell operates on the actual payload in a manner that is functionally equivalent to establishing a remote connection via TELNET or ssh. The bash shell input and output is “tunneled” through the NASA communications scheme and allows the possibility of remote system administration activities from the ground. Just try and implement that capability under Windows! For further information concerning the design of the communications subsystem, please see the excellent article “Linux Out of the Real World”, by Sebastian Kuzminsky, which appeared in *Linux Journal* in July of 1997.

Linux in the Classroom

As the economy of the United States transitions from the Industrial Age to the Information Age, students from all engineering disciplines are required to enable the automation and control of large and small industrial production processes. The per dollar cost of storage media, CPU cycles and interface hardware continues to decline, allowing research agencies and for-profit companies to implement automation of their production processes in an effort to decrease costs and increase profit margin. At the College of Engineering and Sciences, located on the main campus of the University of Colorado, a new teaching paradigm has been realized with the completion of the Integrated Teaching and Learning Laboratory (ITLL). The ITLL was built as a resource for students of all engineering disciplines to facilitate the transfer of theory learned in the classroom to actual practice in the laboratory, in an effort to teach real world engineering and scientific skills.

The ITLL provides the resources to allow students to design and build complex engineering projects. A typical lab station is shown in Figure 15. Each of the lab stations is equipped with a Hewlett-Packard Vectra PC with a 500MHz Pentium-III processor, 384MB of RAM, an 8GB hard drive, the usual NIC and a National Instruments analog and digital data acquisition card. Several of the 80 available systems have been configured to boot Linux and RTLinux, in addition to Windows NT. Standard test equipment at each lab station includes a waveform generator, digital multi-meter, variable-voltage power supply and oscilloscope. The breakout panel provides easy access to analog and digital IO signals and to signal-conditioning hardware. The availability of sophisticated test equipment, coupled with the capabilities of the Linux and RTLinux operating systems, has enabled students to design and construct complex, real world applications integrating both hardware and software to accomplish a specified engineering or scientific task.

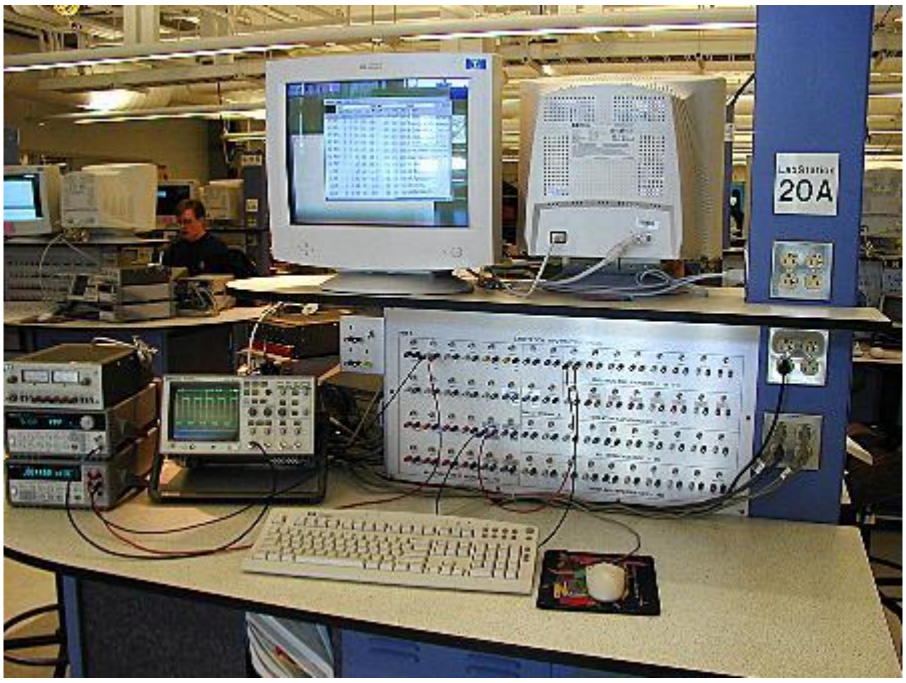


Figure 15. An ITLL Workstation Running Linux/RTLinux.

One such project is the University of Colorado's Robotic Autonomous Transport, a.k.a. the RATmobile, shown in Figure 16. The RATmobile is an unmanned ground vehicle (UGV) that navigates autonomously by sensing static and dynamic features within its environment. This vehicle was designed entirely from scratch by a group of undergraduate and graduate students from several engineering disciplines with supervision and guidance from members of the Aerospace department. Design and construction of the vehicle is truly multi-disciplinary, calling upon the theory and expertise of mechanical, electrical, aerospace and computer sciences.

The vehicle uses miniature video cameras mounted on the front to supply visual information concerning the surrounding environment. Static and dynamic obstacle detection is supplemented by an array of ultrasonic sensors. The sensed environmental conditions are combined into a local coordinate frame and then translated into a world representation map that contains the totality of the robot's knowledge concerning its environment. Depending upon the task at hand, the robot is able to make use of different navigational algorithms to traverse to a desired destination.

One such navigational mode allows for the vehicle to navigate an obstacle course that is delimited by painted white lines on a large field. The lines are approximately ten feet apart with straightaways, sharp curves and bends and even some occurrences of missing lines, which require that the navigation algorithm predict the desired vehicle direction based upon historical information. A second navigational mode makes use of differential GPS information for precise assessment of the robot's position in a world reference frame. In this mode, the vehicle has a priori knowledge of its static environment and can navigate to a predetermined location while avoiding unmapped or dynamic obstacles. Different navigational strategies, such as the optimization of path length or required path energy, are easily implemented.

Linux was chosen as the operating system for this project because it met all design requirements, allowed for a multi- user development environment and was free, which is of concern to students with a limited project budget. The capability of the multi-threaded software control architecture provided by Linux allows for easy implementation of differing vehicle capabilities. The primary functions of the vehicle, visual image acquisition; obstacle detection information acquisition; combination of input sensory data into a world representation; and the determination of the required navigational control outputs of velocity and steer angle, are all abstracted to different threads. If necessary, the CPU scheduling policy associated with different processes can be altered from the default scheduler, which optimizes for the average case, to allow for improved real-time performance under the standard Linux OS. In process control systems, where millisecond and below guaranteed scheduling is required, a real-time Linux variant, such as RTLinux, can be used.

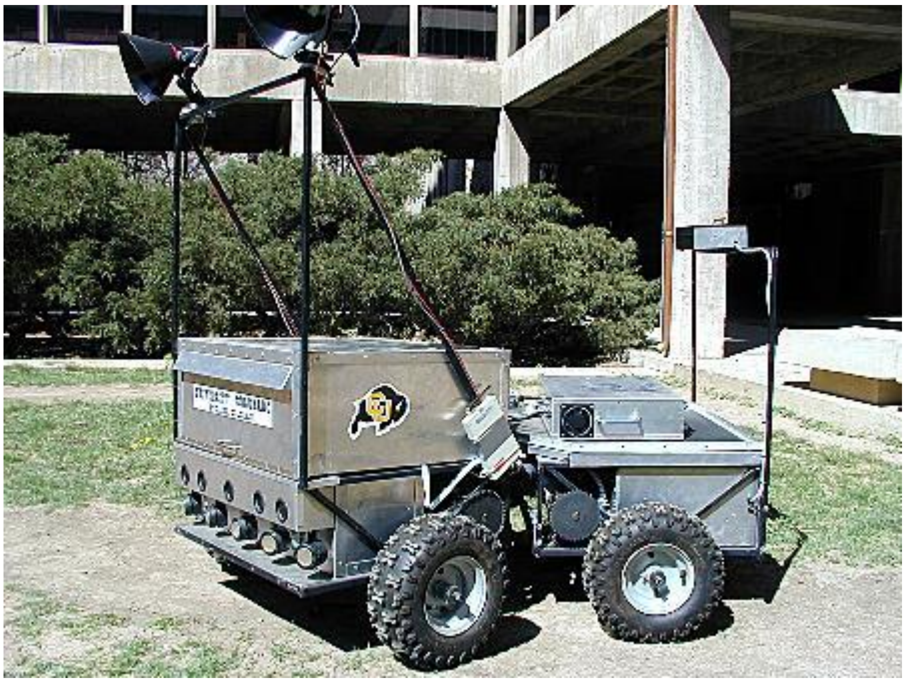


Figure 16. The Robotic Autonomous Transport (RATmobile)

The ITLL stations, running either Linux or RTLinux, are also used in a class emphasizing hardware/software integration skills to undergraduate and graduate students. Advanced programming constructs, in the C and C++ languages, are required by laboratory exercises that emphasize the integration of real hardware, such as motors and micro-controllers. This allows students to develop confidence and expertise not just in proper software programming practices, but in combining control hardware with the standard PC. Examples of final student projects (all using Linux or RTLinux) in this class include:

- A working mockup of a spin stabilized satellite that uses image processing to determine orientation within a star field.
- A directionally controlled scanning laser used to determine mechanical vibrational modes within a structure.
- Transmission of real-time accelerometer data from a radio controlled airplane with subsequent real-time plotting capability.
- The construction of a fluid chamber to model fluid flow through different chambers of the heart with associated real-time data gathering for analysis.

Using the Linux OS and user-configurable development environments as a basis, we are able to impart a greater breadth of knowledge to our students, which is applicable to real world engineering problems. That is a true accomplishment whose accolades belong to the numerous kernel and development programmers whom have contributed to the power and success of Linux.

Linux also serves as a powerful teaching tool within our Computer Science department. The free availability of the Linux OS and associated development packages have opened the doors to a greater variety of course content in classes pertaining to Systems Administration. With a free operating system and the availability of cheap PC hardware, students have the ability to configure and administer all aspects of their very own system within a university computer lab. If something goes dramatically wrong, it is a simple matter to restore the entire disk across a network from a master system. This encourages students to experiment with no disastrous repercussions. Topics such as disk-level file recovery, device installation and management, user and file administration, kernel configuration and networking are all candidates for easily constructed exercises using the free Linux OS as the enabling technology.

Open source distribution of the Linux kernel source tree has allowed courses in Operating Systems Theory to not only teach the fundamentals but, even more importantly, allow the students to experiment with changes to the kernel code. As a result, students are offered a richer set of exercises that can be applied directly to the kernel source and not just on paper. Most assuredly, students will break their systems while experimenting, but system recovery strategies have definite educational value—better now on a cheap PC running Linux than on some multi-million dollar platform in the future. Having the kernel source allows for in-depth exercises in far-ranging topics including management of processes, resources, devices, files and memory, as well as implementation of kernel-space processes as Linux modules, scheduling policies and priorities, and fundamental synchronization methodologies. Linux enables a richer and more realistic educational experience for our students.

Final Thoughts

When designing and building complex systems, which require low-level integration of software and hardware, there are numerous design trade-offs and decisions to be made. Two primary criteria that are always evaluated are: can the system meet the specified performance requirements, and can the system meet imposed budgetary constraints? Based upon past experience, both Linux and its real-time variants meet or exceed the typical performance metrics. The development environment is powerful and easily configurable. Operating system services are ample, and resource abstraction follows the standard, simple, UNIX-like methodology. One design issue that is of fundamental concern to companies within the commercial sector is that of technical support. With the advent of Linux-only technical support companies, along with an increased base of competent Linux programmers, this issue is dwindling in comparison to the numerous advantages of a free, open-source operating system. Budgetary constraints are a non-issue. A world-class group of kernel hackers continue to provide support for cutting-edge technologies and hardware. Students of today, who are the programmers and systems managers of tomorrow, are being taught to harness the enabling power of Linux.

Summary

Linux is used for a myriad of research, commercial and educational purposes at the University of Colorado. The systems designed and built are complex and, must be reliable. The payloads BioServe builds are currently scheduled to spend over three years in combined time onboard the International Space Station, and computer control, implemented under the Linux OS, is a mission-critical design specification. From the performance of past payloads, in addition to the complex projects previously described, the author is confident of continued success. Linux is the enabling resource that allows the integration of these projects. Linux has provided meaningful content for several courses taught within our Aerospace and Computer Science departments and is responsible, in part, for the success of our real-world teaching paradigm, which takes theory out of the classroom and allows for practical implementations of advanced projects and research.

Resources

Acknowledgements

email: gifford@rintintin.colorado.edu

Kevin Gifford (gifford@rintintin.colorado.edu) is the lead Automation and Robotics Engineer for BioServe Space Technologies and is a member of the Research Faculty of the Aerospace department at the University of Colorado where he teaches classes in Hardware/Software Integration and mentors Senior Design Projects. Like a lot of Linux enthusiasts who love their job, he needs to diversify his life with activities that have a satisfaction metric other than increased technical competency.

Archive Index Issue Table of Contents

Advanced search

Focus on Software

David A. Bandel

Issue #77, September 2000

Meeting Room Booking System, W3M, tradeclient and more.

Well, I'm really excited about the new kernel. The netfilter software (iptables) which replaces ipchains (I know, yet another packet filter to learn, but it's not that bad, really) will save me from having to configure both ipchains and ipmasqadm (at least for those systems doing port forwarding as well as packet filtering). I have more and more clients all the time who seem to need this capability. Combining these two (basically NAT and packet filtering) into one integrated bundle will make administration easier. And ease of administration is where I win clients away from Microsoft. I anticipate a webmin module (a framework consisting of modules you can add or remove) will be available to handle the netfilter rules (even if I have to write it myself, and no one, not even me wants that—at least, not if you've ever seen my Perl code). And the fact that the new netfilter also includes support for IPv6 is icing on the cake. I'm constantly amazed at the pace of improvements, not just in the kernel, but in all the software available for Linux today, as can be seen by looking at newer versions of past FOS software highlights.

Meeting Room Booking System: http://mrbs.sourceforge.net/

I know of several places that could use a good meeting room booking system. I've seen a couple that work well, but this one certainly deserves a look. It really doesn't matter where the room to be reserved is; MRBS handles it very nicely. Multi-hour events are handled well, and the overall look is appealing. It requires a web server with php3, MySQL and a web browser.

W3M: http://ei5nazha.yz.yamagata-u.ac.jp/~aito/w3m/eng/

My favorite browser has always been Lynx. It's fast, and I've always built it with SSL. But it doesn't support frames. This has always been somewhat annoying, especially given the number of sites that use frames. Well, w3m supports tables and frames. The frames are handled by converting them to tables and displaying them as such. I was also pleased to see that you could compile in mouse support, colors and SSL. In fact, you could choose the build size, which determined the particular options, or choose a custom build to mix-and-match options. On the downside, I did find that in an xterm, I had to either vary the width of the xterm or scroll across. On a VT, you can only scroll (unless you already have a wide page via frame buffers). This can be annoying if you're used to Lynx automatically sizing to the screen. But that's due to the use of tables, so it is the price you have to pay. I requires (depending on the build size you choose) libm, libgpm, libnsl, libncurses, glibc and openssl (for SSL sites).

tradeclient: http://www.sourceforge.net/projects/tradeclient/

It slices, it dices, it keeps your Calendar, ToDo list, Addressbook, it makes coffee and pays your bills (okay, so it won't pay your bills). I showed this to a client, and their first reaction was: When did Microsoft port Outlook to Linux? Well, I'm afraid I don't know Outlook from Adam (and for that matter, I have had the singular pleasure of not having to work with anything Microsoft in almost a year), but I do know this mail client includes everything but the kitchen sink (if you consider that a plus) and has a very pleasing interface. Try it if you like the “no need for any other software” approach to programs. It requires libgtk, libgdk, libgmodule, libglib, libdl, libXext, libX11, libm and glibc.

webmail.pl: http://opensource.jaos.org/

If you need a good web-mail client that won't strain your resources, this is one you'll want to take a look at. It doesn't have an address book, or folders for saving messages; in fact, it doesn't have much of anything. What it does have is a good-looking interface to read and send mail, period. Perfect for an ISP short on resources. It requires Perl, a web browser, a web server with Perl support, Perl modules: CGI, Mail::POP3Client, Socket, MIME::Base64 and Crypt::Blowfish.

linuxinfo: www.tahallah.clara.com.uk/programming/prog.html

For this program, the three-line output says it all: Linux chiriqui.pananix.com 2.4.0-test2 #2 Sat Jun 24 16:19:55 EST 2000 One Intel Unknown 333MHz processor, 665.19 total bogomips, 128M RAM System library 2.1.3. You can even use it in a server-parsed web page (.shtml) by using the following line:

```
  <!--#exec cmd="/usr/local/bin/linuxinfo" -->
```

Since the output doesn't contain any html commands, you might want to surround the above line with <pre></pre> tags, and perhaps even center tags. It requires glibc.

ftpgrab (download only): ftp.lmh.ox.ac.uk/pub/linux/ftpgrab-0.1.2.tar.gz

Another site-mirror program that currently supports only ftp, but HTTP support is planned. This one is different, in that instead of mirroring all files, ftpgrab will mirror only the latest file version by parsing the version number. This can be very resource-saving if you have sufficient file space for only the latest version. It downloads the most current, then deletes the older one. This can be quite a savings with files like gtk+ and others. It requires libpthread, libstdc++, libm and glibc.

quizzy: http://www.soomka.com/

Okay, maybe you don't need any more tests in your life, given all the tests out there already (LPI, A+, etc.). But if you can find some sample questions, you can test yourself. Questions are simple to add to a file. The program even comes with a script to add questions, although the author suggests using vi. It's multiple choice only, no fill in the blanks, but you can have multiple correct answers (the number of required answers is shown). The program doesn't yet save scores, so you'll need to note your score before leaving the screen. The “examinee” can also cheat. But the instant feedback on right/wrong can be instructional. It requires libncurses, libmenu and glibc.

scanlogd: http://www.openwall.con/scanlogd/

Want to monitor port scans on your systems without the complexity of PortSentry? This logs all port scans. With the proliferation of juveniles who have nothing better to do than run nmap and other scans against networks and systems the world over, this can give you an idea of how much of a target you are. It's only one tool, but a good one. It requires glibc.

email: dbandel@pananix.com

David A. Bandel (dbandel@pananix.com) is a Linux/UNIX consultant currently living in the Republic of Panama. He is co-author of Que Special Edition: Using Caldera OpenLinux.

Internet Appliances with a Twist

Rick Lehrbaum

Issue #77, September 2000

Adomo uses a client-server approach.

Adomo (http://www.adomo.com) wants to fill your home with a network of low-cost, easy-to-use information appliances. Best of all, they'll each have Linux inside.

Adomo's plans are to transcend the boundaries of the PC revolution, making home information appliances as easy to use—and ubiquitous—as microwave ovens. To accomplish this, they chose to redefine the very nature of home information systems. They started out by asking, “What do family members really want?”

A “Back to the Future” architecture.

That quest led to a product architecture that runs counter, in many ways, to the current PC paradigm. Contrary to the popular notion of intelligent interconnected appliances as the wave of the future, Adomo's architecture is a lot like the “pre-PC” multi-user computers and dumb terminals of 25 years ago—but with some important differences.

Adomo's solution creates a home network that consists of two types of devices: a highly intelligent central device (the “server”) which provides computing, information storage and Internet access resources; and multiple wirelessly linked user-interface devices (“clients”) distributed throughout the home.

But there's an important twist: although the user-interface devices function like intelligent information appliances, they don't actually run any application software internally. They just pass data back and forth to the central server.

This approach can work—since today's high speed networks can provide near-instantaneous access to massive amounts of computing power and vast quantities of data. In other words, as Sun puts it, “the computer is the network”. That being the case, the distinction between server and client becomes moot!

This way, the CPU within the appliance needn't be capable of running sophisticated programs. Its main purpose is managing input/output functions, which are mostly user-interface-related. Each appliance runs just one program: a reduced-footprint version of the X Window System. All the application programs run on the central server. Only display output (graphics, sound) and user input (keyboard, mouse, sound) travel back and forth between the server and the appliances.

Thin Is In

By restricting their function to input/output only, Adomo's extremely “thin” information appliances become simplified to the point where they are mere interface terminals rather than computers (even though they do contain microprocessors). This has some interesting consequences.

One is that changes in context—between family members, or between OFF and ON—are nearly instantaneous. Since the appliance is just an interface to software running on the server, all appliances can be used interchangeably. Just sign on, and pick up where you left off. No need to wait around while the OS inside the user interface of a refrigerator or microwave logs out one user, and prepares itself for access by another.

Another big advantage of extremely thin appliances is that they're smaller and cost less. Also, they generate less heat, and run longer on a battery charge.

An interesting side effect is that because the appliances don't contain their own application software, there is hardly anything to go wrong from a software perspective. All they need to do is be able to load their operating software from the server, across the wireless LAN. That, plus their accompanying electronic simplicity, means the appliances are likely to be completely free from the headaches of maintenance and administration—or at least that's the plan.

Even system administration is simplified, since all software for operating the entire home system is concentrated on the server, a Linux system. That should make remote maintenance and support a snap.

Of course, for the Adomo client/server scheme to work, the server had better be powerful enough to handle the activities of all the client devices located throughout the home. Also, the wireless LAN needs to have sufficient bandwidth to carry all the associated data transfers. Adomo claims the performance of their server will support the simultaneous activities of up to eight “typical” family members.

Does that include teenagers playing the latest video games? Probably not. For that reason, Adomo also lets you use one or more normal PCs on the network. In that case, the Adomo server provides connectivity, application sharing, file storage and backup for all devices—both PC or non-PC.

The Server

The Adomo server is functionally similar to a high-end PC running Linux. But, as you can see from the photo (see Figure 1), it looks more like an appliance (possibly an air conditioner) than a PC.

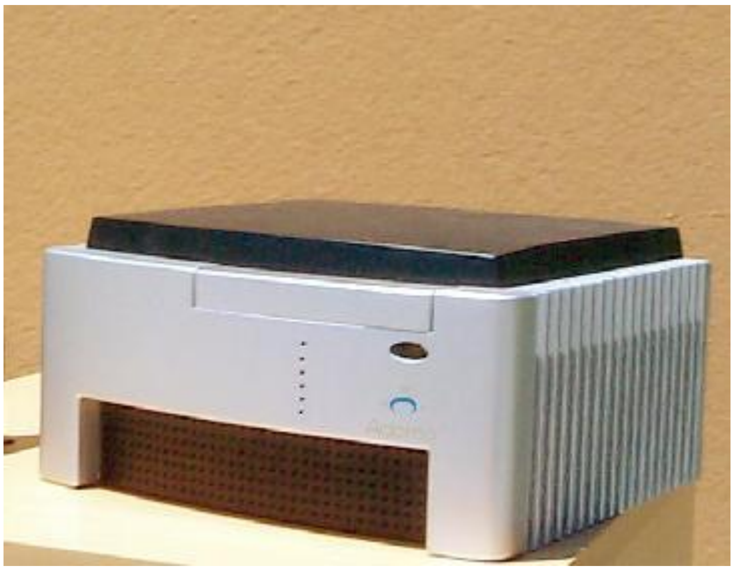


Figure 1. Adomo Server

Inside, there's a custom motherboard based on an AMD mobile K6 processor. Adomo's engineers took advantage of laptop-like components in designing the server, to reduce power consumption and heat dissipation. This lessens the need for noisy fans, and should boost system reliability.

The server is a multi-tasking, multi-user computer system that provides a variety of shared resources and managerial functions, including those of Internet gateway, firewall, e-mail, file storage and system backup. And don't forget: *all* application programs for all client devices run on the server, not on the devices.

Although the server isn't promoted as such, you can add a monitor and keyboard and use it as an additional desktop system. In that case, it will run all the same application programs available to the client devices.

The Wing

The first information appliance Adomo will deliver is a sleek unit called the “Wing” (see Figure 2).

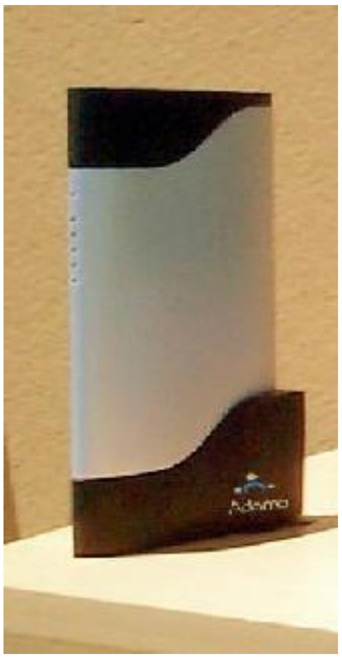


Figure 2. Adomo Wing

The Wing is a multi-function user-interface appliance that contains a 90MHz Motorola Coldfire system-on-chip processor and a VGA display controller, along with interfaces for infrared keyboard/mouse, PS/2 mouse, microphone and speakers. An internal PCMCIA socket provides the wireless LAN connection. The Wing has a small built-in speaker and microphone, and provides jacks for optional connection of external high-quality stereo speakers and microphone.

Since the Wing is simply a nice-looking but “dumb” terminal, it can be used for a variety of purposes. Hook it up to a monitor, keyboard and mouse, and it acts like a PC. Leave those off and connect it to high-quality stereo speakers, and it becomes a wireless music system that plays Internet radio, CDs and MP3s.

Making Use of Voice

One of Adomo's strategies to make their products “family-friendly” is extensive use of voice for both commands and data. What could be more natural for family-oriented systems? Accordingly, family members will be able to leave each other voice messages as easily as e-mail messages, and in much the same way.

Despite recent progress, however, decent-quality voice recognition still requires desktop-level computing power. This limits the voice-recognition capabilities of low-cost information appliances. But in the Adomo client/server architecture, the devices need only send 10KBps voice data to the server. There, high-end CPU resources are available to process it. The result: low-power, inexpensive appliances benefit from high-end speech recognition technology.

Benefits of Linux and Open Source

Why did Adomo choose to use Linux? “On the client side, scalability, modularity and availability of source, which allowed us to customize the OS to meet our exact requirements, were key reasons for using Linux,” says Adomo CEO Samir Lehaff. “On the server side, Linux offered us a highly stable multi-user OS, without the usual licensing and serialization headaches associated with trying to use Windows NT in an embedded product.”

The Adomo application required customizing a Linux server OS to make it suitable for the home environment and developing a set of hosted applications for home users. The server is based on a Red Hat Linux distribution, along with the Netscape Mozilla open-source browser. The applications are aimed at communications, web browsing, data manipulation and organization, backup and household task management. One such application is an MP3 music jukebox. The client devices run a diskless X-terminal version of uCLinux ported to a Motorola Coldfire microprocessor. This provides an economical and versatile platform for a wide range of future devices.

Adomo's embedded Linux-based development project benefited greatly from collaboration with the Open Source community. For example, Adomo had to grow uCLinux from its normal headless “network box” configuration into a full-fledged X-terminal implementation, while still maintaining a small memory footprint. For assistance with that challenge, Adomo turned to embedded Linux specialists at Moreton Bay (now a Lineo company).

Similarly, SuSE provided some important help in implementing a truly embeddable X server for the client devices. This took the form of Tiny-X, a reduced-footprint implementation of the X Window System that requires less than 800KB of memory.

Another challenge was running Linux on client devices that don't contain application program-load memory, beyond a small ROM used for system bootstrap. Instead, the client devices boot Linux remotely via wireless LAN. Help in overcoming this problem came from the Linux Terminal Server Project (LTSP) team.

How “Open” is It?

Will Adomo allow third-party developers or even users to modify how the system operates? “Absolutely!” says Adomo CEO Lehaff. He says the company will post its embedded Linux technology, as GPL, on its web site. The company will also encourage third-party developers to create both software applications and additional client devices.

Production Plans

Adomo plans to launch its products in October 2000, with production ramp-up in time for the 2000 holiday season. Lehaff describes the target market as "technology-friendly families who are faced with organizing and networking a growing number of Internet-connected devices in the home." In addition to the server and the Wing, Adomo plans to introduce two additional products early next year: AdomoPad, a battery-operated wireless touch-screen pad that integrates web browsing and telephone functions, will be used like a jumbo PDA; and AdomoTune, a device that will allow digital music received from the server or the Internet to be played over high-quality audio equipment.

email: rick@linuxdevices.com

Rick Lehrbaum (rick@linuxdevices.com) created the http://www.LinuxDevices.com/ "embedded Linux portal", which recently became part of the ZDNet Linux Resource Center. Rick has worked in the field of embedded systems since 1979. He co-founded Ampro Computers, founded the PC/104 Consortium, and was instrumental in launching the Embedded Linux Consortium.

Jarring Sects and Buttered Parsnips

Stan Kelly-Bootle

Issue #77, September 2000

Stan uses religious metaphors to discuss the Linux community.

The term “theologian” has found a niche in the computer lexicon [sic]. He or she represents, rather unfairly, the dogmatist ever-ready to promote MyGol over YourGol. They (to switch to the gender-free plural) are also wont to niggle and squiggle over *every* jot and tittle. Not just the jots and tittles in the source code are fair game (after all, that misplaced semicolon can cost rockets and lives), but also the fuzzier jots and tittles in a natural-language document such as press releases, pattern proposals, or ISO standards.

We should not be surprised, therefore, if fervently supported socio-computer “movements” betray many of the exegetical biblical excesses noted by historians of religion. (Not to be confused with religious historians.)

The unavoidable keywords are “schism” and “syncretism.” In particular, the OSS (Open Source Software) agendum, one with “community” written into and between every syllable, is now subject to inter-commune dissension. It takes only a few agendums to create an agenda. And Church Father, Eric Raymond, added to the religious dimension with his *Cathedral and the Bazaar* discourse.

This columnist was tickled by the diverse opinions reported in “We Talk to Everybody” (*Linux Journal*, June 2000). The disciples were typically opposed on vital fundamentals (the telltale sign).

Compare this, briefly, to St. Paul's efforts in fusing messianic Judaism and pagan Platonism to spread the gospel through a hostile Roman empire. Christianity endures as a major world “belief system” (yes, this must sound cold and offensive to believers) in spite of almost daily “cultic” splits that outscore the welcomed annual ecumenical reunions. (Walter Martin, the original “Bible Answer Man”, had a marvelous term for petty disputations: “majoring in the minors”.)

You may have guessed that I'm pondering impious parallels with the Linux gospel. Of course, analogies should never be pushed beyond luck or reason, but it's worth a try if lessons are to be learned. It took several Trent Councils to resolve the trinitarian bits, so let's be patient with the LSB (Linux Standard Base) canon. But not overly patient—I'm prepared to sizzle dumb opponents with a just auto-da-fe.

Martin Scorsese and I, about to film *Son of Last Temptations*, have no problem casting Richard Stallman as John the Baptizer and Linus Torvalds as St. Paul. We'll risk the odd anachronism (U.S. viewers will hardly notice) by having the Pauline epistolary admonishments sent by e-mail. We'll show some messages bouncing from corinthians.org and romans.mil. Mel Brooks has scripted a catchy letter to the Hebrews. Our studio historical research team tells us that the Galatians were unruly Celts, so we plan to exploit this fact for the Irish-American audience. FreeBSD will appear as the Essenes writing those "Dead-C" scrolls.

Lots of cast gaps remain for major and minor prophets, not to mention profits. I see Turing as Job, von Neumann as Abraham, Ritchie/Thompson as Ezekiel/Isaiah and, subject to legal approval, Gates must get the nod for Satan. Judas? All we can say at the moment is that he or she wears a red hat. I'm currently co-favorite with Stallone for the major messianic role. I have the edge in C++ syntax, but he just punched me in the gob, screaming "ggg-nooo".

Coming Soon (September 2000): the ultimate book on Linux Security from Prentice Hall PTR: Bob Toxen's Real World Linux Security—*Intrusion Prevention, Detection and Recovery*, including a CD-ROM of vital programs to reduce your "vulnerabilities".

Stan Kelly-Bootle (skb@atdial.net) has been computing on and off since his EDSAC I (Cambridge University, UK) days in the 1950s. He has commented on the unchanging DP scene in many columns ("More than the effin' Parthenon" —Meilir Page-Jones) and books, including *The Computer Contradictionary* (MIT Press) and *UNIX Complete* (Sybex).

TimeSys Linux/RT (Professional Edition)

Daniel Lazenby

Issue #77, September 2000

Linux/RT is a tool for developing and testing embedded and non-embedded real-time applications. TimeSys, the maker of Linux/RT, suggested several application domains where Linux/RT could be used.

- Manufacturer: TimeSys Corporation
- E-Mail: info@timesys.com
- URL: http://www.TimeSys.com/
- Price: $199 US
- Reviewer: Daniel Lazenby

Linux/RT is a tool for developing and testing embedded and non-embedded real-time applications. TimeSys, the maker of Linux/RT, suggested several application domains where Linux/RT could be used. Application domains suggested include industrial automation, process control, telecommunications, web servers, avionics and others.

According to the vendor, the architecture of Linux/RT offers Linux resilience and stability while running real-time applications. To make the architecture real, TimeSys added a few additional calls to their release of Linux and used a real-time resource kernel (RK). As a loadable kernel module (LKM), the real-time RK interacts with the Linux kernel. The Linux kernel interacts with the hardware. Using this architecture, the resource kernel can take advantage of all existing Linux hardware drivers. Existing applications continue to interact with the Linux kernel, while real-time applications interact with the real-time kernel module. With this approach, the whole system will not crash if a real-time application process hangs or crashes. It also means pre-existing (legacy) Linux applications can run along side the real-time application. The RK module may be loaded or unloaded like any other LKM.

Linux/RT is packaged into three editions: Standard, Deluxe and Professional. The Professional Edition comes with Linux/RT, TimeTrace, and according to the web site, an evaluation copy of TimeWiz. The review package did not contain TimeWiz. TimeSys Linux/RT comes with two real-time environments. Linux/RT is based upon the Carnegie Mellon University Linux resource kernel (Linux/RK). Support for Robust Embedded (RED) Linux system event logging is included in Linux/RT. For compatibility with pre-existing systems, TimeSys Linux/RT includes the Real-Time Applications Interface (RTAI). On a side note, only one of the two real-time environments may be loaded at any one time. Figure 1 shows one of the included Linux/RT real-time sample programs.

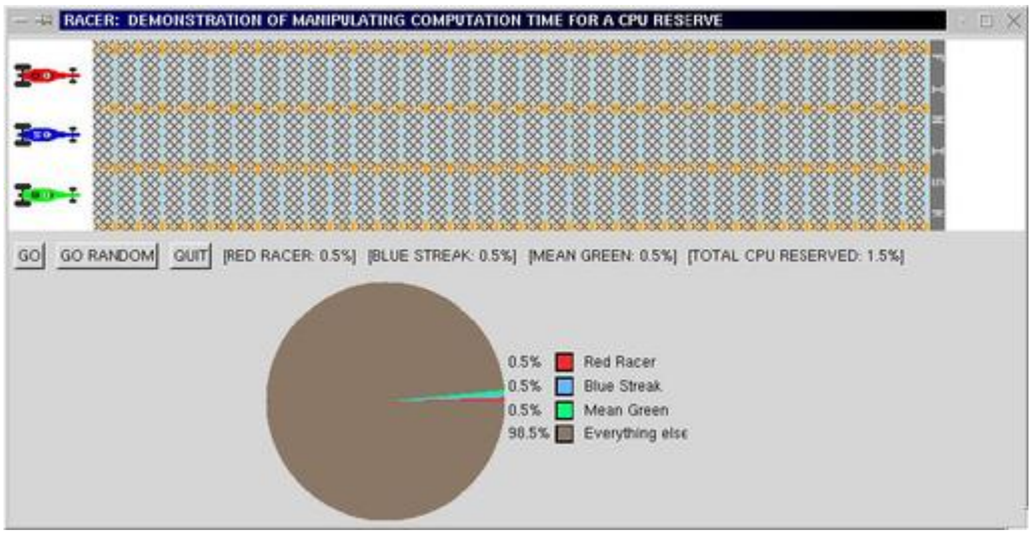
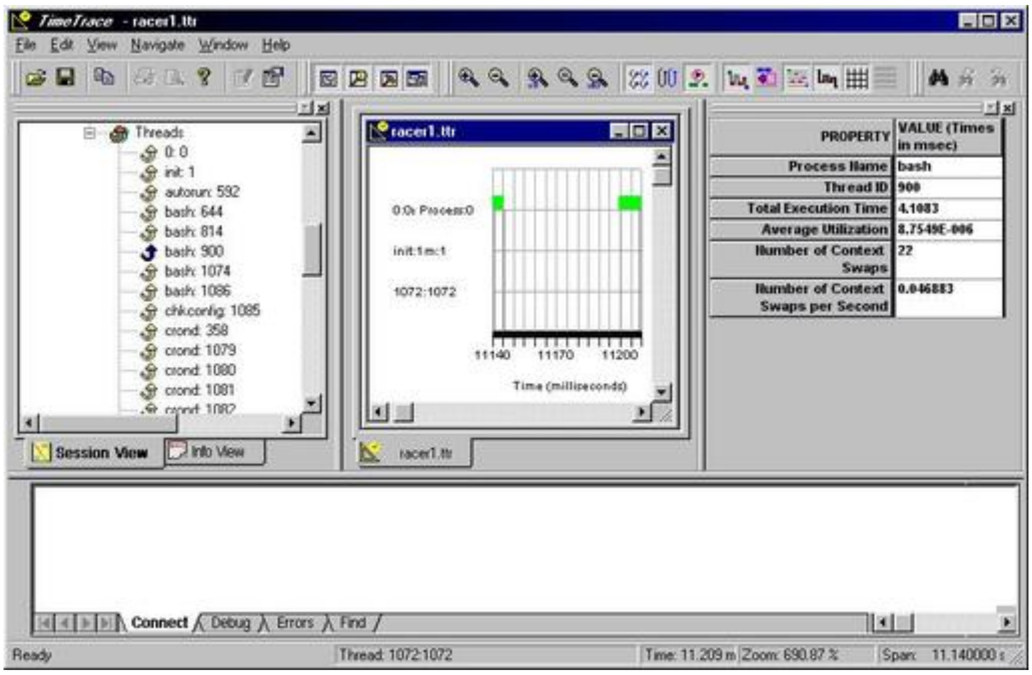


Figure 1. Linux/RT Real-Time Sample Program

According to the model, the resource kernel can reserve some portion of CPU cycles, network bandwidth, disk bandwidth and memory. These reserves are grouped into resource sets and assigned to a process. Version 1.0 of Linux/RT supports several capabilities. These include 256 fixed-priority scheduling levels, CPU resource reservations, high-resolution clocks and timers, periodic real-time tasks and pinning memory used by real-time processes. The review copy of Linux/RT and manual did not appear to support network or disk bandwidth reserves. According to the manual, Linux/RT utility commands may be used to assign a resource set to any Linux process. Another feature of Linux/RT is its ability to provide a quality of service (QoS). Using Linux/RT utility commands, real-time and legacy Linux programs can be endowed with specific QoS levels.

TimeTrace (Standard Edition) for Linux/RT provides the ability to visualize and profile one Linux/RT host running real-time programs or applications. An example of the TimeTrace interface is shown in Figure 2. TimeTrace sits on its own platform and provides a graphical user interface (GUI) presentation of the monitored host. With TimeTrace, you can view scheduling, context-switching, system calls and user events. According to the vendor, TimeTrace makes your real-time task's worst-case and average execution times, and period information, available. The Professional Edition of TimeTrace can monitor multiple Linux/RT hosts.



```
Threads
  0: 0
  init: 1
  autorun: 592
  bash: 644
  bash: 814
  bash: 900
  bash: 1074
  bash: 1086
  chkconfig: 1085
  crond: 358
  crond: 1079
  crond: 1080
  crond: 1081
  crond: 1082
```

PROPERTY	VALUE (Times in msec)
Process Name	bash
Thread ID	900
Total Execution Time	4.1083
Average Utilization	8.7549E-006
Number of Context Swaps	22
Number of Context Swaps per Second	0.046883

```
Ready    Thread: 1072:1072    Time: 11.209 m  Zoom: 690.87 %    Span: 11.140000 s
```

Figure 2. TimeTrace Interface

TimeWiz: Modeling, Analysis and Simulation tool provides the ability to observe the behavior of a real-time program visually. According to the vendor, this tool may be used to analyze worst-case behaviors, simulate average-case timing, model end-to-end performance, represent hardware and software configurations and generate reports. As with TimeTrace, TimeWiz sits on a Wintel platform.

How Hard Is It to Install?

Linux/RT is based on the Debian distribution. A standard Debian install process may be used for new Linux installations. The manual says Linux/RT may also be used with non-Debian distributions, including those of Red Hat, SuSE and Mandrake. I run Caldera's distribution on my computer and chose to use the script provided for installing Linux/RT on non-Debian distributions. The non-Debian Linux install script offered custom or default installs. The custom install offers options to select the target install directories and to register Linux/RT with LILO. I chose to use the default install. The install went well. Caldera 2.4 now uses GRUB as its boot loader. Linux/RT looks for LILO, and did not recognize GRUB. I added the Linux/RT kernel to my GRUB boot list and rebooted. On boot-up, the default TimeSys kernel recognized my SCSI card, SCSI devices and my Ethernet card.

TimeTrace for Linux/RT installation consists of two installation activities. First, TimeSys Linux/RT kernel modules must be installed on the Linux host. Second, you must instal TimeTrace on a Windows NT/95 host. Next, two user-mode collector files (viewrk and viewrtai) installed on the Windows host need to be copied to the Linux/RT host(s).

TimeTrace uses traditional Windows installation methods. The installation went smoothly once the CD was recognized by a CD-ROM drive. I am not sure why my normal CD-ROM would hang with the TimeTrace CD loaded. I use the same CD-ROM drive for all my CD install media. On a lark, I tried my DVD drive. The install CD was recognized immediately. The TimeTrace manual did not suggest any particular Linux/RT file system location for the two collector files that needed to be copied to the Linux platform. I noticed several other RK utility files were placed in the /usr/local/bin directory. So, it seemed like a natural place to put these two files.

How Hard Is It to Use?

Using Linux/RT kernel modules, RTAI or TimeTrace involves issuing a couple of commands. For Linux/RT, it is a matter of loading or unloading some Linux kernel modules. Using TimeTrace presumes a real-time application is already running on the Linux box and the proper user-mode collector program is running as well.

Using the Linux/RK Utilities commands, a Linux/RT resource may be attached to legacy program processes. The legacy process does not have to include any real-time programming constructs. I tried an experiment with my Netscape browser. I attached a resource set to the browser (about 5% of CPU), and then started a large file transfer using FTP and printing a large PDF document. My browser seemed to do okay.

Support and Manuals

Three manuals and a booklet came with the review copy of the product: the TimeSys Linux/RT User's Manual, the TimeTrace for TimeSys Linux/RT User's Manual and a TimeSys Linux/RT Programmer's Manual. The booklet titled “The Concise Handbook of Real-Time Systems” is like a Cliff Notes of real-time terms, concepts and architectures. TimeTrace and Linux/RT User Guides clearly presented the relationship of the two products. The user guides and booklet are illustrated, and for the most part they are well-written. All install directions are high-level and written for the experienced software installer.

About half of each Linux/RT and TimeTrace User Guide is spent discussing real-time software and its attributes. The remaining half of the guide focuses on using the product. Linux/RT's user guide and Release Notes describe how to use a graphical Linux/RT resource manager. Finding the command to start the graphical resource manager was a challenge. In the review copy, it was not compiled and nowhere near the Release Notes said the command lived. I do not think this is the result of installing Linux/RT on a non-Debian distribution. In another place, the Linux/RT User's Manual said the module measure-rk had to be loaded to use TimeTrace. No compiled version of the module measure-rk could be located in my installation. I did locate some source code that looked like it might be the module. The TimeTrace manual indicated one used viewrk file to use TimeTrace. Since there was no module named measure-rk available, I used the viewrk file. Viewrk seem to work okay.

Appendix A contains a list of Linux/RT RK utility commands. Several of these have the wrong capitalization. Others do not exist as listed. I was unable to locate several of the commands in /usr/local/bin directory in the manuals. One example is printing the clock frequency. The list of utilities says the command **clockfreq** is used. The command did not exist in my installation. I did find a **rkfreq** command that provided the clock frequency. There is also a file named charter. I was unable to locate any information on this executable file. No man pages were installed by the product. The PDF versions of the User and Programmer Manuals seem to be just an electronic copy of the printed manuals. A keyword search of the PDF files did not produce any information on the abovementioned files.

The 179-page Programmer's Manual discusses how to create, execute and debug real-time programs. Example program listings are provided to illustrate the various functions and system calls. GNU-Emacs, make, CVS, gcc and gdb may be used to develop and debug Linux/RT-based real-time programs. Compiling or recompiling any of the distributed sample code requires linking to librk.a. Of the three references to this file's location, the Programmer's Manual was the most accurate. Be prepared to edit makefile directories to correctly identify the proper location of the library and header files. Due to missing files, I had some problems compiling the sample code. I think compiling original work should go just fine.

Support is offered via the TimeSys web site. Questions may also be e-mailed and faxed to TimeSys Support. I did not use TimeSys fax support, and cannot comment on its quality. Most of the questions in the web site FAQ struck me as very marketing-oriented. Searchable knowledge bases or other forms of technical on-line support were not readily obvious to me. There was the beginning of a discussion forum. Based on the few postings (at the time of this review), I am not sure of the usefulness of this forum.

I had some problems getting TimeTrace to display a graph of the racer and rolling sample programs, so I sent a support request e-mail. TimeSys seemed to be going through some growing pains, and it took several days to get a response. Once e-mail contact was made, TimeSys support staff worked with me on resolving the problem.

Good/Bad

Overall, I think Linux/RT software looks promising and should be able to do the job. The documentation needs some cleaning up and to more accurately reflect file placement and command usage. At some point, I believe Linux/RT will require fixes and patches. The web site should be able to provide fixes and patches without requiring one to download the 12+MB Linux/RT tar file. If support depends upon registering the product, then there needs to be some simple way for a customer to indicate they bought the product.

Hardware Used in Review

This software was reviewed using the following hardware: a 300MHz Pentium II Wintel machine with 64MB memory and a high-resolution monitor (1024x768), running Win98SE. The Linux machine is a 333MHz Celeron with 128MB memory and a high-resolution monitor (1024x768), running Caldera's eDesktop2.4 distribution of Linux.

Daniel Lazenby (d.lazenby@att.net) first encountered UNIX in 1983 and discovered Linux in 1994.

Archive Index Issue Table of Contents

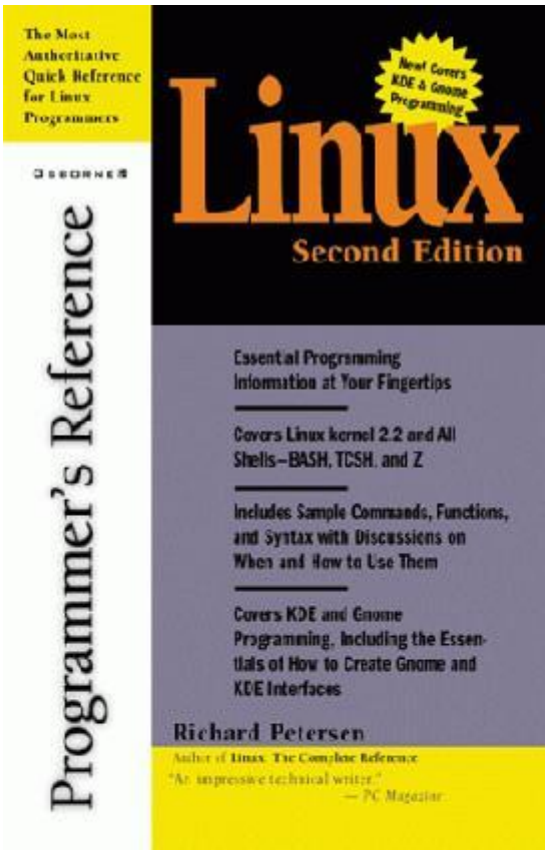
Advanced search

Linux Programmer's Reference, Second Edition

Ibrahim F. Haddad

Issue #77, September 2000

This book covers the configuration, initialization, creation and execution of shell scripts, as well as a description of all programming-related commands for the respective shell.



The Most Authoritative Quick Reference for Linux Programmers

OSBORNE

Programmer's Reference

New! Covers KDE & Gnome Programming

Linux

Second Edition

Essential Programming Information at Your Fingertips

Covers Linux kernel 2.2 and All Shells—BASH, TCSH, and Z

Includes Sample Commands, Functions, and Syntax with Discussions on When and How to Use Them

Covers KDE and Gnome Programming, Including the Essentials of How to Create Gnome and KDE Interfaces

Richard Petersen

Author of Linux: The Complete Reference

"An impressive technical writer."
— PC Magazine

- Author: Richard Petersen
- Publisher: Osborne McGraw-Hill
- E-mail: customer_service@mcgraw-hill.com
- URL: http://www.osborne.com/
- Price: $19.99 US
- ISBN: 0-07-212355-9

- Reviewer: Ibrahim F. Haddad

Every once in a while—and more frequently lately, due to the growing popularity of Linux—a book appears pretending to be a reference, but the material is neither well-organized nor complete enough to truly be called a reference for the subject. *Linux Programmer's Reference* does *not* fall into either category.

Linux Programmer's Reference is divided into three parts, with seven chapters and three appendices. The first part, Chapters 1, 2 and 3, involves the BASH, TCSH and Z shells. It covers the configuration, initialization, creation and execution of shell scripts, as well as a description of all programming-related commands for the respective shell. In addition, syntax, usage and short practical examples are given to make sure the reader understands the idea.

The second part, Chapters 4 and 5, concerns the compiling, debugging and packaging of C and C++ programs. Peterson covers everything from binary formats, static vs. shared and dynamic libraries, the **gdb** debugger, passing by the **make** utility, the revision control system (**rcs**), up to the man pages.

The author recognizes GNOME and KDE programming popularity by including a chapter on each (Chapters 6 and 7) describing the essentials of how to create GNOME and KDE interfaces.

Finally, the three appendices serve as a quick reference for Perl, the Tcl/Tk scripting language, and TeX and LaTeX commands.

The Perl appendix covers file commands, array operations, operators, control structures, functions, pattern-matching operations and regular expressions and subroutine components. The Tcl/Tk appendix lists Tcl operators, the most common Tcl and Tk commands, standard Tk widgets and Tk options. The TeX and LaTeX appendix covers the essential commands of TeX and LaTeX.

As a Linux programming reference, the book has much to offer. It presents valuable, precise programming syntax and advice for every Linux programmer, whether you are a novice, intermediate or expert programmer. In addition to syntax, it provides brief explanations and programming examples for each command, as well as advice on how, when and why to use each command, helping you choose the most suitable one for your particular task.

The examples throughout the book are very structured, short enough to type and try yourself, and still contain all the major concepts intended to be present. In general, the use of examples help to clear up questions the readers may have after reading the concept materials, and Peterson's examples serve that purpose well.

Petersen's writing style is clear and concise, making the book easy to read and follow. However, someone on one of the newsgroups complained that Petersen covers **ci** and **co** commands for RCS without ever noting that they stand for “check-in” and “check-out”, a concept that surely makes rcs easier to understand. Such small notions may seem not important for professionals, but they are of great significance to newcomers—a matter to which Peterson might have paid more attention.

Linux Programmer's Reference is not a “Learn-*X*-in-12-hours” type of book. This book is not meant to teach C, C++ or Perl. It is meant to serve as a reference for those moments when you have a programming-related question, such as when you've forgotten the syntax or wonder whether a feature exists or not. This book will give all you need to get a clear and concise answer without having to search countless pages in different books or seek the help of your colleagues.

The book, as a reference, can find its place on the bookshelves of Unix-experienced users and programmers who are migrating to Linux and want specific Linux information, especially on critical **gcc** compiler and library issues. Linux programmers and newcomers will also find it very useful for its list of shell commands, its programming section and the quick references.

Personally, I found this book to be a useful reference worth the price ($16.99 US at Amazon.com). On the other hand, I felt the author assumed the reader knows his way around Linux and its programming tools. So, if you are a fresh, out-of-the-box newcomer, expect to dirty your hands.

I received a number of comments from some reviewers wishing there was more on scripting languages than the Perl and Tcl/Tk listings in the appendices. I agree with them. If a solid chapter on scripting languages was included instead, it would have been a great book. Nevertheless, it's still a good book, well-written and concise, and it covers the ground. If you need a good reference, *Linux Programmer's Reference* is the book to buy.

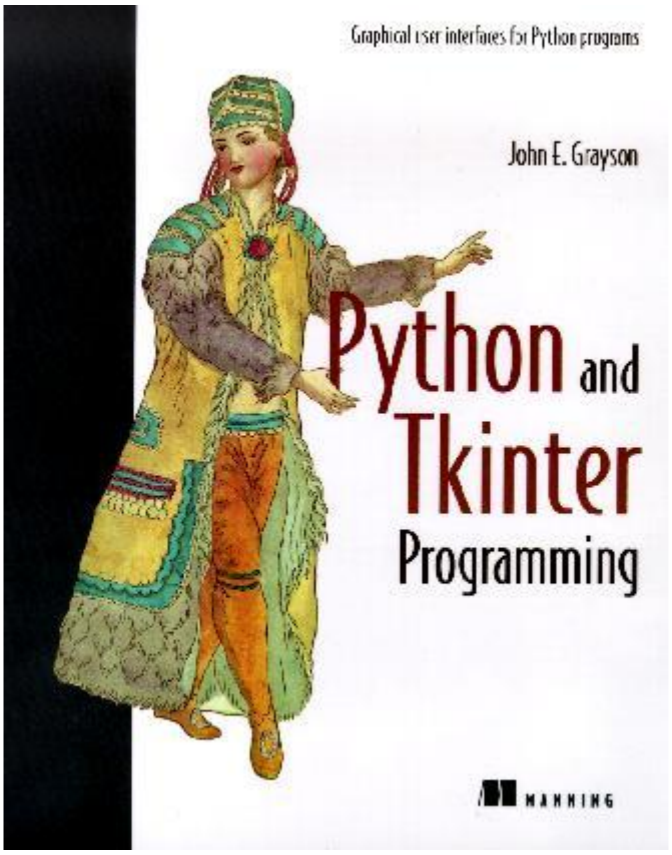
Ibrahim F. Haddad (ibrahim.haddad@lmc.ericsson.se) is a Senior Member of the Technical Staff at Ericsson Research Canada based in Montréal. Ibrahim is currently a D.Sc. Candidate in Computer Science at Concordia University. His research interests include distributed-object technologies, web servers, e-commerce platforms and Linux clusters.

Python and Tkinter Programming

Phil Hughes

Issue #77, September 2000

If you intend to work with Tkinter, buy this book.


Graphical user interfaces for Python programs
John E. Grayson
Python and Tkinter Programming
MANNING

- Author: John E. Grayson
- Publisher: Manning
- Price: $49.95 US
- Pages: 660
- ISBN: 0-201-63448-1
- Reviewer: Phil Hughes

Let's start with the conclusion: if you intend to work with Tkinter, buy this book. If you need to do graphical programming in Python but don't know what tools to use, buy this book. Now that the hard part of the review is done, let me give you a little background on my Tk experiences, and then get on to the details of the book.

About five years ago, armed with John Osterhout's book, *Tcl and the TK Toolkit* I started to play with Tcl and Tk. While I appreciated how you could toss together something very quickly, I was also very frustrated by Tcl. It just didn't feel like a real programming language. There was always the need for a “trick” to accomplish what I wanted. Because of this bad feeling, I never did much with Tcl and Tk.

Last year, when I interviewed Guido van Rossum, I asked him about toolkits for doing graphical programming in Python, and he recommended Tkinter. He also pointed out that the current problem was a lack of good documentation. Python and Tkinter has addressed that problem, and done a fine job as well.

The book assumes the reader has a basic knowledge of Python, but requires no knowledge of Tcl or Tk. The first part, only 28 pages, is a quick review of Python, an intro to Tkinter and a sample application. Thus, by page 28, you will be convinced that you, too, could write a calculator or two. Even in this introductory chapter, the author does an excellent job of discussing the various sections of the code.

The next part, titled “Displays”, goes on for close to 300 pages. This is the meat of the book, covering all the things you will need to build your application. First, widgets are covered, including Pmw (Python MegaWidgets) and then geometry managers. By the end of these two chapters, you should be pretty comfortable putting up any sort of widget you want and getting the screen organized.

Each concept is presented with some explanatory text, some code and a screen shot. The code is on track for teaching you the next thing you need to know, rather than being just a book of sample code. Part of the credit goes to Python itself, where you can write something useful in a few lines, but the author has done a good job of breaking up the material so that a concept can be presented in only a few pages.

The next chapter, titled “Using Classes, Composites and Special Widgets”, gets you seriously using structured programming techniques to develop your application. While this is a little scary for those of us who grew up on FORTRAN and moved to C, it is well worth embracing if you want to be a good Python programmer.

The next four chapters take you through drawing all sorts of objects. First there are “Dialogs and Forms”, where you finish up with a really fancy calculator. Next is “panels and Machines”, where you learn how to do very complex graphics models. A digital multimeter complete with rotary selector switch is one such example. The next two chapters cover more free-form drawing and finally charts and graphs. The part ends with a short chapter on the window manager.

Part three, titled “Putting it All Together...”, is a catch-all for the other things you need to know. This includes extending Python, debugging and issues of performance and style. The part ends with a chapter on distributing Tkinter applications.

The remainder of the book, over 250 pages, consists of appendices. This seems to be the tradition for most Python books. It makes sense, due to the lack of printed reference documentation on Python.

The first appendix is called “Mapping Tk to Tkinter”. If you have an existing application in Tk that you want to port, this is where you will find all the answers. It can also provide some help if you are familiar with Osterhout's book and want a quick translation guide.

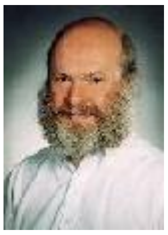
Appendix B is a Tkinter reference. The reference includes descriptions of the classes, along with detailed descriptions of all options. When appropriate, graphics are also included.

Appendix C is a reference on Python megawidgets. Like the previous appendix, the descriptions are detailed and graphics are included when needed.

Appendix D covers building and installing Python and Tkinter. Information is included for UNIX and MS-Windows, with a brief mention of MacOS.

The appendices end with three short sections: Events and keysyms, Cursors and References. While each appendix is very brief, it covers important topics.

The book ends with a comprehensive index. As I said in the beginning, if you are going to work with Tkinter, buy this book. I'm not saying this because it is the only book out there. It is well organized, easy to read and offers a great tutorial combined with a comprehensive reference.



Phil Hughes is the publisher of *Linux Journal*. He can be reached via e-mail at info@linuxjournal.com.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

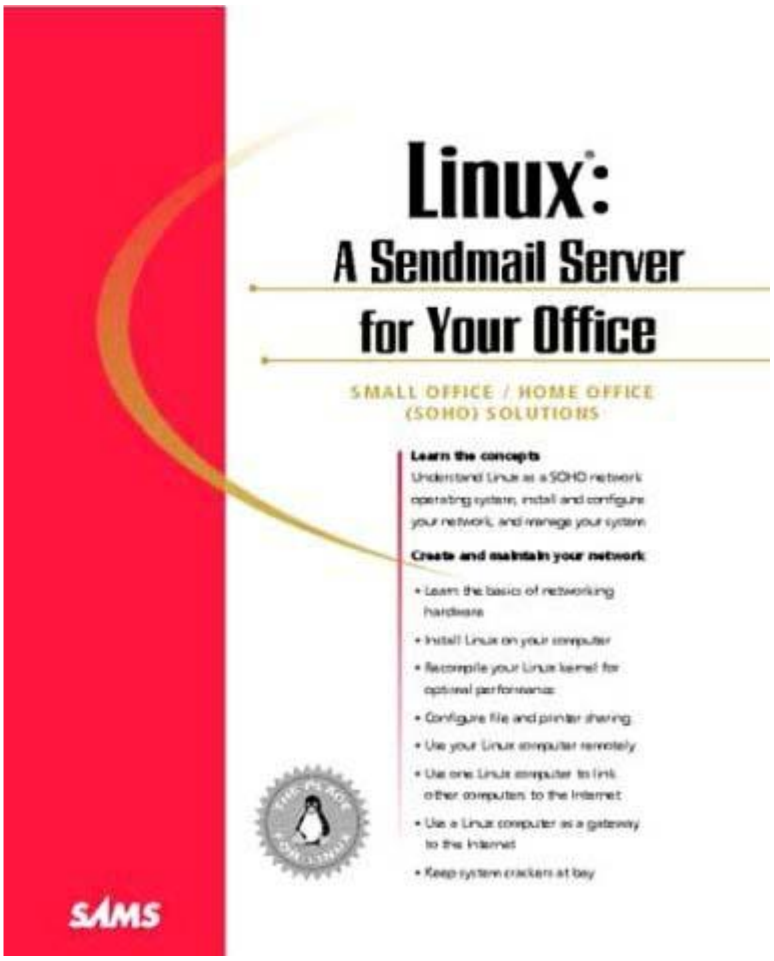
Advanced search

sendmail for Linux

Russell J.T. Dyer

Issue #77, September 2000

The answer to my prayers! Not that it offers any quick fixes, but it explains fairly clearly how to set up an e-mail server.



- Author: Richard Blum
- Publisher: SAMS
- URL: http://www.mcp.com/sams/
- Price: $34.95 US

- ISBN: 0-672-31834-2
- Reviewer: Russell J. T. Dyer

One of the most basic features of the Internet is e-mail. There's nothing complicated about it, right? Well, try configuring sendmail on a Linux server. Recently, I needed to set up a Linux server at work for e-mail service. It seemed like a simple enough task, but it quickly proved to be quite frustrating. It's one of those projects that requires paging through endless scripts, man pages and books. Don't bother asking anyone for help—you're on your own. Being a book lover, though, I saw it as an opportunity to visit the bookstore!

A very comprehensive book about sendmail was published by O'Reilly. I just love O'Reilly's books—usually. Their *sendmail* book is a killer: about a thousand pages long, very detailed and very advanced. I quietly put that back on the shelf after a couple minutes of looking at it (although I did go back to it three times, hoping I was too hasty). It took quite a bit of rummaging to find anything covering sendmail. Most authors touch on it only briefly—they know to stay away from it. I did buy one book that was fairly helpful in learning how to set up various Internet services, including e-mail: *Red Hat Linux 6 Server* by Mohammed J. Kabir (M&T Books, 1999). However, it wasn't quite sufficient. Besides, no good Linux task is solved with only one book purchase.

After many failed attempts at configuring sendmail and three visits to the bookstore (I have an undying belief that all my problems can be solved with books), I discovered Richard Blum's new title, *sendmail for Linux*. The answer to my prayers! Not that it offers any quick fixes, but it explains fairly clearly how to set up an e-mail server. It's about 500 pages of useful information with plenty of diagrams, screenshots and brief (let me say that again: *brief*) configuration examples. I hate it when books are loaded with long scripts that are already on my computer, just so I can admire them in a printed and bound format. There's also the customary CD. This one includes a distribution of Linux, qmail, Eudora Light and a few other related programs.

Blum's book is broken up into three sections: an introduction to e-mail services, installation instructions and sundry items. The introduction is an in-depth educational primer on e-mail. At first I skipped this part, but as I dug into the second section, I realized I didn't know as much as I thought about e-mail, even after something like a decade or so of using it. So, I went back to basics, and read through the first section. It was worth the effort. In addition to a minimal amount of history, Blum covers DNS, SMTP, IMAP, POP and sendmail itself. You see, it's not just sendmail, but everything related that makes it so involved.

With the foundation of the first section, the second describes basic configurations to make sendmail work. Blum explains what's necessary to get e-mail running for typical scenarios. The second section also covers setting up clients in Netscape, Outlook Express and Eudora; integrating e-mail service with an ISP; and the system administration involved.

The third section contains instructions on integrating an e-mail server with a firewall (e.g., ipchains and ipforward). It also includes setting up dial-in clients, e-mail aliases, masquerading and list servers. Yes, you too can be a major domo.

Let me wrap up now with something that Blum's publisher might want to quote: it's a comprehensive, but not overwhelming, manual on sendmail for the intermediate-level Linux system administrator. It's well-written, well-illustrated and well-edited; an educational, extremely useful and desperately needed resource on e-mail service.

Russell Dyer (russell@dyerhouse.com) is an IS manager for a national firm. He's married with three children and is a graduate student in English at the University of New Orleans. He likes to install Linux wherever he can sneak it in: resistance isn't futile, if it's subtle.

Letters

Various

Issue #77, September 2000

Readers sound off.

Keep up the Technical Stuff

Congratulations on a very good issue. As someone who first encountered UNIX via Coherent and then later turned to Linux, I want to let you know that *Linux Journal* provided me with many interesting articles on new and useful open-source products.

As the Linux user base continues to grow, *LJ* is faced with addressing the needs among the growing ranks of newbies as well as those who have followed Linux for several years now. Hopefully, *LJ* will strike a good balance. I for one thoroughly enjoy articles on “nuts and bolts” items, such as kernel internals, libraries and shared module implementation.

Also, case studies are great. At work, we are very interested in using Linux for noncritical “ancillary” applications. These case studies help us learn from and avoid any mistakes others may have made.

Keep up the good work, and don't shy from the in-depth stuff.

PS: The “centerfold” in the June issue was more than a little tongue-in-cheek, and although I found it somewhat funny, it seemed out of place in a quasi-technical magazine. —Allan Peda apeda@interpublic.com

Disgusted

Today I'm ashamed of being a *Linux Journal* subscriber. I am disappointed with *LJ* and SSC almost beyond all reason. Why is the *Linux Journal* eCommerce site (store.linuxjournal.com) running on NT and IIS? Don't you people have a fundamental problem with that? How could you betray the Linux community in such a manner?

Is there some extenuating circumstance? Is this some sort of mistake? How little faith do you have in Linux that you would do this thing?

These are not rhetorical questions. If I don't get a satisfactory answer, my subscription is history. I read *LJ* front to back every month, and I would miss it terribly, but I will not support you if your politics are wrong: in a capitalist society, a dollar spent is a vote cast. Why are you voting for Microsoft? —Dale Lakes dale@multiverse.com

In order to serve our store customers better, we wanted to outsource the web presence and fulfillment to the same company. WAS offers this service, and plans to port their software to AIX. This was the best alternative we could find. If you know someone running Linux who offers the needed services, let us know —Editor

Problem with Dynamic Class Loading

I read the article “Dynamic Class Loading for C++” in the May issue of *Linux Journal*. I've been experimenting with this technique awhile, and I experienced a problem that is not mentioned. I had some name clashes between libraries: if two libraries have an internal class, function or public symbol with the same name in both, once they are loaded, the executable behavior is unpredictable.

It seems that when a library is linked in, the dynamic linker starts searching all libraries for symbols, even for the libraries' internal ones. I do not know if this problem was solved in recent versions of the dynamic linker, or if there is a linker option I missed.

I solved it by putting all internal symbols in name spaces. Making sure there were no name spaces with the same name in two libraries (unless desired) eliminated the odd behavior. —Dario Mariani dmarian@fi.uba.ar

Not Quite Everybody

I just read the “We Talk to Everybody” article in the June issue and found it very instructive, since as a “late” Linux user, I ashamedly confess I didn't know all the names listed in it.

However, I think two more people deserve to have their names written on the “Linux Hall of Fame”: Paul McKerras and David Hinds (hope I didn't misspell their names) for PPP and PCMCIA. Without them, I wouldn't be able to send you this e-mail (I run Linux on a notebook).

I know we have to do a sort of lightweight patching on the kernel sources for these to work, but I am really thankful. —Nguyen Tuan nguyen_a@cnam.fr

Russian Fonts and Office Suites

I have wanted to write about this issue for quite some time now. Everybody is lamenting a lot lately about the “necessity” of having Microsoft Office for Linux as the definitive way for Linux to succeed on the desktop. I happen to have some reservations about that.

First of all, I am lucky enough to work at a UNIX-centric office, so having Linux on my workstation is a given. Aside from my primary job functions, I use Office (StarOffice, that is) for office-related stuff. I use word processors and spreadsheets equally. I used to have Windows on a separate partition as a dual-boot machine, but it's gone now. I needed the space for something else.

The office suite problem still exists for me, though. Being bilingual, I often need to use Russian fonts/encodings on a variety of machines, mostly Linux (x86 and PPC) and Macintosh. Well, it also has to be readable on Windows. MS Word has its own most weird (incompatible) way of encoding (I'd say scrambling) Russian characters, rendering them unreadable on any other hardware/software configuration. So it's unusable for me, period. StarOffice is actually very good in creating nice portable bilingual documents that can be imported into practically any other application. Bummer there's no StarOffice for Macintosh! Corel WordPerfect has the same limitations as StarOffice, but it is even further restricted regarding the OS (no Solaris version). The only “word processor” that is actually available for any platform/OS and works with any national character set is Netscape Communicator. The output of this application beats all competition in terms of universality and portability, but it does suck as a word processor.

Bottom line: I need an office suite that is reasonably full-featured (like StarOffice), available for all major and minor OS platforms (like Netscape), produces absolutely portable text output (again, like Netscape) and doesn't cost an arm and a leg (like Netscape and StarOffice). I want to reiterate that incompatible features of MS Office make it unacceptable for me even for free. Does anybody use a non-stolen version of MS Office anyway? —Alex Aitouganov Alex.Aitouganov@Eclipsys.com

Artists' Guide to the Linux Desktop

I have enjoyed your guide to the Linux desktop series immensely, and I was delighted to see that you wrote about my favorite window manager, Window Maker. Of course, I wasn't too terribly happy with your lack of praise for my darling, but still, I don't want to start a holy/flame war. I merely wanted to inform you of one thing. —Robert Wade robertwade@rocketmail.com

I'm glad you liked the series. There is one more coming, which should be out in the next couple of months, that does basic coverage of the remainder of the window managers out there. —Michael J. Hammel mjhammel@graphics-muse.org

Exaggerations

I know that exaggerating about how long one has been using Linux is a long-standing tradition for some people in the community, but it's getting a little out of hand.

In the June 2000 issue on page 98, the article claims that H. J. Lu discovered Linux in 1990. That's simply amazing, since Linus announced the project on July 3, 1991. (www.li.org/li/linuxhistory.shtml)

Perhaps Mr. Lu has the capability of seeing into the future? —Dave Whitinger dave@whitinger.net

Security Implications

I neglected to mention an important point in my article on syslogd in the July issue of *Linux Journal*: If you plan to use the network features of syslogd, it is very important that this port be behind a strong and properly configured firewall router. If not, the feature should not be used. The network syslogd interface is a great way to do remote denial of service attacks, and even some far more insidious attacks. The denial of service can be to both CPU and disk, since sending tons of spurious messages can make syslogd fill a disk drive. Moreover, such messages may be used to obscure other evil activities which you might lose in a flood of phony messages.

Even more frightening is the possibility that you are watching some log files with, say, poorly written Perl scripts that interpolate variables containing strings from syslog-generated log files. There are some clever Perl hacks where you can compose strings that will execute programs as the user running the script. Such strings could be embedded in fake messages sent to your syslog d<\#230;>mon.

The upshot of all this is that while collecting logs from multiple machines over a network is a very nice and useful feature, you must be sure to prevent machines outside your network from reaching this port.

This is an important point, and I hope you can find room for my letter! Thanks a lot!

Here is the corrected Listing 1 to include with my other comment.

Due to an accident of typesetting, the syslog.conf listing in the July 2000 issue was incorrect. The correct listing appears below.

Listing 1

—Michael A. Schwarz mschwarz@sherbtel.net

upFRONT

Various

Issue #77, September 2000

They Said It and more.

THEY SAID IT

“There is none. Get over it.” —Scott McNealy on privacy

“At this stage in my life, the thing that really turns me on is competence.”

—Billy Joel

“Science would be superfluous if the outward appearance and the essence of things directly coincided.”

—Karl Marx

“Certum est, quia impossibile. (It is certain, because it is impossible.)”

—Tertullianus

“What the hell is content? Nobody buys content. Real people pay money for music because it means something to them. Being a “content provider” is prostitution work that devalues our art and doesn't satisfy our spirits. Artistic expression has to be provocative. The problem with artists and the Internet: Once their art is reduced to content, they may never have the opportunity to retrieve their souls.''

—Courtney Love

“Today I want to talk about piracy and music. What is piracy? Piracy is the act of stealing an artist's work without any intention of paying for it. I'm not talking about Napster-type software. I'm talking about major label recording contracts.”

—Courtney Love

“Linux means never having to delete your love mail.”

—Don Marti

“Maturity is when you quit blaming other people for your problems.”

—Craig Burton

“Who is General Failure, and why is he reading my hard drive?”

—Bob (on Slashdot)

“I'm still freaked out by all this. I just wrote a (bleeping) anthropology paper.”

—Eric Raymond

“My hovercraft is full of eels.”

—Monty Python

Google Gains While FAST Keeps Pace

These days, search engines are praised for their educated guesswork. Each pile of results is presented with an implication: “Here's what we think you want.”

But most serious researchers (i.e., Yours Truly and all *Linux Journal* readers) often don't want an engine to guess. These users want to search for specific strings—or, as some search engines put it, phrases. These include names, text passages, lines of code, diseases and all other series of words.

On December 3, 1999 and June 27, 2000, I tested fifteen of the most familiar search engines by searching for a relatively unique phrase: “He that by me spreads a wider breast than my own”, which is part of a familiar line from Walt Whitman's *Song of Myself*. The phrase occurs far from the beginning of the work and can be found in many documents on the Web, including one on my own site, http://www.searls.com/.

It's a tough test. Only those engines that deeply search an enormous range of sites will yield results. The word “breast” is also a common and controversial word, which might invite spurious results.

Testing for strings also isn't easy on the tester, since search engines have different ways of recognizing phrases. Most require quotes. Others (Hotbot, FAST) have pop-out menu commands. One (Yahoo!) requires clicking a radio button in “advanced” search mode. Others (mostly at the bottom in the surveys) don't search phrases at all.

I run this test quite often for my own purposes and rarely record the results. But last December I did record them, and I repeated the test again on June 27—nearly seven months later. This time I added two more tests: one for an obscure blood disorder and the other for a Linux system call. Here are the results:

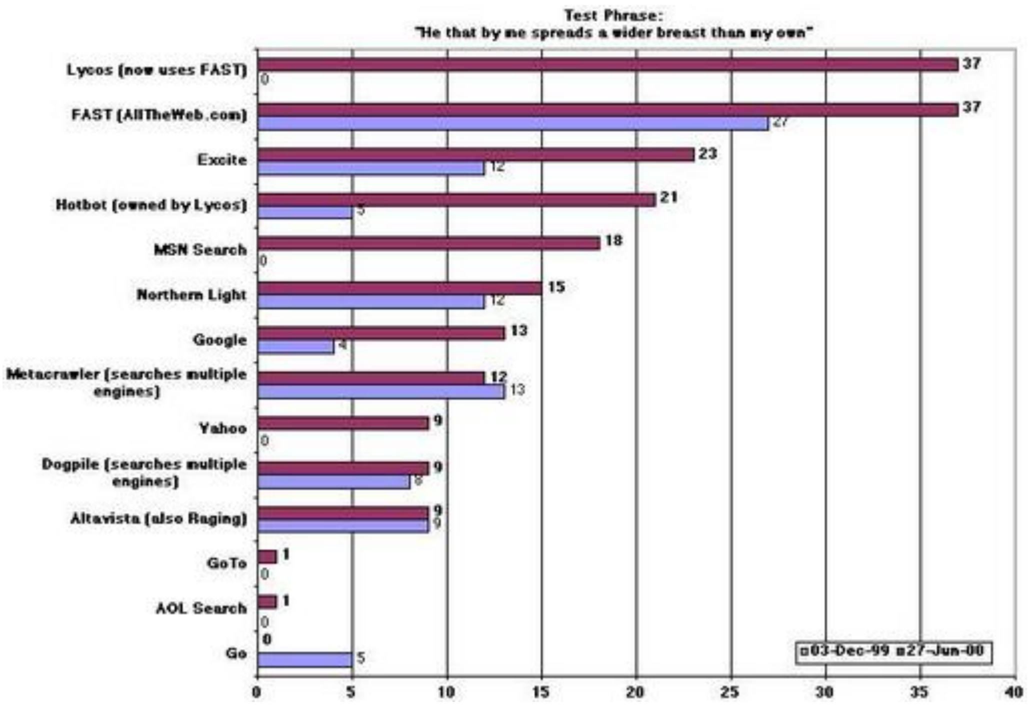


Figure 1. Test Phrase I

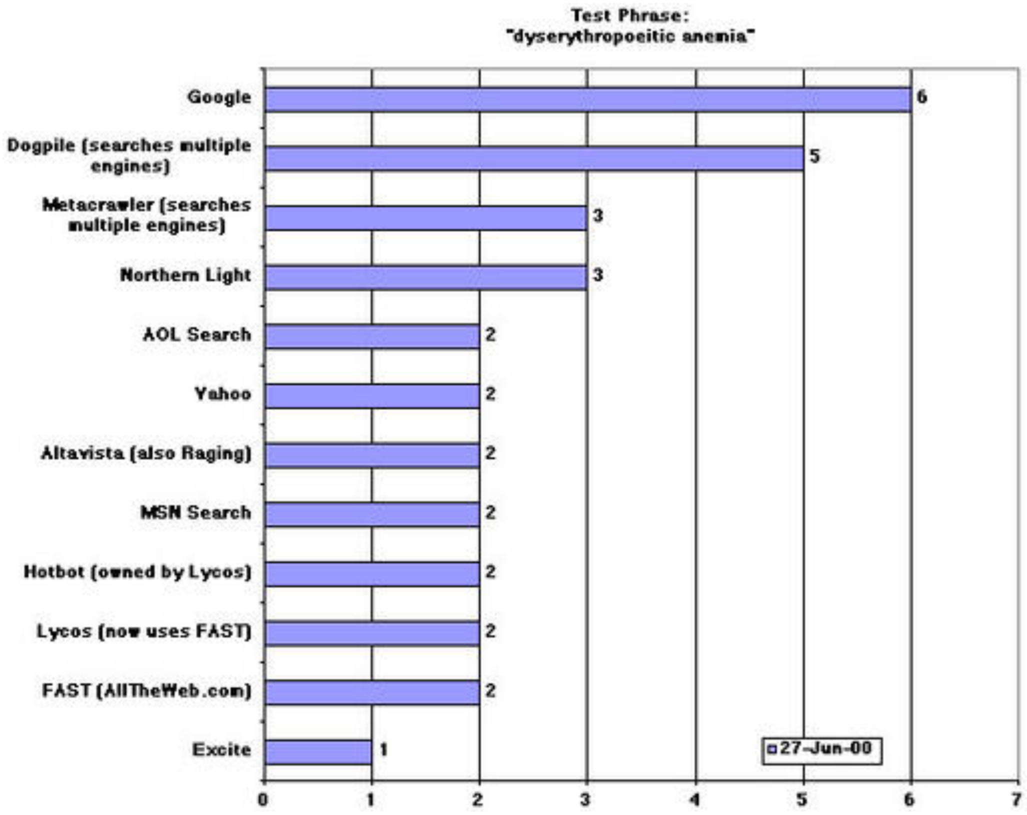


Figure 2. Test Phrase II

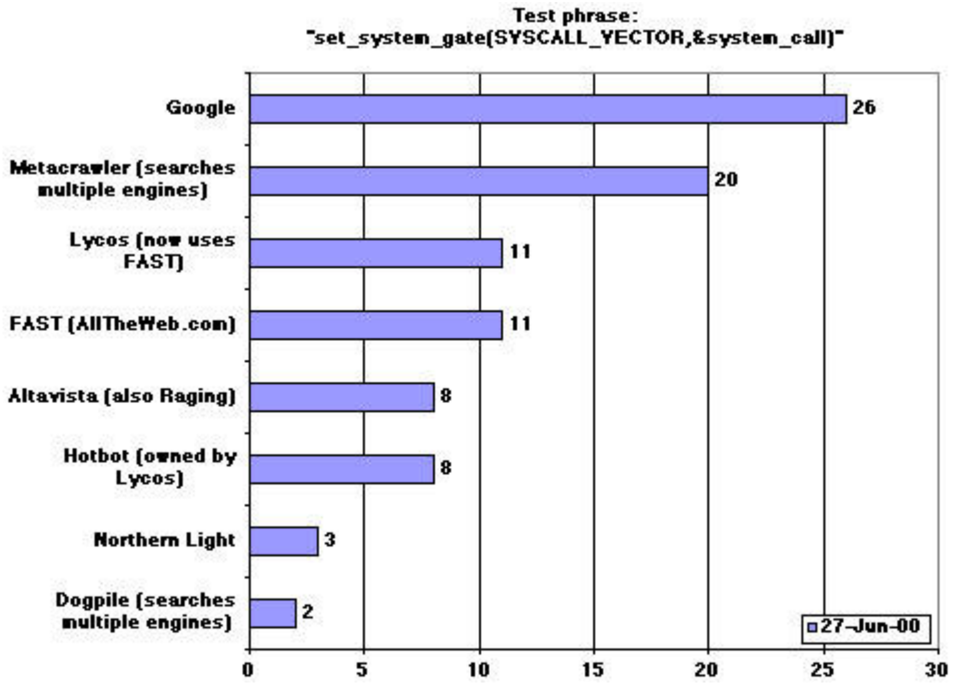


Figure 3. Test Phrase III

As you see, FAST won the first search, by a wide margin, just as it did in December. But Google (see Jason Schumaker's “Interview with Sergey Brin”, page ??) is the clear winner of the second and third searches—and didn't do too bad on the first, either.

Near as I could tell, none of the engines fell for the “breast” bait—at least not in phrase search mode. On that one, they all get a passing grade.

Some of the results, however, were amazing. Go.com found “29,024,074 matches” in the first search, but nothing I wanted. The first ten results all related to breastfeeding or breast cancer. No porn, of course. But I did get a banner ad featuring a happy-looking woman in a cleavage bra. “BREAST AUGMENTATION?” it asked. “Looking for breast augmentation from a doctor in your area? CLICK HERE.” Nice guess, guys.

As we can see, search is consolidating as a business category. By the time you read this, Yahoo! will be using Google (in a deal struck one day before this survey). Other partnerships and cross-investments are sure to follow (Lycos recently bought a 15% stake in FAST, for example). My guess is that there will be fewer search engines by the time you read this.

I just hope there aren't fewer good ones.

> *Doc Searls (info@linuxjournal.com) is Senior Editor of Linux Journal* and co-author of *The Cluetrain Manifesto.*

USPTO Faces Budget Cuts

If you think the U.S. Patent and Trademark Office is issuing too many bad software patents now, wait until October. USPTO Director Q. Todd Dickinson warns of an imminent “reduction in patent quality” when he has to reduce his staff by 1,000 people, including more than 500 patent examiners, out of a total of about 5,000, and cut back or reduce examiners' access to on-line databases. USPTO now has about 3,200 patent examiners.

In a letter to Howard Coble (R-NC) and Howard Berman (D-CA), Chairman and senior Democrat of the House Subcommittee on Courts and Intellectual Property respectively, Dickinson also warns that many patents may be delayed or not issued at all because of the tight budget for fiscal 2001, which begins in October of this year. The letter is online at www.uspto.gov/c_b_directorresponse.pdf

News for job hunters: the USPTO is hiring patent examiners. As a software patent examiner, you will be responsible for researching and understanding the history of inventions, patented or non-patented, in your field and making the decision to grant or refuse patents. According to the USPTO web site, salaries for patent examiners in Engineering and Computer Science range from $27,778 to $53,544. There's a 10% recruitment bonus for computer science and electrical engineering specialists.

LJ INDEX—September 2000

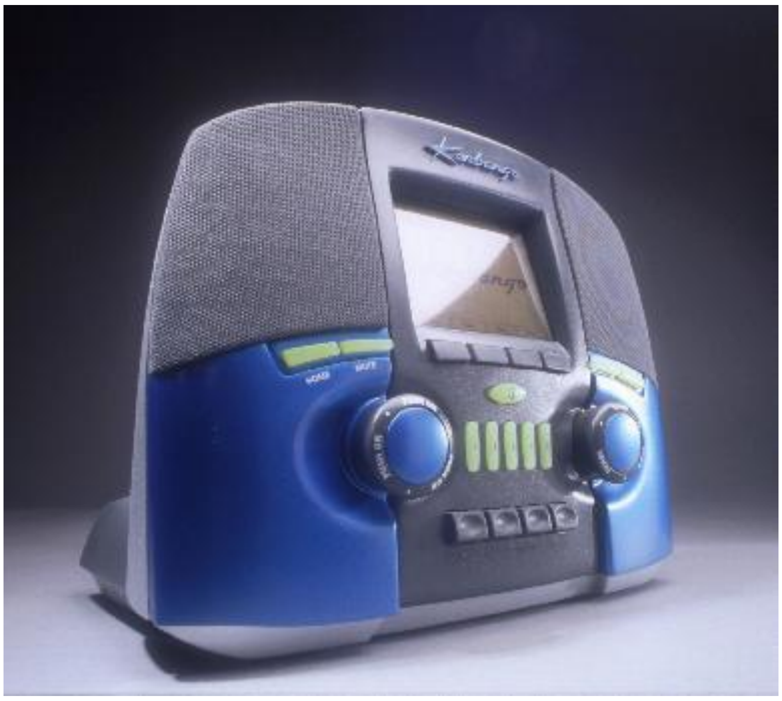
1. Percentage of people who find packaging annoying: **14**
2. Position of shrink-wrap among most annoying packaging types: **#1**
3. Position of CDs among most annoying shrink-wrapped packages: **#1**
4. Percentage most annoyed by CDs, among people who find shrink-wrapped products most annoying: **39**
5. Position claimed by Lycos among the world's search engines, after partnering with FAST: **#1**
6. Number of Web pages surveyed by FAST, according to Lycos, on June 14, 2000: **340,000,000**
7. Number of Web pages Lycos plans to survey with FAST in he coming year: **>1,000,000,000**
8. Number of Web pages in Google's survey index on June 27, 2000: **1,060,000,000**
9. Number of web pages with the word "Doubleclick," according to FAST: **109,973**
10. Number of web pages with the word "Doubleclick," according to Google: **1,129,996**
11. Total compensation of Silicon Valley CEOs in 1999: **$2.3 billion**
12. Percent increase in 1999 Silicon Valley CEO compensation over the prior year: **89**
13. 1999 compensation position of John T. Chambers, CEO of Cisco Systems, among Silicon Valley CEOs: **#1**
14. Chambers' 1999 compensation: **$121,701,629**
15. Chambers' compensation as a multiple of the combined salary of every certified teacher in the San Jose Unified School district: **>2, with $36.6 million left**
16. Number of women in the top ten in compensation: **1**
17. Number of Cisco executives in the top ten: **2**
18. Number of Yahoo! executives in the top ten: **4**
19. Total compensation of all four Yahoo! executives: **$325,463,827**
20. Total compensation of Daimler-Benz top ten executives in 1997: **$11 million**
21. World position of United States in CEO-to-worker pay ratio: **#1**
22. U.S. CEO-to-factory-worker pay ratio in 1999: **34-to-1**
23. Number of women per 175 men among Silicon Valley CEOs: **4**
24. Number of women among the 100 highest-paid Silicon Valley executives: **9**

25. Opening auction price offered by Tracy Cole for her answers to marketing questions: **$20**
26. Number of marketing questions Tracy Cole answers for upwards of $20: **378**

Sources

- 1-4: Cheskin Research packaging study
- 5-11: searches and press releases on Lycos, FAST and Google sites
- 11-24: San Jose *Mercury* News
- 25-26 *The Standard*

STOP THE PRESSES: 3Com, 2, 1 … Kerbango!



The Kerbango Radio, one of the first true Internet appliances, looks like a cross between an old Wurlitzer jukebox and the dashboard of a '54 Buick. It also looks like it'll belong to 3Com soon. On June 27, 2000, 3Com announced that it was buying Kerbango for $80 million in stock. Not bad for a company that was only announced last Fall, employed only 28 people, and still hadn't produced its first and only product. (Though the Kerbango radio is expected to roll out shortly.)

Right now, Kerbango has three unique properties, the radio, its tuning system and its web site, which gives computer users direct access to thousands of stream sources—the “stations” the Kerbango radio will tune when it sits on your kitchen counter, plugged into the household network, listening to streams (those modern “signals”) over your DSL or cable line (or even dial-up, if you're willing to put up with modem-grade lo-fi).

The most significant fact about Kerbango, for our purposes, is the operating system at the heart of the blue box. That would be Linux. We're talking about the same embedded Linux that's the subject of our cover feature this month.

3Com has reason to be eager in this market. IDC expects the Information Appliance business to grow from $2.4 billion to $17.8 billion between 1999 and 2004. This would represent an increase from 11 to 89 million units. Surely a significant part of that 1999 figure is 3Com's own Palm Computing business.

Kerbango was also moving along at a nice clip when the 3Com deal was announced. In a separate agreement, THOMSON multimedia signed a Letter of Intent with Kerbango to brand and distribute an RCA brand Internet Radio that utilizes the Kerbango Internet Radio Tuning Service. 3Com also said it intends to follow a similar licensing strategy with its partners. Presumably that would include Palm.

When this news came in, we got in touch with Jim Gamble, Kerbango's President, and subjected him to a series of snarky questions. Here's how the dialog went.

Doc: Will the name stay?

Jim: Yes.

Doc: Can 3Com actually brand anything?

Jim: Seems like Palm did okay! They grew that from start up to multinational success. I've got no problem re-running that scenario for our radio...

Doc: Will Kerbango remain independent?

Jim: We will be a separate group inside 3Com. They want us to push the radio and our tuning service. In fact, their main input early on has been, “how can we help you do more faster?”

Doc: What will 3Com's “internet audio division” do, and where will Kerbango fit in that? Will the brand include more than the radio?

Jim: The radio and the tuning service is the start. More products will come later too.

Doc: Got any market size projections?

Jim: Most of the projections are for generic “Internet appliances” which conceivably covers a lot of ground. I look at the Arbitron data on Internet radio usage and see that 11 million listeners a month in the U.S. alone is a good market to go after. Too early to say how our numbers will go.

Doc: Will the U.S. finally regain market leadership at the Networked edge of what we've been calling “consumer electronics?”

Jim: I think so. But it will have an international flavor since companies like THOMSON/RCA (who announced they will be shipping a Kerbango powered Internet radio) are thoroughly multinational.

Doc: What we're sensing here at *Linux Journal* is that there is finally a chance that previously non-communicating devices in the home will become Net native and start relating at least with their owners, if not with each other—and that XML-based Instant Messaging will lead the way. Kerbango as a stand alone company didn't play a very big potential role in this scenario; but perhaps Kerbango as the household appliance brand of 3Com could play a huge role.

Jim: Hope so! We view it more as “building a better radio” rather than “building an Internet appliance”. Maybe just semantics, but it makes sense to us.

Doc: We realize this is blue-sky stuff at this point, but being part of 3Com does open the sky a bit. Or maybe the opposite. 3Com has a fine football stadium, but their mass market presence beyond that hasn't exactly been obvious. Some companies are born to brand and promote and make people want their cool stuff. Is 3Com ready to be the next Sony?

Jim: As we like to say at Kerbango, Stay Tuned!

THE BUZZ

What were people talking about in June and early July? Below is a sampling of some of the hotter news stories over the past few weeks, as reported in “The Rookery”, *Linux Journal's* on-line source for news, notes, quotes and reports from the field (updated daily and found at our web site, http://www.linuxjournal.com/):

- Marc Ewing and Frank Batten Jr. selling off large numbers of RHAT stock.
- Oracle paying private investigators to spy on Microsoft.
- An oil spill off the coast of South Africa has endangering a population of approximately 14,000 Jackass penguins.
- Red Hat's deal with Dell.
- Intel jumping on the Internet Appliance bandwagon.
- The release of the CVS repository for the XFree86 development source tree—made available through SourceForge.

- SRO jumping on the Linux bandwagon.
- Poor Corel, part 2!
- Bob Geldof's site running Linux.
- The release of StarOffice 5.2

Archive Index Issue Table of Contents

Advanced search

Focus: Embedded Systems

Don Marti

Issue #77, September 2000

Which embedded Linux tools are going to take off? That depends on accessibility, responsiveness and freedom.

Embedded systems, meet Linux. Linux, meet embedded systems. GNU/Linux as a platform for the next generation of appliances, Internet or other, is one of those great no-brainer technical decisions that comes along every few years. Kind of like Cat-5 cable. But look at how many people agree. If embedded Linux tools were on the shelf at your local computer store, it would take a big shelf to hold them. So which embedded Linux company should you look to when you decide to put Linux on your toaster, or your company's new line of desktop machine tools?

Linux Journal can't give you the answer, because we don't know. But as somebody who has been in the Linux business for a while, I can tell you what works for companies trying to bottle the open-source lightning and sell it, and what doesn't. Which embedded Linux tools are going to take off? That depends on accessibility, responsiveness and freedom.

Focus Articles

Accessibility is easy. Most of the people who will be creating cool stuff with embedded Linux tomorrow are doing something else today. Successful vendors won't assume all customers are embedded systems experts. Maybe they're “power users” of the product category they want to improve with Linux. Or hobbyists, or students. A bundled set with software, documentation and a nifty little targetboard would be a big hit.

Responsiveness is a matter of opening up the company to customer questions and admitting mistakes. For a real-world example of responsiveness in action, read some of what Donnie Barnes of Red Hat posted to redhat-list, back when Red Hat was still refining its distribution with lots of what I'll politely call “user input”. Look for an embedded vendor that isn't afraid to let users help make its products better.

Finally, freedom. Look for a vendor who keeps everything under a license that meets the open-source definition, preferably the GNU General Public License. License hassles are a recipe for driving customers away, as proprietary embedded OS vendors are learning.

What you see in this issue is just the beginning. A lot of you are doing embedded projects right now. If you have an embedded Linux success, let us at *Linux Journal* know. We'll be running step-by-step “how I did it” embedded articles in future issues.

Don Marti is the Technical Editor for *Linux Journal*. He can be reached at info@linuxjournal.com.

From the Publisher: Staff Changes and an Activism Request

Phil Hughes

Issue #77, September 2000

Phil writes about editorial changes.

I expect most of you will already know this from our web announcements, but just to make it official in print, Marjorie Richardson has left *Linux Journal*. Margie worked as Managing Editor and then Editor in Chief of *LJ* since 1997. Her efforts helped us grow, and she will be missed.

I have decided to take this opportunity to make some changes to hone our editorial staff. First, I have done a little re-organization. Rather than have all the responsibility for the content of *LJ* articles fall on one person, I have divided it among three people. While Doc Searls has been responsible for the upFRONT section content and business-related articles, we now make that official.

The other two positions are Technical Editor and Editor at Large. Our *new* technical editor takes over responsibility for the technical articles that appear in *LJ*. This is intended to assure that technical quality is upheld and consistent and that it addresses the interests of *LJ* readers.

The Editor at Large will be the frontman—okay, front-*person*--for the editorial team. This person will work with authors to get articles that fit our editorial direction and be the first-line editorial contact for vendors as well as readers.

To make this all go together, we need an *inside person* who understands how we put *LJ* together. That person is Darcy Whitman. Darcy has been with *LJ* for five years and in the editorial department for the last year. She has proven herself in her ability to organize, and most importantly, she loves vi. So, please welcome Darcy as our new Managing Editor.

What New Technical Editor?

On June 26, Don Marti joined us as Senior Technical Editor. Don Marti is best known for his influential role as Publicity Director of the Silicon Valley Linux Users Group, which currently boasts over 450 members, and also for his outstanding support of the Linux community. We are very excited to have Don Marti join our dynamic team. He's been a part of the Linux movement from the beginning, and has made a tremendous contribution to the community as a whole. I am confident in his talent and leadership ability to ensure *Linux Journal* continues to be at the forefront of the Linux revolution.

When we asked him about his new position at *Linux Journal*, he replied, “I've been a *Linux Journal* subscriber almost since the beginning, and it's always been one of my favorite sources for Linux news and technical information. I'm looking forward to helping put together future issues of a magazine that I really like to read.”

As I write this, we are still evaluating candidates for the Editor at Large position. It is important that we find the right person who has publishing experience and also groks the Linux community.

Until this person is in place, Laurie Tucker, long-time Special Projects person with *LJ*, and I will be filling in as needed. In addition, Heather Mead, who originally joined the company in a marketing position and then moved to Circulation Manager, is transitioning to an Associate Editor position. Heather's fine English skills will be a big plus for the editorial team.

Call for Linux Activism

The most-requested program in the Publishing category of our Software Wish List is FrameMaker, the document layout program from Adobe. Adobe listened, and there has been a beta release out for some time. Now beta 2 is out, and it seems pretty clean and stable. I gave it a test drive and am really impressed with its speed and ease of use. I was even surprised it would import troff documents.

Now it's your turn. If you have any interest in a program like FrameMaker, grab it from the Adobe site (http://www.adobe.com/products/tryadobe/), give it a try and let Adobe know what you think.

A second activism item is SETI, the Search for Extraterrestrial Intelligence. This UC Berkeley project lets people with PCs help analyze data. The good news is that PCs include Linux. You can go to www.setiathome.ssl.berkeley.edu for general information, or directly to our group from the setiathome.ssl.berkeley.edu/team_list.html page and enter “*Linux Journal*” in the search box. Come join us and increase the visibility of Linux and *Linux Journal*.

CALL FOR AUTHORS

email: info@linuxjournal.com

Still Searching

Doc Searls

Issue #77, September 2000

I want a search engine to grep the world for me.

Water, water, every where, Nor any drop to drink. --Samuel Taylor Coleridge

If information were water and search engines were taps, I'd chug about ten gallons a day. I don't just want it. I need it. Name a verb expression for appetite, put it between I and information, and you start to see where I'm coming from.

Most of the information I crave is specific and textual; and since most specific text information involves more than one word in a row, I'm usually looking for alphanumeric strings. Sometimes, but not always, those strings are words.

In other words, I want a search engine to grep the world for me.

Since I'm sure this is no-hope-in-hell territory, I just went looking for a quote to express my frustration. I found it through Google. It's from the prolific Dmitry Kirsanov, writing in the “Advanced” chapters of *HTML Unleashed, Professional Reference Edition*, the full text of which is exposed online at http://www.webreference.com/dlab/books/html-pre/. He writes, “Those accustomed to grep-style regular expressions can't even dream of using something similar with search engines.”

We won't go into why. But we do have to ask if it's necessary for so many engines to be as bad as they are about this. It seems like every time I find a search engine that does The Job, somebody buys it and it goes to hell. Or it goes to hell and then somebody buys it. Whatever order, the bad news is always around the corner. And, without fail, it comes in the form of (here comes that dreaded word)...marketing.

Marketing can't seem to help trading utility for “reach”, ''exposure'', “targeting” or some other ''strategic'' abstraction intended to influence the largest possible population—ignoring the fact that today's common denominators aren't as low and wide as they used to be. Especially on the web.

The uncommon denominators (that's us again) are rather abundant, too. And it's not like marketing lives to ignore the connoisseur. Witness wars among automobile makers over handling characteristics only mechanics and race car drivers can fully understand. (How many of us careen our Chevy Tahoes down black diamond slopes or floor our Acuras to tire-melting speeds on 2-lane Nevada backroads?)

But dot-com marketing is a wacky breed that often lusts after consumers to a degree not seen outside Procter & Gamble, circa 1958. It was consumer-hungry marketing that killed Lycos (the original one that came from Carnegie Melon), then Infoseek, then Hotbot (as Inktomi created it), then Altavista (as DEC created it). Each of those was born to serve intellectual curiosity. Others—Looksmart, AskJeeves, Go, Yahoo! and DirectHit, to name a few—were born good for little other than flat-ass-dumb “consumer” searches for “favorites”, “portals” and whatnot.

Okay, I'll make an exception for Yahoo!, which has always used human beings to catalog the Web. And now it hired Google to do the heavy lifting, which is a good thing. We'll say more about Google after the Altavista autopsy.

I knew Altavista was terminal last fall, when its “advanced” search page was suddenly replaced by bragging about “improvements”. “Tips” were gone. So was a nice and easy way to search for inbound links to a given URL. If the function persisted, no clues to procedures remained (at least that I could find) amidst the fresh marketing poop.

Of course, there was a survey. So I filled it out. This came back by e-mail:

> Thank you very much for recently filling out the survey on the AltaVista Advanced Search page. Your suggestions and comments will help us continue to make AltaVista Advanced Search the best way to location [sic] information on the Internet.
>
> We've found that once users try Advanced Search, they realize the power of the tools that AltaVista has to make your searching more precise. So, we're always trying to come up with new ways to encourage regular

> searchers to try out Advanced Search. And, that's where we'd like your opinions.
>
> As you'll recall, along with your survey responses you also submitted this email address. So if we had any follow-up questions we could contact you again. Well now we'd like your opinions about some promotions we'd like to use to convince users to give Advanced Search a try.
>
> The six-question survey should take only a couple of minutes. Simply click on the hyperlink below. If you are using AOL or another email service that does not support hyperlinks, please copy the hyperlink and paste it into your browser window in the Address box.

The URL delivered me to a silly world whose gods believe that prizes might do what features won't. I wrote this in the Input box:

> I don't want a prize. If your search is so damned “advanced” (and how can it be, now that you fail to even mention the wonderful “link:www.mysite.com -url:mysite.com” feature that only works in BASIC, fercrissake!), I'd be glad to help out for FREE. I'll give you my time if you'll give me your attention. I want truly advanced search functions. THAT's what tempts me. Not prizes.

At the bottom of page two, the system broke down and wouldn't advance me to page three. I gave up and didn't go back except to see how it compared with competition.

But hey, maybe they listened. The “cheat sheet” at doc.altavista.com/adv_search/syntax.shtml restores a lot of the good advice lost from the original Advanced page. But the sad fact is that Altavista isn't as good as it used to be. FAST, Google and even MSN Search yield better results. If you're looking for strings. (For more, see the feature in UpFront.)

I know because I test search engines pretty often. I go deep into a document somewhere in my own domain, http://www.searls.com/, and grab a string of text that appears both in my own site and in a number of others, such as a quote from literature. Then, I run a bunch of engines through the mill. That's how I knew when Altavista started to beat Infoseek, when Hotbot began to beat Altavista and when FAST began to beat Hotbot.

The last time I saved results was December 3, 1999. At that time, FAST, http://www.alltheweb.com/, won. They still win (see UpFront) on some tests but lose on others—usually to Google.

There's an awful lot to like about Google. First, they run their engine on Linux (in case you're asking, Google is a huge Red Hat customer). Second, their user interface is blessedly simple and devoid of hype for anything other than itself, and there's precious little of that. Design-wise, they make great use of white space. Third, they have nicely incorporated the DMOZ Open Directory project's catalog, which is essentially The People's Yahoo!. But fourth (and most importantly), they have done more than any other search company to allow trusting searches of both strings and collections of words at the same time.

Where FAST and Hotbot give you a pop-out menu choice of “the phrase”, “all the words” and “any of the words” (or the equivalents), Google does all of those at once, defaulting first to phrase mode. You can use quotes to narrow the results, but the difference is usually small. That means Google has done a good job of providing the largest possible narrow search. In fact, the one site users want comes out on top so often that Google confidently provides an “I'm Feeling Lucky” button that yields just one result.

Google does have special search functions, all well-explained. I wish they had more, but I can live without them. Google is, for me, by far the most useful and reliable search engine—at least for those hard-to-find word strings.

What's not to like about Google? Well, there's the Patent Issue. They're going after patents on their search methods (and who knows what else), which costs them good will in the Linux/Open Source community. At an event early this year, I talked briefly about patents with Google's co-founder, Larry Page. It was clear that Larry isn't crazy about them. Immediately afterwards, I talked with John Doerr. It was equally clear that John is crazy about patents—to the degree that he believes that patents are one of the things that “make America great.” John, of course, is a VC with Kleiner Perkins, which conspicuously funds Google.

There's also the risk that Google will pursue an advertising-driven business strategy. In current searches, ads show up as annotated, text-only links, posted above search results. These are certainly far less onerous than banners (also harder to block). But they quietly crept in a few months ago. What's the next step?

I don't know. In fact, I just tried to force Google's engine to give me an ad, and it wouldn't do it—not once in ten tries. So I suspect that the company is being cautious. As it should. Far as I know, Google is the only search engine created by and for people who live to search and search to live.

We need more of those. And we need the ones we've got to remember why we like them so much.

Doc Searls (info@linuxjournal.com) is Senior Editor of *Linux Journal* and co-author of *The Cluetrain Manifesto*.

Best of Technical Support

Various

Issue #77, September 2000

Our experts answer your technical questions.

Graphics on a Floppy

How do I create an image file from a floppy disk? I have tried to find this everywhere, with no success. —Bryan Hepkema, bhepkema@blsc.com

If I get your question correctly, to generate an exact copy of what is on a floppy disk, and store it on your hard disk, you can use the command **dd if=/dev/floppy of=/tmp/floppy_image**. You may have to input the correct device name for your floppy unit (possibly /dev/fd0). This will create a binary image of whatever is on the diskette in the file /temp/floppy_image. I would suggest you study the dd command with man (man dd). —Felipe E. Barousse Boué, fbarousse@piensa.com

If you need an image from a file, just insert the floppy and **cat /dev/fd0 > file**. The device representing the floppy is just a file representing the whole disk. That's true for hard drives as well. —Alessandro Rubini, rubini@linux.it

Web Fonts and Printing

How do I get Netscape to print fonts like the web pages actually show? Everything is defaulted to Times New Roman. How do I load and use True-Type fonts? —Greg Crutcher, madcrutch@earthlink.net

There is no short answer to your question. Many Xwindows configuration issues have to be addressed. The right answers can be found at the following URL: pegasus.rutgers.edu/~elflord/font_howto/html/index.html#toc1. It is also worth mentioning that Netscape has its own personality regarding fonts and behavior under X. —Felipe E. Barousse Boué, fbarousse@piensa.com

Check out the TrueType-HOWTO at www.moisty.org/~brion/linux/TrueType-HOWTO.html--Pierre Ficheux, pficheux@com1.fr

Sharing Files with Windows

I have installed Caldera's OpenLinux eDesktop 2.4 on my HP Pavillion 6630 computer. I can dual boot into either Linux or Win 98 and would like to be able to share files between operating systems. Under Linux, the file directory for Windows is visible under hda, and I browse through my Windows files and even open them, but I cannot write to this directory. Is there any easy way to set up a location on the hard drive where I can read and write to files from either Linux or Windows? —Tom Newman, newmanth@usa.net

It looks like you don't have permission to write to the files. That's what I would expect if you work as a user and the directory is automatically mounted at boot. Try adding “umask=0” in /etc/fstab, in the field where “default” appears to allow everybody to write the files. To have it be effective immediately, try **umount /dos; mount /dos**. —Alessandro Rubini, rubini@linux.it

PostgreSQL on Boot

How can I have PostgreSQL start automatically when my computer boots up? I must start the postgreSQL server with the command **nohup postmaster -d 2 > logfile 2>&1 &**, and I must start it as the user “postgres”. I can't figure out where in the /etc/rc.d scripts I should start it from. Thanks for your help! —Warren Killian, warrenk@in.net

The best way to start new programs is by writing a new script, just for them, in /etc/rc.d/init.d (talking RH-hierarchy, other distributions may be slightly different in the pathnames) and then link it from /etc/rc.d/rc3.d. The instructions to write those scripts in the right way should be part of the documentation for your distribution; another good way to do things is to learn (and copy) from what other scripts do. As far as links in /etc/rc.d/rc*.d, there is also a graphic tool to make those links, part of the control-panel tool. Note that Debian is already packaging Postres, so you can look at that script as well. —Alessandro Rubini, rubini@linux.it

The script should use a syntax such as

```
su postgres -c '/usr/bin/postmaster ...'
```

However, a “postgresql” init script is included in the distribution so that it is installed in the /etc/rc.d/inid.d directory. —Pierre Ficheux, pficheux@com1.fr

Trouble Logging In

I got Corel Linux installed alongside MS Win98 okay, but now, when I boot to Linux, the login screen flashes and accepts characters only when the screen is displayed (at about half-second intervals, on and off). I cannot catch the on part in order to successfully log in. What is wrong, and how is it fixed? I tried typing kde at the command prompt, but it cannot load x. It runs okay in VGA mode and logs in to kde okay. I am a total Linux newbie but am familiar with MS-DOS and CP/M (showing my age now!) Many thanks. —Adam Puk, adam-puk@cwcom.net

Your symptoms may mean that the system is driving the screen at incorrect frequencies (the PC world is too varied; it's not easy at all to devise things that work for all possible hardware), but there should be no relationship between that and reaction to keboard input. What you may try is booting in single user and then look for the command that corrupts things by executing the init scripts one at a time. Unfortunately, to do this kind of tracing, you'd need to already have good experience. I'm afraid your best bet is to look for a knowleadgeble neighbor. —Alessandro Rubini, rubini@linux.it

Cannot Access the Web

I am trying to make the system run on my cable modem. I have two other computers running on it, but they are Windows-operated. I have entered all the information that is needed, IP, Gateway, subnet mask, as well as host and domain. But every time I try to run Netscape and go to a web page, it gives me an error message that the web server cannot be found. I cannot seem to figure out what the problem might be; I have tried to follow all the documentation I could get my hands on. Hopefully you can help me. —Chirag Jay Patel, cjpagni@home.com

It looks like you have a routing problem. Try this command:

```
route add default gw <ip_of_your_cable_modem_host>
```

That is the IP address of the host that has attached the cable modem (or the IP address of the modem itself, depending on your hardware setup). What happens here is that this command tells Linux's TCP/IP to go out through the standard and default exit to the Internet, which is your cable modem. —Felipe E. Barousse Boué, fbarousse@piensa.com

Failure to Connect

When I try to connect to my ISP, I hear a dial tone but there is no connection. In /var/log/messages, there is the nex message: Peer is not authorized to use remote address x.x.x.x Where x.x.x.x is het number from the ISP. I searched the

Internet for this but got very little information. Thank you in advance. —Wim van den Broeke, w.vd.broeke@hccnet.nl

Try adding the line

```
noauth
```

to the /etc/ppp/options file so the peer is allowed to set the route from the ISP assigned address, which is assumed not to be set at the moment of connection. Do a man pppd and check all options for the pppd daemon on the “options” setup file. —Felipe E. Barousse Boué, fbarousse@piensa.com

Configuration Questions

I am new to Linux, and I like it a lot. I would like to know how to do the following:1. Set the desktop display to conform to the monitor. 2. Set up the screen to be able to fit the size of my monitor. 3. Change the color scoop from 16bit to more colors. (Still monitor) 4. I am confused about Samba. Please advise on how to connect Samba file share with Windows. —Luke, godmatrix@godmatrix.com

1. The Xconfigurator command will guide you on the X setup. Use “custom” monitor and get the correct values of scan frequencies from your monitor manual specs.

2. The highest resolution can be achieved with Xconfigurator, as well. Make sure your monitor supports high resolutions!

3. After Xconfigurator, you can try editing /etc/X11/XF86Config to define default video modes, let's say 800x600 or 1024x768 on your video card's settings section. You have to make sure you have enough video memory to support high color depths.

4. You need to set up the IP and host name of your Windows PC on /etc/hosts. That is, host names must be correctly resolved. Edit /etc/smb.conf and add your workgroup name to the line: workgroup = YOUR_WORKGROUP_NAME. Then, add the server name to the line: server string = YOUR_SAMBA_SERVER_NAME. Finally, edit the line to say hosts allow = xxx.yyy.xxx. 127. You must write the first three number groups of your IP addresses (denoting the LAN address itself) immediately after a blank space and the “127.”, indicating the localhost. Note the trailing dots after network and localhost-network IP numbers. Lastly, share a directory from Linux to Windows, let's say:

```
[linuxtemp]
   comment = Linux /tmp directory
   path = /tmp
   public = yes
```

```
      writable = yes
      printable = no
```

This will share the /tmp directory under the “linuxtemp” name. Restart Samba with the command /etc/rc.d/init.d/smb restart and double click on the network neighborhood of your Windows box; you should see the Linux's share /tmp directory.

Samba is a complex service. I would suggest you look up further details from the Linux documentation project site at metalab.unc.edu/pub/Linux/docs/HOWTO/html_single/SMB-HOWTO--Felipe E. Barousse Boué, fbarousse@piensa.com

New Products

Ellen M. Dahl

Issue #77, September 2000

Heavy Gear II, Compaq Power Management Software and more.

Heavy Gear II

The Heavy Gear II game has 3-D audio effects and joystick support and makes optimal use of the qualities of Linux in the network. Multi-player games based on rounds, or real time, are possible. A player is the captain of a batallion of robots, equipped with a battle suit called Gear. They have modern technology and high-performance weapons to use in defense.

Contact: Loki Entertainment Software, 250 El Camino Real, Suite #100, Tustin, CA 92780, sales@lokigames.com, software@suse.de, http://www.lokigames.com/, www.suse.de/en/produkte/software/index.html.

Compaq Power Management Software

Compaq's Power Management Software Version 1.7, with LanSafe III, allows IT managers to configure and have control over the enterprise and e-business framework. The new additions help network administrators configure all the networked UPSs, in any environment. The software supports current versions of Red Hat and Turbo Laser Linux in both English and Japanese. Other Power Management Software features include PowerScope; load segmenting; automatic, unattended and prioritized shutdown; automatic alerts and statistical logs; on-the-fly configuration and password protection.

Contact: Compaq Computer Corporation, P.O. Box 692000, Houston, TX 77269-2000, 281-370-0670, 281-514-1740 (fax), www.compaq.com/ups.

Open Motif Everywhere

Integrated Computer Solutions, Inc. issued the first upgrade to its Open Motif Everywhere distribution. This upgrade includes bug fixes generated by the Open Source community and SRPMs for building one's own version of Open Motif from source code. Open Motif Everywhere is the ICS distribution of Open Motif, built from the official Open Group 2.1.30 sources. The binaries and sources to Open Motif Everywhere are available for free download through ICS' Open Motif portal site. A CD-ROM distribution, containing binaries for most popular Linux and FreeBSD distributions, Open Motif source code, Adobe PDF versions of the official Motif 2.1 Manual set and e-mail installation support, is also available.

Contact: Integrated Computer Solutions, 201 Broadway, Cambridge, MA 02139, 617-621-0060, 617-621-9555 (fax), sales@ics.com, http://www.motifzone.net/, http://www.ics.com/.

VXA RakPak

The VXA RakPak is the first rack-mountable tape subsystem to use the tape technology innovation VXA. Ecrix has packaged two VXA-1 tape drives in a single rack unit (1u), creating a data-protection system for rack-optimized servers and storage. RakPak stores up to 132GB of data at a 12MB/s aggregate transfer rate (2:1 compression). The VXA RakPak guarantees data restore based on three complementary components: discrete packet format (DPF), variable speed operation (VSO) and OverScan Operation (OSO). VXA drives are compatible with all major OSes including Linux.

Contact: Ecrix Corporation, 5525 Central Ave., Boulder, CO 80301, 303-402-9262, 303.402.9266 (fax), info@ecrix.com, http://www.ecrix.com/.

Embedix SDK

The Embedix SDK for x86 is a software development kit that simplifies the development of embedded devices and systems. Embedix SDK is designed to reduce the system requirements, development time and overall cost of deploying embedded software solutions. Included is Embedix Target Wizard, a graphical configuration tool that allows OEMs to select Linux components, identify interdependencies and automate the configuration of a small, highly specialized software solution for embedded devices. Embedix SDK ships with a complete version of a Linux development host operating system. Multi-user licenses are available.

Contact: Lineo, Inc., 390 South 400 West, Lindon, UT 84042, 801-426-5001, 801-426-6166 (fax), sales@lineo.com, http://www.lineo.com/.

Gaussian on Linux NetworX

Linux NetworX products are now available with Gaussian 98, the latest in the Gaussian series of electronic structure applications. Gaussian is used by chemists and chemical engineers for research. Users can model chemical systems and phenomena in order to predict their properties, study chemical reactions and apply the fundamental results to their own investigations. Operating on a Linux NetworX cluster system, Gaussian helps users predict energies, molecular structures, vibrational frequencies and other molecular properties in both the gas phase and in solution, and can study both ground state and excited state molecular systems.

Contact: Linux NetworX, Inc., 8689 South 700 West, Sandy, UT 84070, 877-505-LNXI or 801-562-1010, 801-568-1010 (fax), sales@linuxnetworx.com, http://www.linuxnetworx.com/.

HELIOS EtherShare 2.6

Major features of EtherShare 2.6, UNIX-based server software, include implementation of AppleTalk networking, such as AppleShare IP, support for routing, advanced spooling and print support, and PostScript Type 1 and 3 font server for all printers and UNIX Text to PostScript printing. New to EtherShare 2.6 are Japanese Shift-JIS support; user- and password-protected print queues; HELIOS Mail support for MIME with file attachment formats including base64, binhex, uuencode and uuencode; improved memory management by HELIOS Mail, requiring only 1 MB memory regardless of attachment size; and support for Mac OS Navigation services. EtherShare 2.6 is available for a wide variety of UNIX-based systems including Linux Pentium-based servers.

Contact: HELIOS Software GmbH, Attn: Marketing, Steinriede 3, D-30827 Garbsen, Germany, +49-5131-709320, +49-5131-709325 (fax), marketing@HELIOS.de, http://www.HELIOS.com/.

Contact: European Mikrograf Corporation, 269 Mt. Hermon Road, Suite 100, Scotts Valley, CA 95066, 831-461-6061, 831-461-6056 (fax), info@ugraf.com, http://www.ugraf.com/.

NUMA-Q E410 and Netfinity 3500 M20

IBM introduced its Intel-based server, the 64-processor NUMA-Q E410, along with a two-way server, the Netfinity 3500 M20. Powered by Intel's new 700 MHz Pentium III Xeon processors, the NUMA-Q E410 is a scalable server line for e-businesses running in Windows 2000 and Linux environments. Key features of the NUMA-Q E410 include high availability, performance and scalability; investment protection; and NUMACenter. The Netfinity 3500 M20 is a two-way SMP server supporting Pentium III 800 MHz processors and up to 2GB of memory. In addition to the new 800 MHz Pentium III processors featuring a 133MHz Front Side Bus processor architecture, the Netfinity 3500 M20 delivers high-speed I/O with the Ultra 160 SCSI interface and 64-bit PCI.

Contact: Shop IBM, Dept. YES98, PO Box 2690, Atlanta, GA 30301, 800-257-9044, 800-2IBM-FAX (fax), code YEF98, ibm_direct@vnet.ibm.com, www.ibm.com/servers.

InstallShield Java Edition 3.5

InstallShield Software Corporation has released InstallShield Java Edition 3.5. Utilizing “Web Start” Wizard technology jointly developed by InstallShield and Sun Microsystems, Java Edition 3.5 expands InstallShield's installation-authoring solution by allowing the creation of a single installation for multiple platforms, including Red Hat Linux. It includes new Java Native Interface support; dynamic file refresh functionality; a silent install feature; one-step support for JVM Bundling; a powerful IDE; suite installation; default project templates; extensible JavaBean model; a new setup design model; and extended internationalization support. Both the full and upgrade versions are available.

Contact: InstallShield Software Corporation, 900 National Parkway, Suite 125, Schaumburg, IL 60173-5108, 847-240-9111, 847-240-9120 (fax), info@installshield.com, http://www.installshield.com/.

NetMAX IA Software

The NetMAX FireWall ProSuite and NetMAX Internet Server ProSuite allows users to secure, connect and share their networks through a variety of customizable features. Both ProSuites include and install a version of Linux based on the Red Hat distribution. The products feature 128-bit SSL and cache storage, and interface with the new NetMAX VPN Server software. The latest version of the NetMAX Professional Suite bundles the new FireWall ProSuite and Internet Server ProSuite with the current version of the NetMAX FileServer into one integrated package.

Contact: Cybernet Systems Corporation, 727 Airport Blvd., Ann Arbor, MI 48108-1639, 734-668-2567, 734-668-8780 (fax), info@cybernet.com, http://www.cybernet.com/, http://www.netmax.com/.

Teamware Office 5.3 for Linux

The final version of Teamware Office 5.3 for the Linux platform is a ready-to-run groupware product with a customizable user interface and updated communication features. WebService tools as well as server tools are run from the server console. Teamware Office 5.3 for Linux also allows scheduling of meetings, allocating of resources and attaching documents to appointments. Mobile users can access their mail and other Teamware Office services using PDA browsers. Teamware Office 5.3 for Linux can be used on Linux distributions that use RPM for installing binary files.

Contact: Teamware Group Oy, P.O. Box 135, FIN-00381 Helsinki, Finland, +358-0-9-512811, teamware.marcomms@teamware.com, http://www.teamware.com/linux/.

TimeSys Linux/RT 1.0

The TimeSys Linux/RT 1.0 is a real-time version of the Linux OS and a distribution which includes subsystems that offer a scalable and innovative approach to meeting real-time and quality of service constraints. Developers can combine these subsystems to handle a variety of application requirements, from small footprint to full-featured architectures. The distribution is available for free download from the Web and in three different editions (Standard, Deluxe, Professional) for purchase, all of which come with limited installation support.

Contact: TimeSys Corporation, 4516 Henry St., Pittsburgh, PA 15213, 888-432-TIME, 412-681-5522, info@timesys.com, http://www.timesys.com/.

Linux 1-2-3 Fundamentals CD

Linux 1-2-3 Fundamentals is the first disk of computer-based training in a series of open-source educational products. The interactive CBT is a tutorial designed to introduce one to the Linux OS. Topics range from file systems, shells and file security to command-line mail, GUI and KDE. The learning environment of Linux 1-2-3 Fundamentals does not assume the student is a Linux (or UNIX) expert. The CD comes with the Red Hat, Mandrake and SuSE distributions, provided by a partnership with LinuxCentral, an e-commerce web site.

Contact: OmniLinux, 39465 Paseo Padre Pkwy. #2900, Fremont, CA 94538, 510-249-1060 ext. 311, 520-563-1070 (fax), info@omnilinux.com, http://www.omnilinux.com/.

Where to Send Press Releases

Archive Index Issue Table of Contents

Advanced search