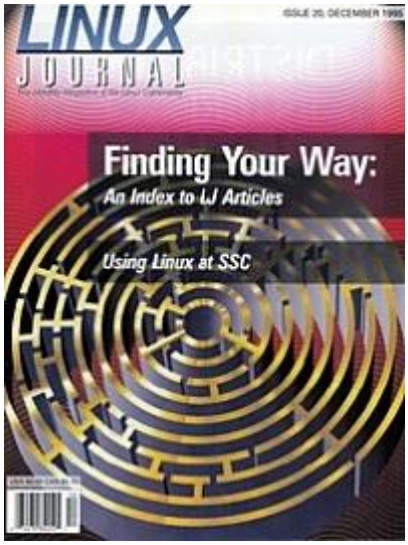
Linux Journal Issue #20/December 1995



New Products
Kernel Korner : Porting Linux to the DEC Alpha: The User Environment *by Jim Paradis*

Archive Index

Advanced search

Linux in the Real World

Kevin Pierce

Issue #20, December 1995

Find out how SSC uses Linux to produce all of their products, including this magazine.

When I came to SSC (publishers of *Linux Journal*), I was told the first thing I had to do was learn the computer system. Having never been exposed to Unix, I set out to discover as much as I could. Coming from an MS-Windows environment, I had a lot to learn. The more I learned about the system we use, the more questions I asked. Here is what I found out.

The first thing I noticed was the multi-tasking capabilities of Linux (I'm not even going to get into Win95). Everyone at SSC has a Linux system (workstation) at their desk, which they log into every morning. In addition, there are two non-Linux systems in the office: a Windows for Workgroups system used for graphics and magazine layout and a Unix System V, Release 4.2 system used to run the Progress database, which has not yet been ported to Linux.

Once logged onto their local system, users can perform tasks locally (such as reading electronic mail) or access the other computers via rlogin, telnet, ftp, and so on.

Our Network

All of the workstations are linked via a thin Ethernet backbone, except for a few which are connected via twisted pair Ethernet to a twisted pair hub, which is then connected to the thinnet backbone.

The main backbone ends at the Orion Firewall System that sits between the internal network and a second, externally visible network that connects to the Internet through a Xyplex router and a CSU/DSU (also known as a “digital modem”) over a T1 Frame Relay connection to our Internet Service Provider,

Alternate Access, Inc. Our Web server, www.ssc.com, is also on this externally visible network, outside our firewall.

There is a third network in the office. The Windows for Workgroups (WfW) machine is on this network with one Linux system also connected to the regular internal network. This setup keeps the large amount of traffic between the WfW system and the Linux system (which drives the Imagesetter) from bogging down the main network.

Network Management

Often, in a multi-user environment like ours, every computer has a unique password file, local to that system. If someone wants to change their password, they have to log into every system individually to make the change office-wide. All of our systems instead use NIS (Network Information Service) to centrally manage all password and group files, access permissions, host address information, and data on a single server. NIS distributes a single master password file to all the systems transparently. Since the network is running NFS (the Network File System), files can be accessed between systems easily. This is easier for both the user and the administrator.

SSC uses sendmail as its mail daemon to monitor and manage the delivery of all electronic mail messages. Sendmail is the de facto standard Mail Transfer Agent (MTA) for most complicated networks. Although it is not easy to configure, it is the most configurable and most flexible of all the mail daemons available. It determines whether each e-mail address is local or remote, delivers local e-mail locally, and sends remote e-mail to remote systems via the Internet (using SMTP, the Simple Mail Transfer Protocol) or UUCP (described later).

All outgoing mail at SSC is routed through a single workstation for delivery via sendmail. This centralizes the e-mail system so there is only one log file, one daemon, one thing to break and be fixed. Incoming mail is queued on the Web server outside the Orion firewall system with smap, a *secure* mail queue program. Smap acts like a normal mail daemon and queues mail. Then smap calls sendmail to process the queued mail, sending it to the real internal hub for local delivery. The smap client implements a minimal version of SMTP, accepting messages from over the network and writing them to the disk for future delivery by smapd. Like anonymous ftp, smap is designed to run under chroot, except it also runs as a non-privileged process to overcome potential security risks presented by privileged mailers running where they can be accessed over a network. Sendmail still runs, but only when it's told to, instead of all the time.

UUCP (**U**nix-to-**U**nix **c**o**p**y) delivered mail is also forwarded to the sendmail hub via smap. Sendmail then sorts the *regular* SMTP mail from the UUCP delivered mail. The SMTP mail is delivered locally and the UUCP delivered mail is spooled in directories where the UUCP system can find them. Since the modems which deliver the spooled UUCP delivered mail to local recipients are on the Web server, which is on the externally-visible network, these files are transfered from the internal system to the Web server with tar and scp, a secure version of the rcp (remote copy) command.

tar (**t**ape **ar**chive) and scp (**s**ecure **c**o**p**y) are used every three hours to transfer the mail automatically to the Web server. The mail is then deleted from the local workstation to avoid duplication.

By handling mail this way, only one machine, the Web server, needs to have a modem and access/exposure to the outside world, and the line doesn't need to remain open.

PPP (Point-to-Point Protocol) service allows employees remote dial-in access outside the firewall by the Web server. Users then access the internal network (and their own desktop workstations, if they wish) using **ssh**, a **s**ecure **sh**ell that encrypts all the data sent between the internal workstation being accessed and the Web server. Employees can also use ssh to get from their home computers to the Web server ensuring a completely secure line, or telnet if they don't have ssh on their home machines.we

All the user home directories at SSC, as well as the local binaries directory, are shared via NFS (Network File System) between all workstations. With the current system, every time users want to read files in their home directory, the files must be transferred across the network to their computers. Soon we will be moving all the user directories from the office file server to each user's own workstation for the following reasons:

- Transferring file across the network is much slower than transfering them from the local hard drives, making file access slower.
- Also the network has a limited amount of bandwidth (amount of information it can carry at one time), and eliminating unnecessary use will speed up the network.

The Web Server

SSC's Web server is an AMD 486DX4/100 machine running Linux and the Apache server software. The server contains SSC's catalog and product information, as well as information on *Linux Journal*, including covers and tables of contents from all issues, selected articles from some issues, and

advertiser indices. We also offer space for sale to *Linux Journal* and *WEBsmith* advertisers if they need a home for their Web pages.

One of the big advantages of Apache is that it can appear to be different servers with different domain names and IP addresses—that is, it is "multi-homed". This makes it possible for our single computer running one Web server to serve as the Web server for Zebu Systems, L.L.C. (http://www.zebu.com/) and Cucumber Information Systems (http://www.cucumber.com/) as well.

The server has been in operation since May and accesses continue to increase. We are currently receiving around 35,000 hits per day. Apache is very efficient and reliable. Even with only 16MB of RAM on the machine, we seldom see a load average of over 0.2. This means there is an average of one process waiting to run 20% of the time, and the rest of the time the machine is idle.

Keeping the Web server outside of the Orion Firewall System keeps the internal network safe in the event the Web server is compromised. The Web server is mirrored from the internal network so that if it is compromised, it can be restored easily.

SSC Computer Schematic

Printing

Most of the printing we do at SSC is done using PostScript:

PostScript is a Page Description Language (PDL) developed by Adobe Systems, Inc. PostScript tells any printer which has PostScript built in, how to print a page that consists of text and/or graphics. The page must be generated by software that includes a driver which converts the page into PostScript code; the code, in turn, is translated by the printer. PostScript is the de facto PDL standard for high-end desktop publishing because, among other reasons, it can operate across a range of platforms, is very precise, and has color capabilities. Holt and Morgan—UNIX: An Open Systems Dictionary.

Text files, such as invoices, that we need to print out are done in ASCII. They are sent to one of the dot matrix printers, or the one laser printer reserved for printing plain text.

We have seven printers on the network, shared via lpd. Users can select their default printer by setting their **PRINTER** environment variable.

Many of the printers are selected based on their location relative to the person using them. Others may be selected because of their speed. One special printer, a Tektronix Phaser III PXi is a wax-transfer color PostScript printer. This is used for producing color proofs of SSC products, magazine covers and pages, and other graphics such as Web pages.

Database

The database we use, Progress, isn't available for Linux. Therefore, the database is run on a Unix System V, release 4.2 system. Progress is run in character mode, and users access the database via rlogin or telnet from their Linux workstations.

This database is used to store all customer and vendor-related information. This includes customer contacts, subscription information, sales transactions, reader service leads, the Linux consultants directory and a whole lot more. We are in the process of moving advertising booking to this database as well. Other *small* databases (article tracking, for example) are written in Perl and run on Linux systems.

We have used Progress for years. (We used to run the whole office on a Xenix system.) If we had it to do all over again, we would select a database methodology that would make it possible to run everything on Linux and not have a *foreign* Unix system in the network. While it is not a reliability problem, it does mean we have to support another operating system.

Magazine Production

When I tell people I work for a publishing company, many ask, “Are you using a Macintosh for your layout?” The answer is “no”. We use Quark Express and Corel Draw! on Windows for Workgroups and tie the process directly into our Linux network. In order to explain this, let's examine the magazine production process from start to finish.

First, Michael K. Johnson, the editor, finds people to write articles for *Linux Journal* on various topics. Note that Michael is located in North Carolina, while the remainder of SSC's staff is located in Seattle. This means that Michael uses his local Linux systems to do much of his work and uses his Internet connection to access machines in Seattle. When the articles are sent in (via e-mail), Michael edits them and sends them to our production editor in Seattle. At this point the articles that he sends are in Quark Tag Format—ASCII text with various *escape sequences* added to them to indicate paragraph types, font changes, and other formatting. We are currently developing a new language closely related to HTML (HyperText Markup Language), the language of the World Wide Web.

Once this language is complete, we will be able to automatically translate articles into Quark Tag Format for layout and into HTML for addition to our Web site.

The production editor files the articles, runs some basic checks [like spelling; yours truly can't be trusted to spell anything right—ED] and prints them out in preparation for our copy editors. The print process is done using a shell script that uses sed and awk to translate the Quark tagged file into troff source and pipes the result through groff to produce a reasonable approximation of what the article will look like when it is imported into Quark and printed.

Articles returned from copy editors are then reviewed by the production editor and changes are made as needed. This step sometimes involves contacting the author for clarification—a step that is generally carried out via e-mail.

The final version of the articles are slightly modified by another shell script which like the first, uses sed and awk to do its work. This is necessary because Quark requires that paragraphs be one continuous line. Also there are some particularly awful Quark escape sequences that we have aliased to simpler escape sequences which need to be converted to the real sequences in order to be interpreted correctly by Quark. These modified files are written to a location on the Linux filesystem that can be accessed directly by the WfW system.

Our layout is done using Quark Express on our WfW system. Once the ads are placed, the articles are put in. The result of the first layout process is a semi-complete magazine. A copy is printed locally on the Tektronix printer for review, and a PostScript image is written to a Linux filesystem so Michael can download it and print it in North Carolina. The locally-printed copy is printed from the WfW machine on a Linux-connected printer; this is much faster than directly connecting the printer to the WfW system, as was originally done.

The production editor merges the changes from Michael and the other reviewers and sends them back to layout for final changes. If the changes are extensive, the article may be re-edited as text using vi or emacs on a Linux system and re-submitted to layout.

After the second layout step, a paper copy of the magazine is printed for review by our proofreader. Changes from this cycle are made, and the layout system outputs a PostScript image of the magazine (in 2 to 8-page chunks) to a Linux filesystem connected to the imagesetter.

Samba

Although the layout system is running WfW and typesetting is running Linux, the file transfer is completely transparent to both departments, just like printing. This is made possible by Samba.

In the most basic sense, everything in the office uses TCP/IP as the underlying protocol. Windows for Workgroups speaks SMB (Session Message Block protocol), a protocol which uses TCP/IP. In order for Linux to speak SMB, we run Samba. Samba is a free implementation of the SMB protocol for Unix. It was developed by Andrew Tridgell in 1992 in an attempt to mount disk space from a Sun to a PC running Pathworks. [See *Linux Journal* Issue 7 for an account of Samba's development—ED] The Linux filesystem appears as network drives to WfW.

So Samba is used to seamlessly transfer the PostScript files from the WfW system to the Linux machine that drives the AGFA imagesetter. This is essentially a camera with a laser instead of a lens; it uses the laser to expose a large sheet of photographic film. There is one sheet of film for each black and white page and four sheets of film for each color page. This film is what is mailed to the printing company.

Once a magazine is finished, the files are archived on an Iomega Zip drive via Samba. The Zip drive is a removable 100 MB magnetic disk drive, much like a monster floppy, which is connected through a SCSI interface to the same Linux workstation that connects to the imagesetter. It connects to any SCSI-2 controller and acts like any other SCSI device. It is slightly slower than a typical hard drive but faster than some old MFM/RLL drives.

The ZIP disks are mounted on the Linux system like any other Linux filesystem. Linux views the new drive as a mounted partition of an existing drive. It can then be added to the Windows for Workgroups system, where the ZIP drive is seen as another network drive.

Other SSC Products

Besides *Linux Journal*, SSC publishes a series of books and references, primarily on Linux and Unix. Most of these products are done using troff and/or groff. The exceptions are the LDP (Linux Documentation Project) books, which are done in LaTeX, and the *Internet Public Access Guide*, which was done in Quark Express.

Again, we produce film directly at SSC to ship to the printer. One of the more interesting recent innovations was the ability to produce *spot color* separations using groff. This came about when we were updating the Korn Shell Reference.

This card is in four colors. Besides the additional cost of having the printer produce the separations it was going to be very hard to proof a multi-color card if only black and white output was available.

Work on the part of Arnold Robbins and SSC staff produced an easy way to write the color changes directly in the groff source. With two targets in the Makefile, one for printing the color output and one for producing the color-separated film, we were able to accomplish the desired task. These changes require groff and won't work with troff, since they are done by including raw PostScript commands in the groff source.

Many people may be asking why we continue to produce products using what many consider outdated tools. For those who really use the Unix environment, groff offers some advantages. For example, many of our recent products have been written by authors located far away from our Seattle offices. By using groff, we can send small ASCII files and use tools like diff to only send changes between locations. It also means we don't need multiple copies of expensive layout programs to accomplish the task.

Yes, We Use DOS Too

Our office manager uses QuickBooks to do our accounting. Why? Because it was available and inexpensive. She uses DOSemu on her Linux workstation for this task, allowing her to quickly switch back and forth between QuickBooks and, for example, her e-mail. In the future we hope to convert this task over to some software that runs on Linux, but for the moment, this offers a reasonable solution.

What's Next?

We have much planned for Linux in the future. For example, we currently do credit card verification off-line. We hope to write the necessary software to do this directly from a Linux system. We also hope that Progress will be ported to Linux. If this doesn't happen we will probably re-write our database system using a database that runs on Linux.

In conclusion, no, we don't use Linux for everything. But we come pretty close. Linux has proven inexpensive, easily expandable and reliable. For those who thought we didn't use Linux to produce the magazine, now you know. With all but one of our employees sitting at a Linux console every day, I think we practice what we preach rather well.

Resources

Some of the tools mentioned in this article may not have been included with your Linux distribution.

- NIS: Documentation is available from sunsite.unc.edu/mdw/HOWTO/NIS-HOWTO.html and binaries are available via ftp from sunsite.unc.edu in /pub/Linux/system/Network/admin/yp-clients.tar.gz.
- Orion: This firewall was available for purchase from Zebu Systems, http://www.zebu.com or (206)-781-9566 and is based on the Mazama Packet Filter software, www.mazama.com.
- **smap**: smap is part of the Trusted Information System's Firewall Toolkit at ftp://ftp.tis.com/pub/firewalls/toolkit/.
- **ssh**: ssh has a home page at www.cs.hut.fi/ssh/.
- Samba: Samba has a home page at lake.canberra.edu.au/pub/samba/samba.html.
- Apache: Apache has a home page at www.apache.org.
- xv: xv is available with some Linux distributions, and is available via ftp from ftp.cis.upenn.edu in /pub/xv/.

LaTeX, groff, sendmail, tar, diff, and other utilities mentioned in this article should be included with your Linux distribution.

Kevin Pierce is a Marketing Assistant at SSC. He grew up in New Hampshire, went to school at Florida State University, and reserves most Saturday afternoons for college football.

Archive Index Issue Table of Contents

Advanced search

Linux Journal Archives

LJ Staff

Issue #20, December 1995

We here at *Linux Journal* believe in empowerment, and giving people the tools to the job themselves, so we are proud to present the *LJ* Archives.

We get a lot of e-mail that begins “I am looking for that article, you know the one, about froodling the dewhacker?” We here at *Linux Journal* believe in empowerment, and giving people the tools to the job themselves, so we are proud to present the *LJ* Archives. You will find below a listing of the articles that we have published, and information on where to find them. Some are available on our Web site, and we have provided the URL for you. Others are not yet electronic, and if you can't wait, the issue number is listed so that you can order back issues. There is also a new feature on the Web site, an interactive search page for articles in *Linux Journal*. (www.linuxjournal.com/).

Features

- **Comparison of Linux, DOS/Win and OS/2** Issue: 1 Page: 1 Author: Bernie Thompson
- **Linux Code Freeze** Issue: 1 Page: 1 Author: Linus Torvalds
- **Interview With Linus** Issue: 1 Page: 4 Author: Robert Young
- **Formation of the XFree86 Project, Inc.** Issue: 2 Page: 1 Author: *LJ* Staff
- **Interview With Patrick Volkerding** Issue: 2 Page: 10 Author: Phil Hughes
- **Optimizing Linux Disk Usage** Issue: 2 Page: 1 Author: Jeff Tranter
- **World Wide Web** Issue: 3 Page: 9 Author: Bernie Thompson
- **Optimizing Memory Usage** Issue: 3 Page: 11 Author: Jeff Tranter
- **Sendmail+IDA** Issue: 3 Page: 30 Author: Vince Skahan
- **Interview with Fred Van Kempen** Issue: 3 Page: 16 Author: Phil Hughes
- **EZ as a Word Processor** Issue: 4 Page: 5 Author: Terry Gliedt
- **Disaster Recovery** Issue: 4 Page: 10 Author: Mark Komarinski

- **Wine** Issue: 4Page: 14 Author: Bob Amstadt
- **Eagles BBS** Issue: 4 Page: 16 Author: Ray Rocker
- **Linux Does Comics** Issue: 4 Page: 26 Author: Robert Suckling
- **Emacs, Friend or Foe?** Issue: 5 Page: 9 Author: Matt Welsh
- **Interview With James MacLean** Issue: 5 Page: 15 Author: Michael K. Johnson
- **EZ for the Programmer** Issue: 5 Page: 5 Author: Terry Gliedt
- **Linux in the Trenches** Issue: 5 Page: 21 Author: G. Wettstein
- **Messages, A Multi-Media Mailer** Issue: 6 Page: 7 Author: Terry Gliedt
- **Mobile Computing with Linux** Issue: 6 Page: 17 Author: Mark Fiuczynski
- **Learning C++ With Linux** Issue: 6 Page: 24 Author: Jeff Tranter
- **The Joy (and Agony) of SLIP** Issue: 6 Page: 30 Author: Warren Baird
- **Tutorial, Emacs for Programmers** Issue: 6 Page: 39 Author: Matt Welsh
- **Making the Most of Andrew** Issue: 7 Page: 7 Author: Terry Gliedt
- **Report from the Front: Linux in Antarctica** Issue: 7 Page: 11 Author: Andrew Tridgell
- **Samba: Unix Talking With PCs** Issue: 7 Page: 22 Author: Andrew Tridgell
- **Linux Performance Tuning** Issue: 7 Page: 26 Author: Clarence Smith
- **Introducing Modula-3** Issue: 8 Page: 9 Author: Geoff Wyant
- **X Window System Programming with Tcl and Tk** Issue: 8 Page: 24 Author: Matt Welsh
- **Linux Command Line Parameters** Issue: 8 Page: 35 Author: Jeff Tranter
- **What is a Linux?** Issue: 8 Page: 59 Author: *LJ* Staff
- **Linus Torvalds in Sydney** Issue: 8 Page: 60 Author: Jamie Honan
- **A Conversation with Linus Torvalds** Issue: 9 Page: 8 Author: Belinda Frazier
- **Connecting Your Linux Box to the Internet** Issue: 9 Page: 18 Author: Russell Ochocki
- **Linux in the Real World: Redesigning SCADA at Virginia Power** Issue: 9 Page: 23 Author: Vance Petree
- **Remote Network Commands** Issue: 9 Page: 33 Author: Jens Hartmann
- **A Conversation with Olaf Kirch** Issue: 10 Page: 13 Author: *LJ* Staff
- **Linux in the Real World: SCADA-Linux Still Hard at Work** Issue: 10 Page: 14 Author: Vince Petree
- **Linux Conference at Open Systems World FedUNIX'94** Issue: 10 Page: 20 Author: Belinda Frazier
- **Using Tcl and Tk From Your C Programs** Issue: 10 Page: 26 Author: Matt Welsh
- **Linux in Amsterdam** Issue: 10 Page: 6 Author: Michael K. Johnson

- **The Humble Beginnings of Linux** Issue: 11 Page: 11 Author: Randolph Bentson
- **Introducing Scheme** Issue: 11 Page: 23 Author: Robert Sanders
- **Introduction to LINCKS** Issue: 11 Page: 26 Author: Martin Sjolin
- **Review of Scilab** Issue: 11 Page: 40 Author: Robert Dalrymple
- **Building Shared Libraries** Issue: 12 Page: 9 Author: Eric Kasten
- **Ethernetting Linux** Issue: 12 Page: 12 Author: Terry Dawson
- **Linux- It's Not Just For Intel Anymore** Issue: 12 Page: 20 Author: Joseph Brothers
- **Leviathan** Issue: 12 Page: 28 Author: Paul Sittler
- **The Pari Package on Linux** Issue: 13 Page: 5 PeterAuthor: Klaus-N/A Nischke
- **Access Information Through World Wide Web: Installing CERN's WWW Server** Issue: 13 Page: 9 Author: Eric Kasten
- **Majordomo** Issue: 13 Page: 13 Author: Piers Cawley
- **Interview with Mark Bolzern** Issue: 14 Page: 19 Author: *LJ* Staff
- **Review: xBase Products for Linux** Issue: 14 Page: 22 Author: Robert Broughton
- **Introduction to Eiffel** Issue: 14 Page: 34 Author: Dan Wilder
- **Review: Intelligent Multiport Serial Boards** Issue: 14 Page: 46 Author: Greg Hankins
- **The LINCKS GPD** Issue: 15 Page: 17 Author: Martin Sjolin
- **Setting up X11** Issue: 15 Page: 24 Author: Greg Lehey
- **HTML: A Gentle Introduction** Issue: 15 Page: 35 Author: Eric Kasten
- **xfm 1.3: A File and Applications Manager** Issue: 15 Page: 50 Author: Robert Dalrymple
- **Linux Goes To Sea** Issue: 16 Page: 19 Author: Randolph Bentson
- **Efficient, User-Friendly Seismology** Issue: 16 Page: 25 Author: Sid Hellman
- **HTML Forms: Interacting with the Net** Issue: 16 Page: 34 Author: Eric Kasten
- **Introduction to Lisp-Stat** Issue: 16 Page: 44 Author: Narasimhan Balasubramanian
- **Writing A Mouse Sensitive Application** Issue: 17 Page: 14 Author: Alessandro Rubini
- **Porting DOS Applications to Linux** Issue: 17 Page: 28 Author: Alan Cox
- **ncurses: Portable Screen Handling for Linux** Issue: 17 Page: 43 Author: Eric Raymond
- **Writing man Pages Using groff** Issue: 18 Page: 18 Author: Matt Welsh
- **LaTex for the Slightly Timid** Issue: 18 Page: 34 Author: Kim Johnson

- **Using grep** Issue: 18 Page: 42 Author: Eric Goedelbacker
- **Flexible Formatting with Linuxdoc-SGML** Issue: 18 Page: 51 Author: Christian Schwarz
- **Getting the Most Out Of X Resources** Issue: 19 Page: 30 Author: Preston Brown
- **Optimizing the Linux User Interface** Issue: 19 Page: 47 Author: Jeff Arnholt
- **LesTif and the Hungry ViewKit** Issue: 19 Page: 56 Author: Malcolm Murphy

News and Articles

- **ICMAKE part 1** Issue: 1 Page: 12 Author: Frank Brokken
- **Onyx** Issue: 1Page: 14 Author: Micheal Kraehe
- **Linux and Hams** Issue: 1 Page: 16 Author: Phil Hughes
- **The Linux FSSTD** Issue: 2 Page: 28 Author: Daniel Quinlan
- **ICMAKE part 2** Issue: 2 Page: 34 Author: Frank Brokken
- **WINE status** Issue: 2 Page: 44 Author: Bob Amstadt
- **Introduction to the GNU C Library** Issue: 2 Page: 18 Author: Michael K. Johnson
- **Linux Distributions** Issue: 2 Page: 22 Author: *LJ* Staff
- **Let's Take Linux Seriously** Issue: 3 Page: 7 Author: Phil Hughes
- **ICMAKE part 3** Issue: 3 Page: 24 Author: Frank Brokken
- **The Open Development of Debian** Issue: 3 Page: 29 Author: Ian Murdock
- **UniForum 1994** Issue: 3 Page: 18 Author: Phil Hughes
- **From The Publisher: Let's Take Linux Seriously** Issue: 3 Page: 7 Author: Phil Hughes
- **Linus Torvalds at DECUS '94** Issue: 4 Page: 20 Author: Bob Tadlock
- **Unix and Computer Science** Issue: 4 Page: 22 Author: Ronda Hauben
- **Linux Sound Support** Issue: 4 Page: 28 Author: Jeff Tranter
- **ICMAKE part 4** Issue: 4 Page: 35 Author: Frank Brokken
- **Slackware 2.0 Released** Issue: 4 Page: 45 Author: Phil Hughes
- **Linux on the Motorola 860x0** Issue: 5 Page: 20 Author: Hamish Macdonald
- **Dialog, An Introductory Tutorial** Issue: 5 Page: 24 Author: Jeff Tranter
- **Writing an Intelligent Serial Driver** Issue: 5 Page: 28 Author: Randolph Bentson
- **Using iBCS2 under Linux** Issue: 5 Page: 30 Author: Eric Youngdale
- **Linux Events: Two Views on Heidelberg** Issue: 3 Page: 32 Author: *LJ* Staff
- **Selecting a Linux CD** Issue: 6 Page: 20 Author: Phil Hughes
- **Fix /etc/gateway** Issue: 6 Page: 21 Author: Cor Bosman

- **Report From the Front:The Linux Review Group** Issue: 6 Page: 23 Author: Magnus Alvestad
- ***Linux Journal* Demographics** Issue: 6 Page: 32 Author: Laurie Tucker
- **Kernel Code Freeze Announced** Issue: 6 Page: 37 Author: Linus Torvalds
- **Harbor** Issue: 6 Page: 46 Author: Michael K. Johnson
- **Overview of the Debian GNU/Linux System** Issue: 6 Page: 59 Author: Ian Murdock
- **Selecting Hardware for a Linux System** Issue: 7 Page: 14 Author: Phil Hughes
- **CD-ROMs and Linux** Issue: 7 Page: 33 Author: Jeff Tranter
- **Linux User Group News** Issue: 7 Page: 39 Author: *LJ* Staff
- **Andy** Issue: 7Page: 40 Author: Andy Tefft
- **The Term Protocol** Issue: 8 Page: 39 Author: Liem Bahneman
- **Linux Meta-FAQ Version 3.11** Issue: 8 Page: 48 Author: various
- **Linux Organizations** Issue: 8 Page: 54 Author: Michael K. Johnson
- **Linux Development Grant Fund** Issue: 9 Page: 42 Author: *LJ* Staff
- **Report on Comdex '94** Issue: 10 Page: 6 Author: Belinda Frazier
- **What Your DOS Manual Doesn't Tell You About Linux** Issue: 10 Page: 23 Author: Liam Greenwood
- **Questions From the *Linux Journal* Booth at Open Systems World** Issue: 11 Page: 20 Author: Kim Johnson
- **Linux in the Real World—Data Gathering With Linux** Issue: 11 Page: 18 Author: Grant Edwards
- **Pentiums & Non-Pentiums** Issue: 11 Page: 34 Author: Phil Hughes
- **Installing Linux via NFS** Issue: 11 Page: 15 Author: Greg Hankins
- **Mr Torvalds Goes to Washington** Issue: 12 Page: 5 Author: Kurt Reisler
- **Linux Tip** Issue: 12 Page: 41 Author: *LJ* Staff
- **Users Mounting Floppies** Issue: 12 Page: 41 Author: *LJ* Staff
- **Linux For Public Service** Issue: 13 Page: 21 Author: Dan Hollis
- **Hamming It Up On Linux** Issue: 13 Page: 26 Author: Brian Lantz
- **Netsurfing With Linux: To Sail The Cyber Sea** Issue: 13 Page: 30 Author: Arthur Bebak
- **A New Project or a GNU Project** Issue: 13 Page: 44 Author: Mark Bolzern
- **Linus Torvalds Receives Award** Issue: 13 Page: 60 Author: *LJ* Staff
- **Linux at the UW Computer Fair** Issue: 14 Page: 17 Author: *LJ* Staff
- **Caldera and Corsair** Issue: 14 Page: 18 Author: *LJ* Staff
- **Linux at Comdex/Fall: A Call for Participation** Issue: 14 Page: 24 Author: Mark Bolzern

- **Product Review: SlickEdit** Issue: 14 Page: 41 Author: Jeff Bauer
- **The Linux File System Standard** Issue: 15 Page: 45 Author: Garrett D'Amore
- **The Trade Shows** Issue: 16 Page: 14 Author: Randolph Bentson
- **Putting Widgets in Their Place** Issue: 16 Page: 30 Author: Stephen Uhler
- **Prototyping Algorithms in Perl** Issue: 16 Page: 39 Author: Jim Shapiro
- **Two Eiffel Implementations** Issue: 17 Page: 21 Author: Dan Wilder
- **Reader Survey Results** Issue: 17 Page: 36 Author: *LJ* Staff
- **Indexing with Glimpse** Issue: 18 Page: 16 Author: Michael K. Johnson
- **Linux on Alpha: A Strategic Choice** Issue: 18 Page: 29 Author: Jon “maddog” Hall
- **Linux Serving IKEA** Issue: 19 Page: 20 Author: Anders Osting
- **The Best Without X** Issue: 19 Page: 22 Author: Alessandro Rubini
- **How To Build A Mac** Issue: 19 Page: 40 Author: Andreas Schiffler
- **Linux on Low-End Hardware** Issue: 19 Page: 38 Author: Trenton B. Tuggle
- **Linux at SCO Forum** Issue: 19 Page: 59 Author: Belinda Frazier

Kernel Korner

- **Introduction (Device Drivers)** Issue: 8 Page: 18 Author: Michael K. Johnson
- **Block Device Drivers** Issue: 9 Page: 11 Author: Michael K. Johnson
- **Block Device Drivers: Interrupts** Issue: 10 Page: 9 Author: Michael K. Johnson
- **Block Device Drivers- Optimization** Issue: 11 Page: 38 Author: Michael K. Johnson
- **The ELF Object File Format** Issue: 12 Page: 14 Author: Eric Youngdale
- **The ELF Object File Format by Dissection** Issue: 13 Page: 27 Author: Eric Youngdale
- **The Linux Keyboard Driver** Issue: 14 Page: 5 Author: Andries Brouwer
- **Memory Allocation** Issue: 16 Page: 16 Author: Michael K. Johnson
- **System Calls** Issue: 17 Page: 12 Author: Michael K. Johnson
- **Porting Linux to the DEC Alpha: Infrastructure** Issue: 18 Page: 22 Author: Jim Paradis
- **Porting Linux To the DEC Alpha** Issue: 19 Page: 16 Author: Jim Paradis

Programming Hints

- **Programming the VT Interface** Issue: 3 Page: 35 Author: Michael K. Johnson

- **Programming the VT Interface part 2** Issue: 4 Page: 31 Author: Michael K. Johnson
- **Strange I/O** Issue: 5 Page: 33 Author: Michael K. Johnson
- **Introduction to make** Issue: 6 Page: 48 Author: Michael K. Johnson

System Administration

- **The df command** Issue: 1 Page: 35 Author: Phil Hughes
- **General System Administration** Issue: 2 Page: 26 Author: Mark Komarinski
- **Making Your Own Filesystems** Issue: 3 Page: 20 Author: Mark Komarinski
- **mtools** Issue: 5 Page: 17 Author: Mark Komarinski
- **Fixing Your Clock** Issue: 8 Page: 15 Author: Mark Komarinski
- **How to Move /home To A New Hard Drive On Your Linux System** Issue: 8 Page: 44 Author: *LJ* Staff
- **Undelete** Issue: 9 Page: 14 Author: Mark Komarinski
- **How To Log Friends and Influence People** Issue: 11 Page: 35 Author: Mark Komarinski
- **Setting Up Services** Issue: 12 Page: 18 Author: Mark Komarinski
- **Anonymous ftp** Issue: 13 Page: 41 Author: Mark Komarinski
- **Upgrading the Linux Kernel** Issue: 14 Page: 30 Author: Mark Komarinski
- **Installing the Xaw3D Libraries** Issue: 15 Page: 54 Author: Mark Komarinski
- **Using LILO** Issue: 19 Page: 52 Author: Æleen Frisch

Novice to Novice

- **Linux Installation and X Windows** Issue: 12 Page: 26 Author: Dean Oisboid
- **DOS Emulation with dosemu** Issue: 13 Page: 34 Author: Dean Oisboid
- **Games, Sound & Other Agonies** Issue: 15 Page: 6 Author: Dean Oisboid
- **Interlude & Exploration: Spreadsheets & Text Editors** Issue: 16 Page: 6 Author: Dean Oisboid
- **Databases** Issue: 17 Page: 37 Author: Dean Oisboid
- **Serendipity** Issue: 18 Page: 26 Author: Dean Oisboid

What's Gnu?

- **What's Gnu?** Issue: 1 Page: 20 Author: Arnold Robbins
- **What's Gnu?** Issue: 2 Page: 14 Author: Arnold Robbins
- **Bash: The GNU shell** Issue: 3 Page: 40 Author: Chet Ramey
- **Bash: The GNU Shell part 2** Issue: 4 Page: 41 Author: Chet Ramey
- **Texinfo** Issue: 6 Page: 51 Author: Arnold Robbins

- **groff** Issue: 7Page: 19 Author: Arnold Robbins
- **RCS: Revision Control System** Issue: 10 Page: 36 Author: Arnold Robbins
- **Plan 9** Issue: 11 Page: 31 Author: Arnold Robbins
- **The GNU Coding Standards** Issue: 16 Page: 47 Author: Arnold Robbins

Stop the Presses

- **Non-Intel Linux, WWW, Brave New Linux** Issue: 3 Page: 6 Author: Michael K. Johnson
- **Linux MIPS, Version 0.9.2 Linux 64k, LSM Maintainer Change** Issue: 6 Page: 5 Author: N/A
- **Linux Gaining Momentum** Issue: 7 Page: 5 Author: Phil Hughes
- ***Linux Journal* at Unix Expo** Issue: 8 Page: 5 Author: Phil Hughes
- **An Amazing Year—Looking Into the Future** Issue: 9 Page: 7 Author: Phil Hughes
- **Documentation?** Issue: 10 Page: 5 Author: Phil Hughes
- **Time Again For Reader Input** Issue: 11 Page: 10 Author: *LJ* Staff
- **Linus Torvalds Releases Linux 2.0** Issue: 13 Page: 22 Author: *LJ* Staff
- **Changes to *Linux Journal*** Issue: 14 Page: 10 Author: Phil Hughes
- **Linux at DECUS** Issue: 15 Page: 10 Author: Michael K. Johnson
- **ELF Tools Released for Linux** Issue: 16 Page: 10 Author: Michael K. Johnson
- **WYSIWYG Editor, BLADE, Linux at OSW** Issue: 17 Page: 10 Author: Phil Hughes
- **Win95 and Linux** Issue: 18 Page: 11 Author: Phil Hughes

Cooking With Linux

- **Virtual Dramamine** Issue: 3 Page: 46 Author: Matt Welsh
- **It's Linux Jim, But Not As We Know It!** Issue: 4 Page: 39 Author: Matt Welsh
- **Thou Shalt Not Use MS-DOS** Issue: 5 Page: 35 Author: Matt Welsh
- **Your Mileage May Vary** Issue: 6 Page: 38 Author: Matt Welsh
- **Amsterdam on Fifty Guilders a Day** Issue: 12 Page: 11 Author: Matt Welsh

Book Reviews

- **Linux Installation and Getting Started** Issue: 1 Page: 10 Author: Phil Hughes
- **Newtons Telecom Dictionary** Issue: 3 Page: 23 Author: Phil Hughes
- **Internet Public Access Guide** Issue: 3 Page: 23 Author: Morgan Hall
- **Cyberia, Life in the Trenches of Hyperspace** Issue: 5 Page: 44 Author: Putnam Barber

- **The Whole Internet User's Guide and Catalog** Issue: 5 Page: 44 Author: Putnam Barber
- **Firewall and Internet Security: Repelling the Wily Hacker** Issue: 6 Page: 36 Author: Danny Yee
- **Making TeX Work** Issue: 8 Page: 50 Author: Vince Skahan
- **UNIX—An Open Systems Dictionary** Issue: 8 Page: 52 Author: Laurie Tucker
- **Linux vom PC zur Workstation Grundlagen, Installation und praktischer Einsatz** Issue: 8 Page: 52 Author: Martin Sckopke
- **Linux Anwenderhandbuch Leitfaden fur die Systemverwaltung** Issue: 8 Page: 53 Author: Martin Sckopke
- **Unix Systems for Modern Architectures** Issue: 9 Page: 30 Author: Randolph Bentson
- **Your Internet Consultant** Issue: 11 Page: 59 Author: Phil Hughes
- **The Tcl and the Tk Toolkit** Issue: 11 Page: 60 Author: Phil Hughes
- **A Quarter Century of Unix** Issue: 12 Page: 45 Author: Danny Yee
- **The X Mosaic Handbook for the X Window System** Issue: 12 Page: 25 Author: Caleb Epstein
- **The Linux Sampler** Issue: 13 Page: 58 Author: Harvey Friedman
- **Running Linux** Issue: 14 Page: 8 Author: Grant Johnson
- **X User Tools** Issue: 15 Page: 8 Author: Danny Yee
- **Sendmail: Theory and Practice** Issue: 16 Page: 12 Author: Phil Hughes
- **Casting the Net: From ARPANET to Internet and Beyond** Issue: 17 Page: 48 Author: Danny Yee
- **The Unix Philosophy** Issue: 17 Page: 51 Author: Belinda Frazier
- **Build a Web Site** Issue: 18 Page: 15 Author: Brian Rice
- **The HTML Sourcebook** Issue: 18 Page: 15 Author: Brian Rice
- **HTML for Fun and Profit** Issue: 18 Page: 15 Author: Brian Rice
- **Teach Yourself Perl in 21 Days** Issue: 19 Page: 15 Author: David Flood

Product Reviews

- **Motif 1.2.3 Runtime and Development System** Issue: 6 Page: 12 Author: Dale A. Lutz
- **Unix Interactive Tools** Issue: 6 Page: 22 Author: Clarence Smith
- **Crisp Text Editor** Issue: 6 Page: 27 Author: Robert Broughton
- **Doom** Issue: 8Page: 22 Author: Michael K. Johnson
- **BRU—Backup and Restore Utility** Issue: 11 Page: 43 Author: Jon Freivald
- **Xfig** Issue: 12Page: 6 Author: Robert Dalrymple
- **InfoMagic** Issue: 12 Page: 25 Author: Caleb Epstein

- **Metro X** Issue: 15 Page: 30 Author: Bogdan Urma
- **AX Graphical Display Server** Issue: 15 Page: 31 Author: Mark Ganter
- **Motif for Linux** Issue: 15 Page: 48 Bogdan Urma
- **Moo-Tiff Development Environment** Issue: 17 Page: 55 Author: Bogden Urma
- **IGEL Etherminal 3X** Issue: 19 Page: 35 Author: Michael K. Johnson

Archive Index Issue Table of Contents

Advanced search

Linux Journal Readers' Choice Awards

LJ Staff

Issue #20, December 1995

A few months ago, we asked *Linux Journal* readers to rank their favorite Linux-related software, hardware and books during a two-week Web and e-mail-based survey. Here are your choices.

This first year, the Readers' Choice awards have only three broad catagories. One of the most common requests was for us to expand our catagories significantly, which we will consider for next year's awards.

Books

Tied for first place were *Running Linux*, by Matt Welsh and Lar Kaufman, and *Sendmail: Theory and Practice*, by Frederick M. Avolio and Paul A. Vixie. A close second was *Tcl and the Tk Toolkit*, by John Ousterhout.

Running Linux has sold out of several printings, and O'Reilly has announced that they will be selling it with a companion CDROM package containing Red Hat Commercial Linux.

Hardware

First place in the hardware catagory was the Cyclades family of multiport serial boards, and second place was the Comtrol family of multiport serial boards.

Both of these vendors fully support the Linux drivers for their products, and the Cyclades driver is part of the mainline Linux source tree. In addition, when Cyclades released their first PCI-based multiport serial board, they prepared the Linux driver before drivers for any other operating system.

Software

First place goes to **Ishmail**, a powerful mail-reading application for Linux. Tied for second were **BB Tool**, a stock charting application, and *BRU*, the Backup and Restore Utility. Many readers wanted to vote for more than one subcatagory of software; they considered choosing between an application and a tool (for instance) impossible and insane. (We would like to thank them for doing the impossible and becoming temporarily insane for us...)

While there were obviously many readers who like Ishmail, we suspect that one of them posted to an Ishmail mailing list about the survey, since the majority of the votes for Ishmail came in over a period of only a few hours. Even without those votes, Ishmail would still have won, as well as we can tell.

Archive Index Issue Table of Contents

Advanced search

Linux System Administration

Æleen Frisch

Issue #20, December 1995

Slightly more tedious and complex than adding a disk drive to other microcomputer systems.

The steps required to add an additional disk drive to a microcomputer system are somewhat more tedious than those needed for larger systems. Most of the complexity comes from the fact that disks can be shared by distinct operating systems on microcomputers.

The terminology related to disk partitions varies somewhat between UNIX and other microcomputer operating systems. For example, DOS distinguishes between the primary partition (the principal, bootable DOS partition) and other extended partitions on the same hard disk; a UNIX disk partition is simply a separately accessible portion of a disk. DOS allows for a maximum of four physical partitions per disk. Under DOS, a physical disk partition can be further subdivided into multiple parts, known as logical drives. The first step in adding a disk to a microcomputer system is to decide how the drive will be split between DOS and UNIX (if applicable). The fdisk utility is used to create physical disk partitions on microcomputer systems (DOS also provides an fdisk utility). The cfdisk utility, a screen-based version of fdisk, is also available under Linux. The following considerations apply to the myriad of fdisk versions that you may encounter:

- The conventional wisdom is to use the native version of fdisk to create and operate on the disk partitions for each operating system. In other words, use the DOS version for the DOS partitions, and the UNIX version for the UNIX partitions. In practice, you can often get away with using a different version. Things generally work fine, except when they don't.
- Keep records of the partition numbers, starting and ending blocks, total sizes, partition type, and other data for each disk partition table as it is displayed by every version of fdisk that you have. An easy way to do this is to print the partition table from each version. If the table is ever damaged

—and this does happen from time to time—you will probably need the information to recreate it and recover your data. Having the data from all the versions ensures that you can redefine partitions following the same alignment patterns and requirements as observed by the various operating systems when they created the partitions initially.

Making a New Disk Available to Linux

We'll look at the process of attaching a new SCSI disk to a Linux system in detail. The process would be the same for other disk types (for example, IDE), although the special files used to access the device would be different (for example, /dev/hdb).

After attaching the disk to the system, it should be detected when the system is booted. You can use the dmesg command to display boot messages or check /etc/boot.log if you're using the sysvinit facility:

```
scsi0 : at 0x0388 irq 10 options CAN_QUEUE=32 ...
scsi0 : Pro Audio Spectrum-16 SCSI
scsi : 1 host.
Detected scsi disk sda at scsi0, id 2, lun 0
scsi : detected 1 SCSI disk total.
```

If necessary, create the device special files for the disk (needed only when you have lots of disks). For example, this command creates the special files used to access the fifth SCSI disk:

```
# cd /dev; MAKEDEV sde
```

Note also that disk ordering happens at boot time, so adding a new SCSI disk with a lower SCSI ID than an existing disk will cause special files to be reassigned—and probably break your /etc/fstab setup.

Assuming we have our special files all in order, we will use fdisk or cfdisk (a screen-oriented version) to divide the disk into partitions (we'll be saving about a third of the disk for a DOS partition). The following commands will start these utilities for the first SCSI disk:

```
# fdisk /dev/sda
# cfdisk /dev/sda
```

We'll need the following subcommands:

Action	Subcommand	
	fdisk	cfdisk

Create new partition	n	n
Change partition type	t	t
Make partition active/bootable	a	b
Write partition table to disk	w	W
Change size units	u	u
Display partition table	p	N/A

cfdisk is more convenient to use as the partition table is displayed continuously. A cfdisk subcommand always operates on the current partition (highlighted). Thus, in order to create a new partition, move the highlight to the line corresponding to Free Space, and then enter an n. cfdisk will prompt for the partition information:

```
Primary or logical [pl]: p
Size (in MB): 110
```

If you'd rather enter the size in a different set of units, use the u subcommand (cycles among MB, sectors and cylinders).

We use the same procedure to create a second partition, and then activate the first partition with the b subcommand. Then, we use the t subcommand to change the partition types of the two partitions. The most commonly needed type codes are 6 for DOS, 82 for a Linux swap partition, and 83 for a regular Linux partition.

Here is the final partition table (output has been simplified): Don't put any hairspaces in between the dashes below, or they will blow up. They don't have to look separated.

```
cfdisk 0.8 BETA
              Disk Drive: /dev/sda
Heads: 16  Sectors per Track: 63   Cylinders: 1023
Name      Flags   Part Type   FS Type    Size (MB)
/dev/sda1 Boot    Primary     Linux          110.0
/dev/sda2         Primary     DOS             52.5
                  Pri/Log     Free Space       0.5
```

If you've changed the partition layout of the disk—in other words, done anything other than change the types assigned to the various partitions—reboot the system at this point.

Next, use the mkfs command to create a filesystem on the Linux partition. mkfs has been streamlined in the Linux version and requires little input:

```
# mkfs -t ext2 /dev/sda1
```

This command creates an ext2-type filesystem. If you want to customize mkfs's operation, the following options may be useful:

- **-b**: Set filesystem block size in bytes (the default is 1024).
- **-f**: Set filesystem fragment size in bytes (the default is 1024).
- **-c**: Check the disk partition for bad blocks before making the filesystem.
- **-i**: Specify bytes/inode value: create one inode for each chunk of this many bytes. The default value of 4096 usually creates more than you'll ever need, but probably isn't worth changing.
- **-m**: Specify the percentage of filesystem space to reserve (accessible only by root). The default is 5% (half of what is typical on other UNIX systems). In these days of multigigabyte disks, even this percentage may be worth rethinking.

Once the filesystem is built, run fsck:

```
# fsck -f -y /dev/sda1
```

The **-f** option is necessary to force fsck to run even though the filesystem is clean. The new filesystem may now be mounted and entered into /etc/fstab, which is the subject of the next section.

The Filesystem Configuration File: /etc/fstab

Here are some sample entries from /etc/fstab from a Linux system:

```
# device   mount  type  options   dump fsck
/dev/hda2  /      ext2  defaults    1    1
/dev/hdb1  /aux   msdos noauto      1    2
/dev/hda1  none   swap  sw          0    0
/dev/sda1  /chem  ext2  defaults    1    1
```

The general format for an entry is:

```
special-file loc type opts
dump-freq pass-number
```

The fields have the following meanings:

- ***special-file***: The name of the special file on which the filesystem resides. This must be a block device name.
- ***loc***: The directory at which to mount the filesystem. If the partition will be used for swapping, use **none** for this field.

- ***type***: The kind of partition the entry refers to. The value for local filesystems under Linux is **ext2**. Other common type values are **nfs** for volumes mounted remotely via NFS, and swap for swap partitions and ignore, which tells mount to ignore the entry.
- ***opts***: This field consists of one or more options, separated by commas. The type field, above, determines which options are allowed for any given kind of filesystem. For ignore type entries, this field is ignored. For local filesystems, the options field may include the following keywords, separated by commas:

rw	Read-write filesystem
ro	Read-only filesystem
suid	The SUID access mode is permitted (default)
nosuid	The SUID access mode is not permitted
noauto	Don't automatically mount this filesystem
usrquota	User quotas may be placed in effect
grpquota	Group quotas may be placed in effect

- Multiple options are separated by commas, without intervening spaces. On many systems, the keyword defaults may be placed into this field if no options are needed.
- If the filesystem type is nfs, many more options are supported (see Chapter 13).
- ***dump-freq***: A decimal number indicating the frequency with which this filesystem should be backed up by the dump utility. The dump utility is in alpha testing and is not available on most Linux systems, so unless you use dump, you can ignore this field.
- ***pass-number***: A decimal number indicating the order in which fsck should check the filesystems. A pass-number of 1 indicates that the filesystem should be checked first, 2 indicates that the filesystem should be checked second, and so on. The root filesystem must have a pass-number of 1. All other filesystems should have the same or higher pass numbers. For optimal performance, two filesystems that are on the same disk drive should have different pass numbers; however, filesystems on different drives may have the same pass number, letting fsck check the

two filesystems in parallel. fsck will usually be fastest if all filesystems checked on the same pass have roughly the same size. This field should be 0 for swap devices (0 disables checking by fsck).

Viewing and Modifying the Superblock

The tune2fs command may be used to list and alter fields within the superblock of an ext2 filesystem. Here is an example of its display-mode output:

```
# tune2fs -l /dev/hdb1
Filesystem magic number: 0xEF53
Filesystem state:        clean
Errors behavior:         Continue
Inode count:             13104
Block count:             52208
Reserved block count:    2610
Free blocks:             50528
Free inodes:             13093
First block:             1
Block size:              1024
Fragment size:           1024
Blocks per group:        8192
Fragments per group:     8192
Inodes per group:        1872
Last mount time:         Wed Dec 31 19:00:00 1969
Last write time:         Thu Mar  2 04:19:16 1995
Mount count:             6
Maximum mount count:     20
Last checked:            Thu Mar  7 15:27:34 1996
Check interval:          2592000
Next check after:        Fri Apr  5 16:27:34 1996
```

The final items in the list concern when fsck will check the filesystem, even if it is clean. The Linux version of fsck for ext2 filesystems will check the filesystem if either the maximum number of mounts without a check has been exceeded or the maximum time interval between checks has expired (20 times and 30 days in the preceding output; the check interval is given in seconds).

tune2fs's **-i** option may be used to specify the maximum time interval between checks in days, and the **-c** option may be used to specify the maximum number of mounts between checks. For example, the following command disables the time-between-checks function and sets the maximum number of mounts to 25:

```
# tune2fs -i 0 -c 25 /dev/hdb1
Setting maximal mount count to 25
Setting interval between check 0 seconds
```

Another useful option to tune2fs is **-m**, which allows you to change the percentage of filesystem space held in reserve dynamically.

Hints for Splitting Linux Across Two Disks

UNIX versions designed for microcomputers tend to assume that such systems have a single disk large enough to accommodate all of the filesystems that it will use. If what you actually have is a smaller amount of space on each of two disks but not enough on either one to hold all of UNIX, there is usually no built-

in way to install the operating system anyway. However, a procedure like the following will usually be successful:

- Install a minimal operating system on the partition on the first disk.
- Add the partition(s) from the second disk to the system.
- The general strategy is to create symbolic links to the partition on the second disk to allow the operating system to be split across them. This can mean copying some directories to the second disk after installation and then creating links in the original location, as in this example (/d2 is the mount point for the partition from the second disk:

```
# cd /d2
# tar -cvf - -C /usr/lib terminfo | tar -xvpf -
# rm -rf /usr/lib/terminfo
# ln -s /d2/terminfo /usr/lib/terminfo
```

- Alternatively, if you know or can determine the location for an operating system component before installing it, you can pre-set up the symbolic link, then install that component, and the files will be written to the right location to begin with. For example, the following commands will cause the manual pages to be written to the second disk:

```
# mkdir /d2/man
# chown bin /d2/man; chgrp bin /d2/man
# chmod 755 /d2/man
# ln -s /d2/man /usr/man
```

- When selecting components to move, avoid placing anything required to boot the system on the second disk.
- Continue this process until everything you want is installed.

Æleen Frisch manages a very heterogeneous network of Linux and other UNIX systems and PCs. After finally finishing the second edition of *Essential System Administration*, she has gone back to her true calling in life, pulling the string for her cats, Daphne and Sarah. She can be reached via e-mail at aefrisch@lorentzian.com.

Archive Index Issue Table of Contents

Advanced search

PracTcl Programming Tips

Stephen Uhler

Issue #20, December 1995

This month, find out how to test the speed of your Tcl programs and zero in on the slow parts with the time command.

The other day I wrote a Tcl program, and it was really slow. Even after I got all of the bugs out, it was still slow, so it was time to optimize its performance. My first rule of program optimization is to do nothing; just wait for a faster machine. Unfortunately, in this case, I can't wait that long.

My second rule of optimization is to avoid spending any effort optimizing code that doesn't matter, either because it never gets run (don't laugh, it happens more than you think), or it doesn't take a significant fraction of the total run time of the program.

As an aid to program optimization, Tcl provides a **time** command that measures and reports the time it takes to execute a Tcl script. It's called as:

```
time script count
```

where ***script*** is the Tcl script to time, and ***count*** is the number of times to run the script, it's run once if no count is given. **time** reports its result in the form:

```
3000 microseconds per iteration
```

The time is the average elapsed time for each time the script is timed.

Occasionally, it is useful to use the **time** command as-is to time procedures as they are run on various inputs. Once you have pin-pointed the “time hog” procedure in your application, **time** can be used this way to measure the performance effects of recoding techniques or new algorithms (or faster machines).

However, for any serious program optimization, a more systematic approach is needed. We can write some procedures that use the **time** command to enable us to compute an execution time profile of an entire application.

One way to profile an application would be to re-write every procedure, surrounding its body with a **time** command, whose result can be tabulated in some way. Since Tcl lets us ask about the arguments and body of every procedure, the procedures can be rewritten automatically by the profiler.

The disadvantage of this approach is that it only works for procedures and not for the built-in commands. Considering that one of the primary ways of speeding up a Tcl application is recoding some of the procedures in C and turning them into built-in commands, it would be nice to be able to profile commands as well as procedures.

Instead of re-writing procedures, we'll rename them and write a new procedure with the original name that calls (and *times*) the renamed version. This way, profiling can be done on both commands and procedures.

Gathering the Data

The first step is to write the replacement procedure. It will be the same for all procedures and commands except for its name, and the name of the renamed original it will call. The easiest way to accomplish this is by writing a procedure body template and using the **format** command to substitute in the appropriate names.

```
proc profile_proc {proc} {
   global Template
   rename $proc $proc:OLD
   set body [format $Template $proc]
   proc $proc args $body
}
```

The procedure **profile_proc** takes the name of a Tcl procedure, renames it by adding the suffix **:OLD**, then uses the template (in **Template**) to write a new procedure that times the calls to the original. Although different commands and procedures will require differing numbers of arguments, by using **args**, we can arrange for them to be passed to the original (**:OLD**) procedure un-changed.

The global variable **Template** contains the **format** template for the new procedure body, which substitutes the procedure name in two places, one to call the original and another to log the results. The **1$** in the format specifier indicates the first parameter to format should be used both times.

```
set Template {
    global Profile_fd
```

```
    set time [lindex [time {set result \
            [uplevel [list %1$s:OLD] $args]}] 0]
    set level [expr [info level] - 1]
    if {$level > 0} {
        set caller [lindex [info level $level] 0]
        regsub -all {(.*):OLD} $caller {\1} \
        caller
    } else {
        set caller TopLevel
    }
    puts $Profile_fd [list %1$s $time $caller]
    return $result
}
```

The timing information is written to a file so that it may be analyzed off-line. We will make sure the variable **Profile_fd** contains a file handle to which we can write timing information. First, we run the "original" command with **time**. By using **uplevel**, we can insure the original command will run in the same stack level it expects, so if it uses any **upvar** commands they will work properly. Since nothing in Tcl prevents a procedure name from having special characters in it, the **%s:OLD** needs to be passed though the **list** command. The original arguments to the procedure, that were gathered up into the single list **args**, are expanded into their original form by uplevel. The **lindex 0** extracts just the time portion of the **time** command output and saves it in the variable **time**.

To properly account for the time spent in each procedure, not only is the timed procedure's name required, but its caller's name is needed as well, as will be apparent when it comes time to account for all of the time.

The **info level** command is used to determine the caller's name (or **TopLevel**, if called from the top level). The **regsub** gets rid of the **:OLD** to make the book-keeping easier.

Finally, the procedure name, time spent, and caller name are all written to the logging file, and the result returned by the renamed procedure, **result**, is returned to the caller.

The procedure **profile_start** is used to turn on profiling.

```
proc profile_start {{pattern *}} {
    global Profile_fd
    set Profile_fd [open /tmp/prof.out.[pid] w]
    foreach i [info procs $pattern] {
        profile_proc $i
    }
}
```

First, it opens the file that will receive the timing information, and calls **profile_proc** for the procedures we wish to profile. If commands are to be profiled as well as procedures, the **info procs** could be changed to **info commands**. At this point, the application can be run as usual, except that for each profiled command or procedure, a line of timing information is written into the file.

To turn off the profiling, **profile_stop** is called.

```
proc profile_stop {} {
    global Profile_fd
    close $Profile_fd
    foreach proc [info procs *:OLD] {
        regsub {(.*):OLD} $proc {\1} restore
        rename $restore {}
        rename $proc $restore
    }
    profile_summarize /tmp/prof.out.[pid]
}
```

The procedure **profile_stop** closes the log file, removes the special profiling procedures, restores the names of the original procedures, and calls **profile_summarize**, which prints a summary of the profile data. Fancier versions of **profile_start** and **profile_stop** could check to make sure a procedure isn't profiled twice, or that time isn't wasted profiling the profiling routines.

Analyzing the results

Analyzing the profile data is a bit tricky. The time attributed to a particular procedure consists of not only time spent in that procedure, but the time spent in all of its children ("called" procedures) plus the time taken by the profiling code that *times* the child procedures.

We can approximate the time spent in the profiling code by running a procedure with—then without—the profiling code and computing the difference. The procedure **profile_calibrate** does just that.

```
proc profile_calibrate {}
    global Profile_fd
    proc profile_dummy {args} {return $args}
    set Profile_fd [open /tmp/[pid] w]
    time profile_dummy 10
    set before [lindex [time profile_dummy 10] 0]
    profile_proc profile_dummy<\n>
    set after [lindex [time profile_dummy 10] 0]
    close $Profile_fd
    rename profile_dummy {}
    rename profile_dummy:OLD {}
    return [expr ($after - $before)]
}
```

A dummy procedure **profile_dummy** is created and timed. Then **profile_proc** is called to add the profiling code, and **profile_dummy** is timed again. The result is an approximation of the timing overhead.

```
proc Incr {name {value 1}} {
    upvar $name var
    if {[info exists var]}  {
        set var [expr $var + $value]
    } else {
        set var $value
    }
}
```

The **profile_summarize** procedure uses a souped-up version of the Tcl **incr** command (called **Incr** that permits **incr**ing a variable before it is set by automatically initializing it to zero before incrementing it.

```
proc profile_summarize {file} {
    puts [format "%35s calls   ms   ms/call %%" \
          name]
    set fd [open $file r]
    set total 0
    set overhead [profile_calibrate]
    # read in the data, and accumulate the values
    while {[gets $fd line] > 0} {
        set name [lindex $line 0]
        set time [lindex $line 1]
        set parent [lindex $line 2]
        Incr count($name)
        Incr sum($name) $time
        if {$parent != "TopLevel"} {
            Incr sum($parent) \
                 "- ($time + $overhead)"
        } else {
            Incr total $time
        }
    }
    close $fd
    # sort and print the results
    foreach name [lsort [array names count]] {
        if {$count($name) == 0} continue
        set ms [expr $sum($name)/1000]
        puts [format "%35s %4d %7d %6d %4.1f%%" \
            $name $count($name)  $ms \
            [expr $ms / $count($name)] \
            [expr $sum($name) * 100.0 / $total]]
    }
}
```

After a bit of initialization, **Profile_summarize** works in two stages, reading and tabulating the timing information, then sorting and printing the results.

Each line of timing information is read into three variables: the procedure **name**, the elapsed **time** in µs, and the **parent**, or calling, procedure name. Two arrays, **count** and **sum**, both indexed by procedure name, are used to keep track of the number of times each procedure is called and the accumulated time for each procedure. Next, the time attributed to each procedure is subtracted from the time credited to its parent, along with the timing overhead. Only procedures that are called from the top level have their times included in the **total** time counter. Otherwise, the time would be counted twice, once for the procedure and again for its caller.

Once all of the data is tabulated, it is sorted by procedure name. The results are printed in the form of procedure name, number of times the procedure was called, the total elapsed time spent in the procedure, and the percentage of the total program execution time spent in the procedure. The time values are divided by 1000 to convert them from µs into ms. Note that after about 35 minutes, the number of µs overflows a 32 bit integer, so profiling runs shouldn't be much longer than a half hour.

Listing 1 at the bottom of this article shows a subset of the results of a profiling run for an HTML library package. For this test case, it's clear that **HMrender** accounts for a proportionally large share of the total running time of the program. Even if it could be recoded to run infinitely fast, the overall application would be sped up by, at most, 27%. The negative time attributed to **HMtag_a** is probably due to the variability of the profiling overhead calculation.

The data in the logging file looks like:

```
...
HMrender 152083 HMparse_html
HMmap_esc 553 HMrender
HMzap_white 629 HMrender
HMx_font 3056 HMcurrent_tags
HMset_font 2022 HMcurrent_tags
HMcurrent_tags 14424 HMrender
...
```

Conclusions

Even though there can be considerable variability in the elapsed times of procedures, depending upon the current load on the processor, these simple profiling routines can quickly point out those parts of an application that are consuming most of the running time and would be good candidates for optimization.

> Stephen Uhler is a researcher at Sun Microsystems Laboratories, where he works with John Ousterhout improving Tcl and Tk. Stephen is the author of the MGR window system and of PhoneStation, a TCL-based personal telephony environment. He may be reached via e-mail at Stephen.Uhler@Eng.Sun.COM.

Listing 1: Sample Run

```
% source sample.tcl
Starting sample HTML viewer...
% source ~/column/profile/test
% profile_start
  [Run the application for a bit]
% profile_stop
```

HMextract_param

name	calls	ms	ms/call	%
HMcurrent_tags	465	1478	3	3.2%
HMdo_map	12	3	0	0.0%

186	773	4	1.7%	
HMgot_image	12	1199	99	2.6%
HMlink_setup	25	59	2	0.1%
HMmap_esc	467	307	0	0.7%
HMparse_html	3	2970	990	6.4%
HMrender	453	12544	27	27.0%
HMreset_win	3	220	73	0.5%
HMset_font	465	6292	13	13.5%
HMset_image	12	1401	116	3.0%
HMset_state	3	5	1	0.0%
HMstack	306	295	0	0.6%
HMtag_/a	33	124	3	0.3%
HMtag_a	33	-13	-1	-0.0%
		...		

Caldera Network Desktop v 1.0

Roger Scrafford

Issue #20, December 1995

The Caldera Network Desktop fulfills its promises and hints at a plug-and-play system for the masses.

It's slick, It's attractive! It installs on your i486 computer with a minimum of fuss. It does everything it says it will do, and—it's only in pre-release! (Version 1.0, Preview 1 is the version available at the time of this writing. [Preview 2 is now shipping from Caldera, and will be reviewed in a later issue—ED])

OK—back to reality. Though it looks familiar, there is nothing quite like Caldera Network Desktop (CND). It's something new to the Linux community. It includes a number of more-or-less independent packages, but here's the kick: some of them are commercial. That is, they are proprietary—you cannot redistribute them as you can with the usual Linux software. One to a customer, unless you get multiple-user licensing.

The manual puts it this way:

Caldera has included a Desktop metaphor, a NetWare client, a font server, and other commercial software that runs on top of the Linux operating system. Because Caldera has licensed these commercial components from other companies, they cannot be freely distributed, but are licensed on a per-copy basis.... You must have a license for each computer which runs these programs.

Well, that's pretty clear. But just to make sure you are informed, the manual goes to some lengths to include the GNU General Public License, the UC Berkeley copyright, and license terms for pthreads (technology used by the NetWare Client). This may be Linux but it isn't (entirely) free. It's a combination of freeware and proprietary software.

The Package

Version 1.0 of the Caldera Network Desktop Preview arrived at my doorstep via UPS from Banta ISG, in Provo Utah. The Caldera box told me I held "The Complete Client/Server Internet Solution;" what you find inside is an excellent 124-page *Getting Started* guide and one CD, which includes:

- Linux 1.2.8 from Red Hat
- an attractive X-window GUI
- 15 fonts in TrueType, Type 1 and SPEEDO formats
- a WEB browser and server
- a NetWare client (for NetWare 3.x and 4.x servers)
- servers for mail and FTP

I noted along the way that CND is English-only (except for the usual Linux internationality).

The *Getting Started* book was well-organized, and contained everything I needed to know to select a kernel and complete a Caldera Network Desktop installation.

So which is it? Is CND to be one of those software packages you install, but don't tinker with? It's advertised that way. Or is it a package in the Linux tradition—install, but grab your screwdrivers and immediately start modifying? Let's see.

Red Hat

CND is built on top of the Red Hat Commercial Linux distribution. Preview 1 shipped with the 1.2.8 Linux kernel. It was compiled with IPX support and **CONFIG_MODVERSIONS** enabled, and allows you to disable verbose boot messages.

The standard C and X11 tools and libraries are included. Preview 1 shipped with version libc version 4.5.26 and gcc version 2.5.8. libc version 4.6.27 is also included on the CD.

The documentation discusses some of the differences between the Red Hat Linux file system structure and the Linux File System Standard (FSSTND). And, as with other Linux distributions, your starting kernel has a bunch of stuff you don't need for your particular machine; compiling a new kernel after Caldera installation is, as always, a Good Thing. [*Linux Journal* covered this in issue 7, November 1994, and it hasn't changed very much since. Just be sure to say "yes" when you are asked about **CONFIG_MODVERSIONS**—Ed]

On occasion during my trials of CND, I found myself eating Flaming Death at the hands of Caldera support folks, and I was not alone. Slackware users, it seems, should have known better than to use that shady distribution. No matter, I say; if I came from a non-Red Hat environment, CND's purported ease of use should have smoothed the transition. [Preview 2 gives some of this capability—Ed] Curiously, Caldera's Web page (http://www.caldera.com/) hints that it is possible to unbundle the Network Desktop from the Red Hat distribution, and a few information files on the Caldera site actually give you tips on how to do it—on top of Slackware.

Networking

CND recognizes the following:

- Novell's IPX
- Novell's NetWare 3.x, 4.x (NDS) file server access (no printers yet).
- TCP/IP/NFS, etc.
- Samba 1.9.00 (an SMB server)

I did not test the Samba server, but can praise the absolutely transparent Novell and TCP/IP services. Applications and X windows are one thing, but it's this sort of functionality that will make Linux a contender at my own workplace.

X windows

The desktop environment, called Looking Glass, is based on Visix's Looking Glass Professional. It runs as an application on top of X11 and is used in conjunction with an unmodified version of the fvwm window manager. From the CND FAQ:

There is a general purpose file typing facility for the desktop metaphor: actions can be defined for a given file type. And drop actions can be defined: dropping an HTML file on the browser will launch the browser on that file.

A graphics card supporting at least 256 colors is required. Graphics cards which are supported by XFree86 are also supported on CND. A smooth program called Xconfigurator is also provided, which creates an XFree86 configuration file for many popular video cards.

Web Browser and Server

The Web browser provided is Arena. It is serviceable, but while it was being ported to Linux it acquired some color allocation problems. While it claims HTML 3 compliance, it doesn't support forms or e-mail.

The Web server installs and works invisibly. I found no problems with it, but did not explore scripting capabilities, or its performance under stress.

Technical Support

For the time being, Caldera support "is limited (mainly via e-mail and our WWW and FTP servers on the Internet). No individual support is provided." The Caldera page at www.caldera.com included pointers to a wealth of information, including Linux in general, Red Hat in particular, and, of course, CND FAQs and other technical information.

Installation

To make all this work you need to install it on an i386 or i486 computer with one 3-1/2" floppy, at least 80MB of hard drive (although the excellent "Express" install uses 140MB, and a complete install seems to be at least twice that), 8MB of RAM (a more realistic 16 MB if you plan to use X-windows), a CD drive which the supplied Linux boot kernels will recognize, and an appropriate net connection. With that and three blank floppies ready for the necessary boot, root and recover disks, you're ready to install—or are you?

The years have taught me that a single software installation on a single hardware combination can give a ludicrous impression of the product. Consequently, I have taken the trouble to install Caldera on hardware including:

- Mitsumi and Matsushita CD (with Soundblaster)
- 100MB, 130MB, 220MB and 820MB hard disk
- Western Digital and Tseng VLB

If this seems a waste of time, trust me; each element plays a part in setting up a user's reaction—"This is a hunk of junk", "this is great", or "this is really strange."

I have performed these Caldera installations in two different environments. The first is my place of work, where I have an Ethernet connection to our WAN and from there, through a firewall, to the Internet. The second is at home, where my connection to the outside world is via the phone company and a local Internet provider.

Something you must not ignore: Believe the documentation when it tells you how much disk space is required. The "express" install calls for not less than 140MB; anything less leaves you with an incomplete, totally unbootable installation. Start over, with plenty of disk space—I'd say nothing less than 220MB (cheap these days). Because part of the point of Caldera is that nifty "Looking Glass" desktop, that means lots of space is needed. You'll want space for a swap partition, X itself, all the Caldera-specific paraphernalia, net stuff,

and so on. And you'll want to build kernels every now and then, so you'll need gcc, the libraries, and so on. For all this, remember to keep your video specs handy, and to have fun. The manual has a handy table which allows you to calculate the disk space required, or make decisions at to what programs you can afford to skip. It's invaluable.

After your initial installation, software may be installed or removed by means of either the Red Hat Program Package (RPP) command-line tools, or the Linux Installation Manager (LIM) graphic interface to RPP.

Quirks

Some of the quirks are a result of Linux itself. During installation, for instance, after ten minutes of disk activity, the screen goes blank! Of course—it's only the screen blanker, what could be more logical? But if this is your first shot at a Linux installation, you might be tempted to do something nasty, like reboot. (Hint: press the shift key; anything else will be interpreted as an “OK” when it gets around to checking for keyboard input, and you might not want that next “OK”.)

Although the CND installation creates standard users, it seems to forget their passwords; it creates standard groups using the User Private Group scheme.

The bootroot program (located in the CD's <\\>dos directory), which is supposed to lead you through the creation of the startup floppies, proved to be a memory hog, refusing to write anything to the disks on some of my test machines. I found that once I selected the appropriate “boot” and “root” files, rawrite was the best means of creating the necessary disks on those computers.

My workplace computer has a Matsushita CD player connected to the computer through a Media Magic sound card (I used “other SCSI” during kernel selection). Both work well, but this CD player has a motor-driven CD tray; during installation, the kernel causes the tray to go in and out like a cuckoo—some six times during an install. Don't blame either Caldera or Red Hat, as I was tempted to do; this seems to be a Linux Fact of Life.

Politics

Caldera asserts that despite recent improvements in commercial distributions, Linux still lacks acceptance as an operating environment in the commercial world (although some recent *Linux Journal* articles have shown matters to be gradually changing). Some of the reasons given: Linux is often perceived as having been developed by unskilled students, and installation and configuration are challenging for the uninitiated. Further, they point out, Linux

offers no accountability: it is all but unsupported by mainstream applications, and it can't provide a complete solution to users' needs—networking, gui, and so on.

I leave it up to you to decide if any of these are straw men. In any case, the entrepreneurs at Caldera label these “barriers to growth.” And in order to remove those barriers, they have brought us the Caldera Network Desktop. They will “add value to Linux by creating and providing a platform for commercial products that can appeal to major users and spread the use of Linux to new areas that traditionally would not have considered using it.”

The future, Mr. Gittes

What lies ahead for this package? Again, from the CD's preview document:

- a more mature WWW/HTML browser
- sophisticated, commercial-grade tape backup system
- a new graphical interface to many utilities and programs
- commercial personal productivity applications
- better Internet access applications

Although it isn't yet possible to upgrade from one release to another, Caldera says that “such tools are planned for the final 1.0 release.” [Those tools are in Preview 2, which has just been released—Ed] Something Caldera calls the InfoTrack database support system will become part of the overall technical support. OpenDoc support is in the offing. And ELF work is under way. [Again, Preview 2 is based on ELF—Ed]

How about Linux in general? A number of commercial packages are said to run on Linux, including Word Perfect and Oracle 7. Indeed, Caldera includes the SCO Word Perfect demo, and one of Caldera's future offerings includes Word Perfect itself. Perhaps Linux is being moved in this direction, with or without Caldera.

Conclusion

Caldera says it wants to “shield end-users from the ordered chaos that creates and grows Linux, so they can use it as their operating system of choice.” Well, I've grown jaded over the years—cynical, I suppose (must be that stint as a Windows 95 beta tester). When so much of a software package actually works, I am surprised; and this first release of Caldera's Network Desktop has been, to my mind, a remarkable success.

My suspicion is that the people who buy Caldera will be expecting a software package they can simply install and run. And, if you take a few precautions (have enough disk space; know your video numbers; and—yes—RTFM), that's exactly what happens.

Some take the commercialization of the Net to signal the End of Things as We Know Them. Will products like Caldera mean the end of Linux as we know it? I think not—the philosophies are not mutually exclusive. True, Caldera “uses” Linux, which is GNU freeware, to make money. Still, I think Caldera will prove to be good for Linux. The solidity of Linux makes a product like Caldera possible; and the success of Caldera will make Linux accessible to people who don't want to tinker—who just want to learn, or maybe even do some work.

Meanwhile, in true Linux tradition, “programmers the world over” are doing a fine job of bashing this package—breaking it, fixing it, feeding the fixes back to the folks at Caldera. The “First Customer Ship” will be a better, more solid product because of this test cycle.

I like the product. It can't pretend to be plug-and-play, but it installs easily, runs well, looks great, and—unless you try to stretch it too far—keeps on running. Caldera wouldn't exist without Linux; Linux could continue to exist without Caldera, but this certainly ups the ante.

Roger Scrafford wrestles with Linux, Novell, and Win95 at his day job in Seattle. You can reach him via e-mail at rscraff@aa.net.

Archive Index Issue Table of Contents

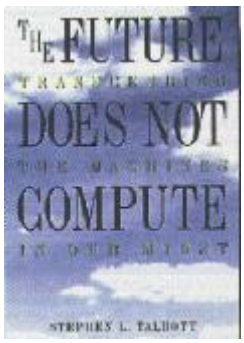
Advanced search

The Future Does Not Compute: Transcending the Machines in Our Midst

Danny Yee

Issue #20, December 1995

I got about two thirds of the way through *The Future Does Not Compute* before giving up on it.



- Author: Stephen L. Talbott
- Publisher: O'Reilly & Associates
- Pages: 481
- ISBN: ISBN 1-56592-085-6
- Reviewer: Danny Yee

At heart *The Future Does Not Compute* is an attack on the excesses of technological optimism, particularly those associated with computers. Talbott is worried about the identification of technology with the ends it is supposed to help us achieve (of communication systems with community, for example) and with what our growing reliance on technology indicates about ourselves: he sees computers as bringing out the worst in people and as a kind of reflection of a wider malaise. Part one is a broad survey of the relationship between computers and human communities (which wanders as far afield as globalization and business ethics); part two considers the effects of computers in education; part three is about the effects of computers on writing and

language and on the elevation of “information” to an end in itself; and the final part is about the “evolution of consciousness”.

I got about two thirds of the way through *The Future Does Not Compute* before giving up on it. It wasn't its anti-technological stance which turned me off, but rather the way Talbott went about presenting his ideas. All too often he lapses into complete mysticism, with such outbreaks as

> The solitary bird, gripped by an unknowing intensity as it ploughs through the trackless ether, hears, on the dull edges of its consciousness, a call of destiny sung by hidden choirs.

(which is defended by a counter-attack on a ridiculous straw-man reductionism). Even where he avoids outright obscurantism, Talbott lacks any kind of respect for philosophical niceties. He keeps on attributing intentionality to machines, for example, in such fashion as

> the computer ups the ante in a game already extremely perilous. It relentlessly, single-mindedly, apes us even in those habits..., for it is endowed in some sense with a mind of its own.

and talks about them “acting in their own right”, but at the same time refuses to allow that there is *any* sense in which they can approach human beings. For all he goes on about the “mechanical” being present in our “inner selves”, Talbott's rhetoric relies heavily on a rigid and fundamental dichotomy between the natural and the artificial, between man and machine. He makes no attempt at all to justify this assumption: even the references to Searle and Penrose I kept expecting never turned up.

Talbott has an eye for an effective phrase and an impressive vocabulary (sometimes he verges on poetry), but this is taken to extremes: he is always prepared to sacrifice clarity and content on the altar of rhetoric and effect. Terms like “human nature” and “consciousness” and phrases such as “our trust as stewards of the Earth” and “sleepwalking subservience to technology” are used without any explanation in critical contexts. Here are some longer examples:

> We must tame technology by rising above it and reclaiming what is not mechanical in ourselves.

Where, as a child, I differentiated myself from the animal, now I must learn to differentiate myself from the machine—and this differentiation lies in the deepening of *consciousness*. It is therefore not only a matter of pointing to established human nature; it is also a matter of *realizing* human nature in its

movement toward the future. Related problems are that Talbott presents his ideas in a rambling and disconnected fashion, in short sections within short chapters, and that he has a tendency to say the same thing over and over again in different ways.

The Future Does Not Compute also lacks any kind of sociological or psychological depth. The discussion of globalization, for example, doesn't engage at all with historical or economic perspectives on the subject, and analysis of at least one concrete example (of a network group which claims "community" status) might have given some basis to his analysis of the relationship between technology and community; as it is, all he can offer on these subjects are empty generalizations.

The most distressing thing about *The Future Does Not Compute* is that I actually agree with quite a lot of what Talbott has to say, both on general subjects like education and the evils of assuming there are technological solutions to all human problems and on more specific ones such as profit-seeking business ethics and naively optimistic claims by artificial intelligence researchers. I fear, however, that many people will use his book as another excuse to reject all discussion of some important questions as obscurantist nonsense. In the end I abandoned *The Future Does Not Compute* because I wasn't learning anything from it: it is emotionally stirring but intellectually vacuous.

> Disclaimer: I requested and received a review copy of *The Future Does Not Compute* from O'Reilly & Associates, but I have no stake, financial or otherwise, in its success.

All book reviews by **Danny Yee** are available via anonymous FTP ftp.anatomy.su.oz.au in /danny/book-reviews (index INDEX) or URL www.anatomy.su.oz.au/danny/book-reviews/index.html He can be reached via e-mail at danny@cs.su.oz.au. Comments and criticism welcome.

Archive Index Issue Table of Contents

Advanced search

Letters to the Editor

Various

Issue #20, December 1995

Readers sound off.

Speedy Delivery

I'd like to congratulate *Linux Journal*. I bought all the back copies I could get and found something of real value in each one. What I particularly appreciate, aside from the topical coverage of the content, is the excellent distribution you use: US mail. I'm amazed. My October issue arrived in my Auckland NZ post box on 22 September! This compares very favourably with other US magazines which typically take over three months.

Keep it up guys, I like it!

—John Hardcastle johnh@helec.co.nz

We use ISAL: International Surface Air Lift. This means that all the magazines for non-US destinations are sent in a single package overseas, and from there are distributed by national surface mail. We have found this to be highly effective—in fact, some international subscribers receive their issues of Linux Journal before some US subscribers.*--Editor*

Alternate Paper

Just read your column about the idea of printing *LJ* on recycled paper and use other environmentally responsible solutions. I just wanted to tell you that I fully support this—even if it would cost me $10 more per year.

—Christian Perrier bubulle@bubhome.frmug.fr.net

Yesterday I learned about the increase in *LJ* due to the cost of paper for printing. David Niemi and I discussed alternative papers that could be used. I

suggested that hemp paper *might* be cheaper. David thought that you would be open to using hemp. So I contacted the Hemp Industries Association (thehia@aol.com) about companies that sold hemp paper. There is also a person to contact about hemp paper. Maybe it is cut costs and be environmentally sound. If you want I can contact the companies and get information about hemp paper for magazine publishing. I would probably need info about the size, color runs, pages, etc to pass on though.

Hopefully hemp may be a way to make *LJ* more cutting edge!

—Gregory J. Pryzby gjp@sddi.com

I am very encouraged to see you considering tree-free & acid-free alternatives for *LJ*. I would definately be willing to pay extra money for a more environmentally sound magazine. The pleasure I already derive from reading *Linux Journal* could reach nirvana like states if I knew everything I was reading was printed on ecologically sound hemp paper.

I've taken the liberty to pass on your e-mail address and information about the paper alternative discussions to Paul Stanford, founder of Tree-Free Eco Paper, based in Portland, Oregon. I hope that you are able to come up with a solution that makes both environmental and ecomonic sense. Paul can be reached at treefreeeco@igc.apc.org.

—Robert Lunday robert@skunk.hemp.net

P.S. A colleague of mine wanted me to tell you that she would buy your magazine simply *because* it was printed on hemp paper.

While I approve in principle of using hemp/straw paper for publishing the *Linux Journal*, the cost of the publication is a factor for some of us. I'm living on a graduate school stipend, and while an extra $5 here or $5 there probably won't break me, every little bit helps. Perhaps *LJ* should consider student/educator rates that would be closer to the present subscription price should the magazine be published in the future on straw/hemp paper and general subscription prices be forced to rise to cover costs.

Sincerely,Kenneth E. Harker kharker@cs.utexas.edu

First of all congratulations to *LJ*. I am a subscriber since the early days. I also gathered all *LJ*. (I think that the nature of human beeing to gather). In my case, I wouldn't mind to pay $5/year more if I know that I spend the money for the environment.

Thank you for doing *LJ*.

Best regards,Rene von Arx rene.von-arx@alcatel.ch

In your editorial dialogue with Charles Stickelman, you asked what subscribers would think about increased cost of *LJ* if you switched to hemp/straw based paper (if they became available). I would like to say that I would be willing to pay more for such a product, and would probably be willing to go up another $15.00 per year. If we want to become environmentally responsible, we need to make our demands known to the suppliers of products who have alternatives available. If we are not willing to pay for environmentally friendly products, we aren't in much position to gripe about other people.

—James A. Robinson jimr@simons-rock.edu

In Issue 18, you ask the question about who among us would be willing to spend more on the magazine in order to get low-chemical biodegradable papers and inks.

While I am far from an “environmentalist”, I see no point in putting chemicals into the environment when there are perfectly acceptable natural alternatives (e.g. Soy Ink and low-acid paper).

In the same vein, I am no fan of drug use (even marijuana), but given the other uses for hemp (rope, paper, etc), its high quality in these applications, and its possible abundance, I find it deplorable that we have such draconian laws and enforcement to “control” it.

I guess you only asked about higher subscription rates for “environmentally-friendly” magazines... I would be willing to pay $3-5/yr more for the time being. I suspect that as more publications make use of these components they will become cheaper and we will not have to pay higher prices for long.

—Michael George 71540.164@compuserve.com

I'm not against the use of more environmentally friendly printing methodologies. I am however, concerned about any degredation in the current good quality of *LJ* paper and printing.

I've found that some of the newer environmentally friendly journals are easily smudged, making them a poor choice for use as long term reference material.

So for me, it's not an acceptable tradeoff if quality and durability is lessened in the change. I propose that if this is the case, we hold off on any modifications until a solution is found that provides a gain on both these issues.

Keep up the outstanding work.

—Best regards,James Cassidy jcassidy@proton.genesoft.com

At current alternative paper prices we need thousands of subscribers willing to pay a premium price. However, we will keep an eye on price and availability of these products.--Editor

Not Paper

I have a great idea for saving money on paper costs. I would like to see an electronic version of the *Linux Journal*. It would save paper, ink, printing costs, and lots of trees. The only thing to figure out is how to distribute it. You could encrypt it with the subscriber's public key, or something like that. I am sure that something could be worked out.

I have been surprised that publishers have not moved into the electronic market yet. The potential for selling information is tremendous.

The primary reason that I am interested is because I am blind. When I get my journals, I have to find someone with the time to read me the articles I am interested in. Since almost no one has time, it usually involves paying someone by the hour to do this. $5.00 per hour can get steep.

I would be willing to work with anyone who be interested in doing this. I have a lot of ideas on security, making sure the magazine gets to the intended subscriber, and other issues related to electronic publishing.

I am willing to donate my time as I hope that as more things are made usable by me, it will increase my money making potential. I will be able to generate more income and spend less to keep up with the “printed” media.

I hope someone takes a serious look at this note and that it does not go to the great black bit bucket.

Thanks in advance for your time. I look forward to hopefully working with some one on this.

—Kelly Prescott kellypre@linkup.com

Right now, we are working on distributing it via our WWW site.

We are in the process of writing a markup language for *Linux Journal* that is similar to HTML and will allow us to do many different kinds of distribution, including paper, WWW, and potentially other kinds of electronic distribution. You are not the only blind user to have contacted us, and we would like to be able to meet your needs better.

Controlling distribution is not a big issue. Most profits come from advertising revenue, even though we have some of the lowest ad rates in the industry. The more subscribers we have, the more advertisers are interested in buying advertisement.--Editor

Archive Index Issue Table of Contents

Advanced search

Just Browsing

Phil Hughes

Issue #20, December 1995

Hundreds of Web servers (including our own) and Internet Service Providers (including the one we use) are based on Linux systems.

As we all know, Linux is what it is today because of the Internet. Without the international communications made possible by the Internet, such a development effort would not have been possible.

In addition, Linux is returning much to the Internet. Hundreds of Web servers (including our own) and Internet Service Providers (including the one we use) are based on Linux systems. To their credit, many commercial vendors have jumped on the Linux bandwagon. Some just sell hardware pieces, some sell complete Linux systems and some sell software for Linux. Looking at the ads in this magazine will give you a pretty good feel for which companies have decided to jump on the Linux bandwagon.

SSC has seen the connection between Linux and the World Wide Web and is starting a new magazine, *WEBsmith*. If you are an *LJ* subscriber, you will see the premier issue of WEBsmith bound into your January, 1996 issue of *LJ*. While there is a lot more to the Web than Linux, we want to promote the Linux/Internet connection.

Now, for the bad news. As I am writing this, the new version of Netscape, the most popular Web browser, was just released. Although a Linux version of Netscape exists, it is not supported. Also, while Netscape has secure server software available for other platforms, it is not available for Linux.

I don't know how you feel about this but, to me, it makes me think we are being treated as second class citizens. It's not that Netscape doesn't work on Linux. It's just that apparently Linux and the Linux community is not being taken seriously.

We are a community of activists. We have made Linux go from nothing to the operating system of choice for hundreds of thousands of people around the world. And we have helped Linux make inroads into the commercial world. I think it is time we do a little activism with regard to Linux and Web browsers.

We don't have to start from scratch. Arena, available on many Linux distributions and archives, is a work-in-progress browser for HTML 3.0. While not complete, Arena offers some very nice features. My favorite is that it actually verifies that your HTML is correct. The first time I ran Arena on our Web site I found that about 90% of our pages produced the bad HTML error message. It's not that I am proud of this, it's just that I see having a tool that checks your work as being valuable.

Arena is from the World Wide Web Consortium (W3C), an industry consortium run by the Laboratory of Computer Science at Massachusetts Institute of Technology. In Europe, it is a collaboration of MIT with CERN, the originators of the Web, and with INRIA, in France. Arena is built using the library of common code called the **W3C Reference Library**. It is currently available as a binary for most major Unix platforms (where, in this case, major includes Linux).

To quote the W3C on their plans for Arena

> Arena will continue to be a testbed browser for HTML3 and style sheets. We do not have the resources nor the intent to make Arena a full-featured web browser, but welcome initiatives to help add functionality.

I haven't talked to anyone at W3C but, if there is interest in the Linux community, I am willing to spearhead an initiative within the Linux community aimed at developing further functions for Arena.

One such initiative comes from David Bonn, of Mazama Software Labs, who suggested writing what you might call *browser tools* that would make it possible to easily embed an HTML browser in your application. This has the advantage that you could build systems where browsing HTML was just a part. For example, you might want to build a system for your office where it would be possible for clerical people to access procedures that were in HTML format. You could include this in the application that they commonly ran instead of having them learn about a new program in order to read these procedures.

I am sure there are lots of other ways to go. At this point I am just sending the idea of a new development effort up the flagpole to see if Linux community members are interested. Let us know what you think. Send us mail or, better yet, e-mail us at info@linuxjournal.com.

Resources

World Wide Web Consortium URL: www.w3.org

Arena Information: www.w3/org/pub/www/Arena

Archive Index Issue Table of Contents

Advanced search

Finding Files and More

Eric Goebelbecker

Issue #20, December 1995

All about the find command.

Not long after getting their first Linux system, new users usually need to locate a file somewhere on their system. So they learn the following command from a friend, or maybe a book or magazine:

```
$ find / -name filename -print
```

Now while this command does work perfectly fine, the syntax does seem awkward to people unfamiliar with the **find** command. Why should we have to specify **print**? [Note: On Linux systems, and other systems that use GNU find, we don't. But standard Unix find insists on it, so you might as well get used to it if you use Unix as well as Linux.]

For that matter, why should we have to specify **name**? Why not just **find *filename***? It's this seemingly cryptic structure that makes find one of the most under used commands in the Unix toolbox.

A look at the find man page (on any system, not just Linux) completes the confusing picture. For someone not familiar with Unix, **find**'s “operators” and “expressions” make it an awfully complicated program just for locating files.

If all you want to do is locate a file, there is a better way to do that:

```
locate filename
```

This will work on a properly set-up Linux system with GNU find. Why have a complicated command like **find** when we already have a simple command like **locate**? Because find is good for much more than just finding files. (Good Linux distributions some with update properly set up. If yours isn't, you can run **updatedb** as root to update the database it uses, or simply use find as shown above).

The Caldera/Redhat system that I use at home has several entries in the crontab that run this command:

```
find /tmp/* -atime +10 -exec rm -f {} \;
```

This command deletes any files in /tmp that haven't been accessed in the past ten days. The fact that find only deletes files that haven't been **accessed** in the past ten days rather than files that were **created** that long ago is a subtle, but very important point. Find gives us access to the very valuable set of information stored about files and directories in Unix filesystems.

Like most Unix filesystems, the second extended filesystem ("ext2") that is used on most Linux systems stores a more extensive set of data about files than just their name, size and last-change-date the way systems such as DOS do. It also stores an owner and group, access mode, the dates that the file was last modified and accessed, the date that the file last changed status, and the type. (Don't worry, we'll explain these as we go).

With the exception of the names, all this information is stored for each file and directory in a structure called an inode. In Unix filesystems, directories are simply *files* that contain a list of filenames with inode numbers.

fs	mode	User	Group	Access	Modify	Status	-noleaf
ext	y	y	y	●	y	●	y
ext2	y	y	y	y	y	y	y
hpfs	★	★	★	y	y	y	n
minix	y	y	y	●	y	●	y
isofs	★	★	★	n	n	y	n
isofs/RR	★	★	★	n	n	y	y
msdos	●	★	★	●	y	●	n
nfs	y	y	y	y	y	y	y
sysv	y	y	y	y	y	y	y
umsdos	y	y	y	y	y	y	y
xiafs	y	y	y	y	y	y	y

key:
● follows modify time
★ specified by mount option

isofs/RR is iso9660 filesystem with Rock Ridge extensions.
Most Linux CD's are made with the Rock Ridge extensions.

Table 1. Inode Entry Fields

Table 1 has a list of inode entry fields and how they are "translated" for the different filesystem types supported by Linux. While this table may not mean much to you yet, it should be self-explanatory by the time you finish reading this article.

The Command Line

Let's analyze the find command line:

```
find starting-point options criteria action
```

- ***starting-point*** One or more directories from which to start searching. The default is the current directory.
- ***options*** Modify the methods used for searching in several ways.
- ***criteria*** Specify which files are chosen, and which are ignored. All files found are chosen by default.
- ***action*** What to do with the files that are chosen. GNU find has a default action of **-print**, but standard Unix find has no default action, and will abort and complain unless an action is explicitly provided.

The Starting Point

The *starting-point* parameter has two effects on find's actions. The most obvious is that it specifies in which directory (or directories; there can be more than one starting point) to start looking for files. The other effect is on how the chosen filenames are treated, as this example shows:

```
$ cd /usr/X11/man
$ find man5 -print
man5
man5/XF86Config.5x
man5/pbm.5
man5/pgm.5
man5/pnm.5
man5/ppm.5
$ find /usr/X11/man/man5 -print
/usr/X11/man/man5
/usr/X11/man/man5/XF86Config.5x
/usr/X11/man/man5/pbm.5
/usr/X11/man/man5/pgm.5
/usr/X11/man/man5/pnm.5
/usr/X11/man/man5/ppm.5
```

When a user is simply looking for a file, this difference in behavior does not matter very much. But when you want to use the output from find to drive another program, it can be very important, depending on the program being driven.

In addition to the starting point, we have control over some other aspects of find's behavior, such as how it should handle soft links, how to evaluate file timestamps and how deep to follow directory structures. These are controlled by *options*.

The **-follow** option tells find to follow soft (or symbolic) links to the actual file. A soft link is a file that “points” to another file. To demonstrate this option, create

(as a normal user, not as root) a soft link with **ln** in your home directory that points to file that belongs to root.

```
$ cd
$ ln -s /vmlinuz ./kernel
```

Now use ls to produce a long listing for the file.

```
$ ls -l kernel
lrwxrwxrwx ... kernel -> /vmlinuz
```

The first column of the mode, **l**, tells us it is a soft link. We also are told what file the link “points” to.

Now let's demonstrate the effect of find's **-follow** option by searching through the directory for files belonging to root, using it. (uid 0 is root; we'll cover the -uid option in more detail later.)

```
$ find . -uid 0 -print
nothing is printed
$ find . -follow -uid 0 -print
./kernel
```

You created the link to the kernel, so you own the link, called ./kernel. But the file /vmlinuz is owned by root.

The **-daystart** option modifies the behavior of find when it comes to evaluating time. When **-daystart** is specified, find will measure days from the beginning of the day instead of from 24 hours ago. (We will cover the parameters related to time later.)

Frequently a user will need to find a file that he or she knows is somewhere on local hard disk, and not on a mounted cdrom or network volume. An easy way to keep find from straying to these other disks is with the **-xdev** option.

```
$ find / -name document -print
```

will cause find to search for the file “document” in every directory under /, which can be very slow with a CDROM or network filesystem mounted.

```
$ find / -xdev -name document -print
```

will instead cause find to limit its search to the device that / is mounted on. (An alias for **-xdev** is **-mount**) Of course, if you have more than one local filesystem, you will need to execute a different search for it. Perhaps

```
$ find / /usr -xdev -name document -print
```

if you have two partitions, one for / and one for /usr. Alternately, you can say

```
$ find / -fstype ext2 -name document -print
```

if all your local partitions are ext2 filesystems.

Another way to save time on searches is to use the options related to directory depth.

```
$ find /usr -maxdepth 4 -name document -print
```

will limit find's search for document to directories four level deep or less “under” /usr.

Another option related to directory depth is **-depth**, which causes the directories to be selected before any files in them. We'll see later why this is useful.

The **-noleaf** option is used for searching filesystems that aren't Unix-like. Table 1 tells for which filesystems specifying **-noleaf** may speed up your search.

We already had an example of finding a file by name. Other mechanisms for matching filenames are **-path**, which matches by directory name, **-iname**, which is similar to **-name** but case insensitive, and **-ipath**, which is also case insensitive.

Pick and Choose

Criteria allow you to select files.

Each file has access, status, and modification times, and find provides three time-based criteria, one for each of these values. They can be checked in increments of days or minutes, and files can be compared based on these times.

The modification time is set every time the file's contents are changed.

```
$ find . -mtime +10 -print
```

will print out files that have not been modified in the past ten days, similar to our second example.

In the previous example we used the plus sign to signify “more than.” In addition to this, find also supports the minus sign to indicate less than.

```
$ find / -mtime -5 -print
```

will print out files that were accessed less than 5 days ago. The absence of these operators will cause find to choose **exact** matches. As mentioned before, the **-daystart** option will modify the search so that the dates are based on the most recent midnight instead of 24 hours before now.

To use minutes instead of days, use the **-mmin** criterion.

```
$ find . -mmin +10 -print
```

will output files that have been modified more than ten minutes ago.

The **-newer** criterion

```
$ find . -newer document -print
```

will output files that have been modified more recently than document.

The command sets both the *access* and *modification* times on files. If the file does not exist, it will be created. We can use it for an example.

```
$ touch foo
```

will create a file named “foo” in the current directory, if there isn't already one there. Now,

```
$ find -mmin 1 -print
```

should output **foo**, but

```
$ find -mmin 2 -print
```

should not.

For access time, which indicates the last time the files were opened, find has similar options. For days there is **-atime**, for minutes **-amin** and for comparisons **-anewer**.

Status time initially indicates creation time, and then follows any modifications to the file or its *inode*. It can be used with **-ctime**, **-cmin**, and **-cnewer**. These criteria match files based on the last time a file's ownership, access mode, or other characteristics have been changed.

Find also has a **-used** option. It will match files that have been accessed since their status was last changed:

```
find -used +2
```

will find files that have been used more than two days since their status was last changed.

I've mentioned file modes a few time throughout this article. File modes express which users may perform certain operations on a file, what type of file it is and also some other information about the file. find allows us to match files based on their mode.

Before I go over these options, I will explain file modes and how they are displayed and set.

Users most commonly come in contact with file modes when they concern file ownership and access. A file belongs to an owner and a group, therefore it follows that access is controlled with respect to three entities: owner, group and world. (“World” is made up of users that are not the owner and do not belong to the affiliated group.)

Access is controlled with respect to three actions: Reading, writing (which includes deletion) and execution. Let's look at the output of a long listing with ls.

```
$ ls -l foo
-rw-rw-r-- 1 eric staff  0 Sep  6 22:55 foo
```

(I've deleted some of the spaces ls normally creates in order to fit the entire output.) The leftmost column of the output has ten characters that show use foo's mode and file type. From the left, the first is used by ls to show us the type of file. For example, if it were a link or directory we would see an **l** or **d** there.

The remaining nine characters show us the access mode. In groups of three, the show us the rights for owner, group, and world, in that order. Each triplet has a field for read **r**, write **w** and execute **x**.

```
$ chmod 777 foo
$ ls -l foo
-rwxrwxrwx 1 eric staff  0 Sep  6 22:55 foo
```

We have turned on all permissions for all users on the file “foo”.

The **chmod** command can use two different kinds of notation, symbolic and octal. While symbolic notation is easier to remember for most people, I used octal notation, because it is the format for modes that find expects. With this notation each number represents the octal permissions for each user class.

The permissions are calculated by adding the following:

- 4 Read
- 2 Write
- 1 Execute

So if you want to give the owner of a file full permissions and group and world only read and execute permissions, you want to “set” all bits for owner, and the read and execute bits for the others:

```
Owner = 4 + 2 + 1 = 7
Group = 4 + 1     = 5
World = 4 + 1     = 5
```

So the command would be:

```
$ chmod 755 program
$ ls -l program
-rwxr-xr-x 1 eric  staff 106410 Sep  6 22:55 program
```

The listing shows the mode we expected.

Back to find: the **-perm** criterion accepts this type of notation.

```
$ find . -perm 777 -print
```

would find all of the files in and under the current directory that have read, write and execute permissions set for all users.

The **-perm** option also supports the **+** and **-** operators.

```
$ find . -perm +600 -print
```

would output any files that are readable **or** writable by their owner.

```
$ find . -perm -600 -print
```

would output any files that are readable **and** writable by their owner.

Therefore the **+** acts as a boolean “or” and the **-** acts as a boolean “and”.

The ability to find files based on their permissions is an important security tool. Later, I will cover some important special file modes, and how find can help protect a system from attacks that use them.

File size is another option offered by find. File sizes may be specified in 512 byte blocks, two byte words, kilobytes or just bytes. Since size is a numeric option + and - are also supported.

```
$ find . -size +4096k -print
```

will print the names of any files larger than four megabytes.

```
$ find . -size -1c -print
```

will print the names of any files smaller than one byte. The **-empty** option also matches empty files.

For 512 byte blocks the number should be followed by a “b”, for 2 byte words a “w”.

There is one caveat when searching for files by size. Some files, such as /var/adm/lastlog, have more space allocated than they actually use. These files are known as “sparse” or “holey” files. Like ls, find will report these files by the space they have allocated, not the space they are actually using. If you have any doubt about how much space a file is using, use the du command.

```
$ ls -l /var/adm/lastlog
```

reports a size of 16032 (15k) on my system;

```
$ du -k /var/adm/lastlog
```

reports only 3k.

Our first example showed us how to find a file when we know the exact name. Find will also accept the ***** wildcard, but the file name must then be quoted in order to prevent the shell from expanding the file name before passing it to find.

```
$ find / -name "*gif" -print
```

will output all of the files ending in “gif” on the entire system.

In addition to simple wildcards, find also supports regular expressions with the **-regex** option.

```
$ find . -regex './[0-9].*' -print
```

will locate any files in the current directory that begin with a number. Note that the regular expression is applied to the entire path, which makes the expression rather difficult to write. For more information about regular expressions see the man pages for grep or the article in the October issue of *Linux Journal*.

Another search criterion is file type.

```
$ find / -type d -print
```

will list all of the directories. Here is a list of the file types and the appropriate letter to use to search for them.

- **b** block special files such as a disk device.
- **c** character special files such as a terminal device.
- **d** directory
- **p** named pipe
- **f** regular file
- **l** symbolic (soft) link
- **s** socket

If you are unfamiliar with any of these file types, don't worry. You can learn as you go.

Files can also be matched by user of group id. As demonstrated earlier,

```
$ find . -uid 0 -print
```

will output all files belonging to root.

```
$ find . -uid 120 -print
```

will output all files belonging to the user with UID 120.

To make things easier,

```
$ find -user eric -print
```

will output all files belonging to eric.

Find also has similar options for groups: **-gid** and **-group**.

More than printing!

Now that you know how to locate just about any file, what can you do with them besides print their names?

```
$ find . -fprint foo
```

sends a list of the files in the current directory to a file “foo”. If the file does not exist it is created. If it does, its contents are replace.

Find also offers the **-printf** action. This allows output to be formatted.

```
$ find . -printf 'Name: %f Owner: %u %s bytes\n'
```

produces a table of files with their name, owner, and size in bytes.

The **-printf** action has many predefined fields that cover all of the information available for a file. See Table 2 for an incomplete list of options. Find also has a **-fprintf** switch which will send the output to a file, like **-fprint**.

Table 2. printf Options

Escape Sequences\a - Alarm Bell**\b** - Backspace**\f** - Form Feed**\n** - Newline (not provided automatically)**\c** - Carriage return- Horizontal tab**\v** - Vertical tab**** - A literal backslash**\c** - Stop printing and flush output

Formatting Sequences%b - File size in 512 byte blocks**%k** - File size in 1k blocks**%s** - File size in bytes**%a** - Access time in standard format**%A** - Formatted access time (see man page for options)**%c** - Status time in standard format**%C** - Formatted status time (same a %A)**%F** - Type of filesystem**%p** - File name**%f** - File name with path removed**%P** - File name with find argument removed (file instead of ./file)**%u** - User name**%g** - Group Name

(See the man page for complete listing)

A third option for output is **-ls**. This option produces a listing of files that is the equivalent of the output from **ls -idls**. The **-fls** option will send this to a file.

Of course, simply producing formatted lists of files is not the limit to find's usefulness. Find also allows us to execute commands on them with **-exec** and **-ok**. **-exec** executes a command for each file that matches.

Our earlier example demonstrates a common use for the **-exec** option: deleting old and unused files.

```
$ find /tmp/* -atime +10 -exec rm -f {} \;
```

After the **-exec** switch itself, we specify the command, any options (such as the **-f**), and then **{}**, which represents the matched files. The command line must then be terminated with **;** (the **** is to prevent shell expansion).

```
$ find . -type f -exec grep -l linux {} \;
```

would execute the command **grep -l linux** on all regular files in and under the current directory.

The **-ok** switch operates the same way, but will prompt the user for confirmation before executing the command on each file.

```
$ find . -ok tar rvf backup {} \;
```

This command will descend through the current directory and below, asking the user which files should be added to the tar archive “backup”.

This leads us into some practical uses for find.

Sometimes it's necessary to duplicate a directory or directory structure. For this purpose many users utilize the cp command with the **-r** option. However, this command does not always create an exact copy!

Create a directory with a file and a link in it.

```
$ mkdir test
$ touch test/bar
$ ln -s /vmlinuz /test/foo
$ ls -l test
-rwx--x--x eric staff 0 Sep  9 bar
lrwxrwxrwx eric staff 8 Sep  9 foo -> /vmlinuz
```

Now copy it with **cp -r**

```
$ cp -r test test1
$ ls -l test1
-rwx--x--x eric staff      0 Sep  9 11:18 bar
-rw-rw-r-- eric staff 318436 Sep  9 11:18 foo
```

The cp command followed the soft link and copied the kernel into the new directory!

Let's try a different approach:

```
$ rm -r test1
$ cd test
$ find -depth -print | cpio -pdmv ../test1
$ ls -l ../test1
-rwx--x--x eric staff 0 Sep  9 bar
lrwxrwxrwx eric staff 8 Sep  9 foo -> /vmlinuz
```

This method uses cpio to copy files to the new directory. Find produces the file list by descending the directory structure. Even though our example was only one directory deep, we know that find can descend an entire directory structure. We also know that we can also control which directories it descends and which files it outputs.

In the above command I added the -depth option. It insures that directory names are output before the files in them. This allows cpio to create the directories before trying to copy files into them.

The cpio command is another multipurpose tool in the Unix toolbox. It can create archives in a variety of formats and also extract from them. It also handles the output of find's **-print** option perfectly. Combined, these tools could

form a simple backup system. (Please note: I am presenting this purely as an example. Systems that support many users or that have irreplaceable data on them should use more extensive and robust backup systems.)

```
$ find . -depth -print \
  | cpio -ov --format=crc > /dev/fd0
```

find reads the contents of the current directory, and the filenames are piped to cpio, which copies the files to the floppy in the System V R4 archive format with CRC checksums. (This format is preferred to the default since it is platform independent, supports larger hard disks, and provides at least simple error checking.)

When cpio reaches the end of each floppy it prompts us with:

```
Found end of tape.  To continue, type device/file
name when ready.
```

In order to continue, type:

```
/dev/fd0 RETURN
```

Of course, if you are lucky enough to have a tape drive or other storage system, you may not have to do this, though cpio can also span tapes if the archive does not fit on one.

This system does have at least one drawback: if the data to be stored will not fit on one unit, the backup cannot be fully automated.

The first backup of my home directory spanned ten floppies. I reviewed the contents and noticed two subdirectories that probably were not worth backing up, so I altered find's arguments:

```
$ find . \
  \( -path ./.netscape-cache -o -path ./lg \)\
  -prune -o -print | \
  cpio -ov --format=crc > /dev/fd0
```

This introduces some more find options. The **\(** and the **\)** are parentheses with **** to prevent shell expansion. Find allows parentheses to logically group expressions. This was necessary since I have two expressions in the command

```
\( -path ./.netscape-cache -o -path ./lg \)
```

Inside the parentheses we have two **-path** statements separated by **-o**. This is a find “or” statement.

```
\( -path ./.netscape-cache -o -path ./lg \) -prune
```

Find's **-prune** option causes find to not enter a directory. Therefore, we can translate the above to “If the path is ./.netscape-cache or ./lg do not descend into the directory.”

After this clause we see another **-o** statement. If the file does not meet the criteria for pruning, it is printed instead.

So, my entire home directory with the exception of my Netscape cache and lg directory is now backed up.

This is fine for an initial backup. But what about next week when I want to backup my directory, but I've only really touched a few files?

```
$ find . \
  \( -path ./.netscape-cache -o -path ./lg \) \
  -prune -o \( -mtime -7 \) -print | \
  cpio -ov --format=crc > /dev/fd0
```

This adds one more clause: “If the file is not under the netscape cache or the lg directory, check if it has been modified in the past 7 days. If it has, then print the name.” The name is then sent to cpio to archive.

Obviously these command lines can get very complicated. It's usually a good idea to test them by piping the output through more before using cpio.

In addition to **-o** find also has an “and” operator, **-and**, and a negation operator **-not**. When multiple match criteria are specified, **-and** is implied.

```
$ find -mtime -5 -type f -print
```

prints files that have been modified during the last five days **and** are regular files.

```
$ find -mtime -5 -not -type f -print
```

prints things that have been modified during the last five days that are **not** regular files: directories, soft links, etc.

But wait, disaster has struck! Your (sister, son, daughter, little brother, mom, spouse, whoever) has deleted a very important file! Time to use that backup.

```
$ cpio -t < /dev/fd0
```

produces a table of contents from the archive. As it does during backup operations, cpio prompts for the next disk while it reads the table of contents.

```
$ cpio -i core < /dev/fd0
```

The -i switch tells cpio to extract the named file. The absence of a file name cause cpio to restore the entire archive.

System maintenance tasks can also be simplified with find. Our second example demonstrated using find to clean out older files.

```
$ find /home -name core -o -name foo \
  -exec rm -f {} \; 2> /dev/null
```

This command cleans out any core dumps or files named “foo” from home directories. (Although some files named “foo” can be very important!)

```
$ find /var/adm/messages -size +32k \
  -exec Mail -s "{}" root < /var/adm/messages \;
  -exec cp /dev/null {} \;
```

This is another example from the crontab on my Caldera/Red Hat system. It uses the implicit “and” function to mail the system messages file to root and then empty it.

Find also has an important security application. Two of the file modes that I did not cover earlier are SUID and SGID. These modes provide a user with the rights of the owner or group of a program when the program is executed.

An example of this is the passwd program. This program allows users to change their password. In order to do this the /etc/passwd (or /etc/shadow) file must be modified, which is a function only root should be able to perform. Since the passwd program belongs to root and has the SUID mode set, it can modify the necessary file. When passwd completes the user's rights return to normal. The passwd program is responsible for making sure the user can't do anything wrong while acting as root.

```
$ ls -l /usr/bin/npasswd
-r-s--x--x 1 root /usr/bin/npasswd
```

(/usr/bin/passwd is linked to /usr/bin/npasswd on my system.) The **s** in the execute field for owner signifies SUID. A SGID program would have **s** in the execute field for group.

This mechanism has obvious security implications. A user (or invader) who has compromised a system could install a program (such as a shell) with this mode set and then do whatever they wish whenever they want by running that program.

In octal notation SUID is expressed as 4000 and SGID is 2000, so

```
$ find / -perm 4000 -print
```

produces a list of SUID files on a system.

```
$ find / -type f \( -perm 2000 -o -perm 4000 \) \
  -print
```

produces a list of regular files that have SGID or SUID mode set.

This list could be saved to a file (with **-fprint**) and compared each day with the output from the previous day.

This article does not cover every option for find. This was also only a cursory explanation of filesystems and access modes. Hopefully, I was able to provide you with enough information to make using Linux a little easier and a lot more rewarding.

Resources

Essential System Administration by Æleen Frisch, O'Reilly and Associates

Practical Unix Security by Simson Garfinkel and Gene Spafford, O'Reilly and Associates

The manual pages.

Eric Goebelbecker is a systems analyst for Reuters America, Inc. He supports clients (mostly financial institutions) who use market data retrieval and manipulation APIs in trading rooms and back office operations. In his spare time (about 15 minutes a week...), he reads about philosophy and hacks around with Linux. He can be reached via e-mail at eric@cnct.com.

Archive Index Issue Table of Contents

Advanced search

New Products

LJ Staff

Issue #20, December 1995

Tcl and Tk Reference Card, Soft Braille for Linux: BRLTTY 0.22 BETA Release and more.

Tcl and Tk Reference Card

Specialized Systems Consultants, Inc. (SSC) announced the publication of the Tcl Pocket Reference and the Tk Pocket Reference. The Tcl ("tickle") Reference describes Tcl 7.3; the Tk ("tee-kay") Reference describes Tk version 4.0. Tcl is a small, embeddable, extensible scripting language. Tk is a Toolkit of widgets, which are graphical objects similar to those of other GUI toolkits such as Xlib, Xview and Motif. When Tcl and Tk are used together, the Tcl/Tk programming system can be used to rapidly build useful applications (such as adding X-based front-ends). The Tcl and Tk References are sold individually for $3.00 each or together as a package (ISBN: 0-916151-80-8) for $4.50.

Contact: SSC, Inc. P.O. Box 55549, Seattle, WA 98155 (206) 782-7733 Fax: (206) 782-7191 E-mail: info@linuxjournal.com. URL: http://www.ssc.com/

Soft Braille for Linux: BRLTTY 0.22 BETA Release

Nikhil Nair and James Bowde announced the first public release of BRLTTY, a software system to allow access to the console of a Unix system for users of soft Braille displays. BRLTTY requires a Linux system with kernel version 1.1.92 or later. BRLTTY only works with text-mode applications. The package is available at sunsite.unc.edu in the directory /pub/Linux/utils/console.

Contact: nn201@cam.ac.uk (Nikhil Nair) or jrbowden@bcs.org.uk (James Bowden)

Porting Linux to the DEC Alpha: The Kernel and Shell

Jim Paradis

Issue #20, December 1995

Last month, Jim described the task of porting the kernal and basic environment needed to get to the shell prompt. In this last of 3 parts, he tells us about building a real environment.

Achieving the shell prompt was only the beginning of the trial, not the end. Now we had to get other utilities working so as to allow us to debug more of the system and to make it into a real, usable UNIX-like system.

I set Brian Nelson to the task of porting more utilities, starting with the “fileutils” and “shellutils” subsets of the MCC 1.0+ distribution. Meanwhile, I realized that better debugging tools would expedite the debug process, and I started thinking about implementing some kind of remote debugger support for gdb. My first implementation was under the ISP CPU simulator. The reason for this is that I could add code to ISP to examine and modify the state of the machine and communicate with a debugger without having to code a breakpoint handler into the kernel itself.

GDB has a remote debugging protocol built into it; all I needed to do was to add code to ISP to respond to gdb commands and to encode the simulated machine state for GDB's consumption. Getting this all working was only a few days' work.

Device Drivers

While all this was happening we realized that Linux/Alpha was turning into a serious project and that we could use some help in the device driver space. While console callback drivers had served admirably to get us up and running, they were not equal to the task of supporting a production system. We recruited Jay Estabrook from the Digital UNIX group in this capacity, and he has proved to be an immensely valuable addition to our team. Within his first two

weeks with the group he produced a native text-mode VGA driver and a native keyboard driver for Linux on the DEC 2000 AXP (“Jensen”) series.

Porting device drivers to Linux/Alpha presents some interesting challenges. Fortunately, many of the problems only needed to be solved once and the results would be applicable to many different drivers.

The Alpha CPU has no concept of I/O bus access; there is no Alpha equivalent of the Intel inb/outb instructions for communicating with the I/O bus. In order to implement a PCI- or EISA-based Alpha system, some sort of “glue” logic is needed to translate Alpha load/store accesses into I/O bus accesses. On systems based on the DECchip 21064 and 21164 CPUs, this glue logic is implemented in an external chipset (DECchip 21071 series). On systems based on the DECchip 21066, this glue logic is built into the CPU. This glue logic sets aside certain areas of the system physical address space for communicating with the I/O busses. To perform a bus access, one takes such information as the I/O port number and the size of the transfer and encodes it into a special memory address (This encoding is different for the different glue logic implementations). One may then load the data from or store the data to this address.

Interrupt handling is also different on Alpha systems. On most Intel-based systems, the mapping between bus IRQs and interrupt vectors is fixed and straightforward, and the hardware dispatches directly to the interrupt vector associated with a particular IRQ. On Alpha, all interrupts are dispatched by PALcode. The Digital UNIX PALcode vectors all device interrupts to a single routine. One of the arguments to this routine is a number (called the “SCB vector” for reasons I don't need to go into here) indicating which interrupt was received. Unfortunately, the mapping between bus IRQs and SCB offsets is not the same across all platforms. This means that we need extra code in the interrupt handling path to map the SCB vector back to an IRQ number. In fact, there are different versions of the mapping routine for the different platforms.

It turns out that many Intel Linux device drivers rely on the fact that the BIOS puts the device into a known state before the operating system even sees it. We discovered this when we were debugging the interrupt handling for the keyboard driver (and, later on, the SCSI driver). Apparently the interrupt controller on Intel boxes is initialized by the BIOS to trigger on the *transition* of an IRQ line (edge triggering) rather than on the *state* of an IRQ line (level triggering). We were having no end of problems with “spurious” interrupts until we added code to the CPU initialization routine to set the interrupt controller properly.

The Miniloader

While our early decision to use the SRM console was good to get the project off the ground, it turns out that the SRM console is not the best choice for Linux. First of all, the SRM console is memory-hungry because it must implement numerous functions required by OpenVMS and Digital UNIX. Many of these functions are not needed by Linux. Second, and more serious, is the fact that the SRM console is not freely redistributable. Digital charges a substantial license fee to third parties for resale rights to the SRM firmware, as well as a per-unit charge for each copy of the SRM firmware sold by Digital or a third party. While the end-user typically never sees these charges, they do serve to raise the price of the hardware when it is sold in a Digital UNIX or OpenVMS configuration. In addition, requiring the SRM console for Linux/Alpha would present a significant burden to clone vendors who wished to build and sell Alpha systems for the Linux marketplace.

For these reasons, we decided to investigate the possibility of developing a freeware “miniloader” for Linux/Alpha. The miniloader could be much smaller than the SRM console because it needs to implement only the functionality necessary to initialize the system, load the PALcode, and load Linux. It could also be freely redistributable in source and binary form.

Unfortunately, developing a replacement for the console firmware is a non-trivial undertaking. Fortunately, however, we had help in the person of Dave Rusling from the Digital Semiconductor facility in Reading, England. Dave had much experience in low-level hardware support on the evaluation boards produced by Digital Semiconductor, and he had already done significant work for the Linux/Alpha project in the area of PCI support. He eagerly took on the task of developing the miniloader.

The miniloader consists of system initialization code, an OSF-compliant PALcode library, a bootstrap loader, and console callback routines. It presents to the bootloader and kernel an interface similar to that seen in the SRM console, with stripped-down functionality. The miniloader only implements those SRM features and callbacks that are used by Linux. As of this writing, the miniloader has succeeded in booting Linux on several models of Digital Semiconductor evaluation boards, as well as on the low-cost AXPpci/33 “NoName” motherboard and the high-performance “Cabriolet” motherboard.

Cutting Over to 1.2

While we were working on our 32-bit port, Linus was toiling away in Helsinki on his 64-bit Alpha port. We knew that we would want to cut over to using his code base at some point so that we would be in sync with the rest of the Linux community. The main question was *when* we would do the cutover. Our port

had more demonstrable functionality earlier (e.g. device support, networking, utilities), but Linus was catching up fast. We decided to continue working on the 32-bit port for demonstration purposes while keeping track of Linus' progress, and we would cut over to Linus' code base when doing so would yield a system of roughly equivalent functionality.

This point came in March, 1995 when Linus posted a message to the linux-alpha mailing list with the subject “self-hosting linux on ftp.cs.helsinki.fi”. While one of our own major goals was to have a self-hosting Linux/Alpha system, we had not been able to realize it due to the immense complexity of porting the GNU compiler suite in our cross-development environment. Linus very neatly sidestepped the entire cross-build issue by making his Linux/Alpha system calls compatible with their Digital UNIX counterparts. Therefore, he could achieve self-hosting simply by running the compilers from his Digital UNIX system on his Linux/Alpha system.

While this self-hosting environment did not meet our criterion of being 100% freeware, it was a useful starting point. Instead of building the GNU tools in a cantankerous cross-development environment and testing them on an immature operating system, we could prototype and debug our *entire* development environment on a Digital UNIX system. When we were satisfied with its functionality, we could then copy it over to a Linux/Alpha system with reasonable assurance that it would work. This, in fact, is exactly how we put together the self-hosting demo that we exhibited at DECUS in May, 1995 (“Linux at DECUS”, *Linux Journal* issue 15, July 1995).

Getting By With a Little Help From Our Friends…

An operating system is much more than just a kernel, as any of the creators of Linux distributions could tell you. In order to be able to provide all of Digital's Linux/Alpha contributions to the Linux community free of charge, we necessarily had to limit the investment we made in the project. As of this writing, Digital is funding three full-time engineers, a part-time product manager, a part-time technical writer, and several loaner Alpha systems (including the Alpha systems that Linus has been using). In my project plan outline for Linux/Alpha I pointed out that Linux was unique in that we could do the project with such limited resources. Given the history of Linux, I reasoned, once the Linux/Alpha code became available, developers all over the net would add functionality and fixes. My prediction turned out delightfully true. Several people, both inside and outside of Digital, made significant contributions to the project at no cost to Digital. The result is of enormous benefit to both Digital and the Linux community as a whole.

Linus Torvald's own contributions, of course, are legendary. I mention him here because without his tireless work the project would have taken a different turn and probably would not be as successful as it is today (Linus, if you're reading this, we *could* use a little breathing room between releases. At least let me finish *compiling* one release before you turn out the next!)

Another major champion and supporter of the Linux/Alpha project is David Mosberger-Tang of the University of Arizona. He was literally the first on his block to own an Alpha-based AXPpci/33 motherboard, and he provided all of the initial patches to enable both the 32-bit and 64-bit kernels to function on that platform. He has also been a valuable resource and a second set of eyes to assist in untangling sticky problems. In addition, he has ported numerous system and utility packages that would have taken us days or even weeks to do ourselves in our spare (ha!) time.

It has been said that “any sufficiently advanced technology is indistinguishable from a rigged demo,” and this could certainly be said of the DECUS demo that we staged. While the toolset was capable of building and linking the kernel, the 64-bit C runtime library was not yet stable enough to build user utilities. Fixing this was on our “to do” list along with a lot of other things, but it turned out we didn't have to. Shortly after we released the 64-bit development tools to Digital's FTP area, Bob Manson of Ohio State University released a working 64-bit library based on our earlier 32-bit work. Bob also released several useful sets of utilities that, again, it would have taken us weeks to get around to porting on our own. He is also rumored to be working on modifying gcc to generate floating-point code that is capable of recovering from exceptions.

The BLADE Releases

After showing off Linux/Alpha at DECUS, it became clear that some kind of end-user-installable distribution was needed. At that point, Linux/Alpha resembled in some ways the early days of Intel Linux: the “system” consisted of a motley collection of source and binary archives scattered over several FTP sites on different continents. Putting together a running system out of these pieces was a job only a dedicated hacker would be willing to go through with.

One difference between then and now is that now there are high-quality commercial Linux distributions (Plug-And-Play, Red Hat, and Slackware, to name just a few) that can serve as the basis for equivalent Linux/Alpha distributions. We knew, though that it would be some time before these distributions were ported and qualified for Linux/Alpha. In oder to sustain the momentum built up at DECUS, we needed some kind of Linux/Alpha distribution sooner than that. That's when we decided to embark on a project designed to become obsolete: BLADE. BLADE stands for **B**asic **L**inux **A**lpha

Distribution **E**xpletive (I picked the name by starting with “LAD” for “Linux Alpha Distribution” and grepping through /usr/dict/words for this combination and playing acronym games with some of the results). BLADE is a Linux/Alpha kit that can be installed without needing to build kernels or use a host development system.

BLADE was designed to be deployed quickly, and it's pretty rough around the edges. There's only one automatic installation script, called install_subset. A lot of steps that are done automatically by other distributions must be handled manually in BLADE. We provide full step-by-step instructions, though, so the user knows what steps need to be taken.

The first release of BLADE (V0.1) provided basic functionality and a development system in character-cell mode. There was no networking, no GUI, and a limited utility set. However, it was self-hosting, and it came with kernel sources and the gnu compiler suite. One could build the kernel or any utilities one desired using BLADE. In fact, we used BLADE V0.1 as our primary development system for BLADE V0.2. BLADE V0.1 was based on a modified 1.2.8 kernel and supported the AXPpci/33 only.

The second release of BLADE (V0.2) added more utilities and networking functionality. Graphics were still not available, but one could perform development and basic networking (ftp, telnet, rlogin) in character-cell mode. BLADE V0.2 was also the first release to support the Linux/Alpha Miniloader (aka MILO/Alpha) on the AXPpci/33. MILO is a drop-in system firmware replacement that allows the user to boot and run Linux without requiring the SRM console firmware. BLADE V0.2 also added a kernel boot disk for the Digital Semiconductor 275-MHz EBPC/64 evaluation motherboard. This is the fastest system to date that supports Linux. BLADE V0.2 is also based on the modified 1.2.8 kernel.

Currently under development is BLADE V0.3. BLADE V0.3 will be based on a 1.3 kernel and will add support for the X window system (see below). It should also support more system types.

X Marks The Spot

Thanks to the tireless efforts of my colleague Jay Estabrook, as of this writing X11R6 is up and running on Linux/Alpha. Most of the standard libraries and client executables are in place. At the present time, only the S3 server has been ported, and it has only been qualified on a few video cards. The current plan is to let this be a sample server and solicit other parties (e.g. the XFree86 Consortium) to port other servers.

One major problem with supporting multiple video cards has to do with the on-board ROM BIOS that many cards have. This BIOS typically contains code to initialize the card and to set video modes. Unfortunately, this BIOS is nearly always written in 80x86 assembly code. To execute it on an Alpha system requires an Intel execution engine. We are investigating several strategies to provide this functionality as part of MILO/Alpha in source form, and rumor has it that our old friend David Mosberger-Tang has made good progress in this area.

Surf's Up!

As I write this, Linux/Alpha is being exhibited at UNIX Expo in New York City in all of its X-windows glory. We have ported the freeware web-browser **chimera**, and these systems are available for surfing the web and for connecting to remote systems on the Internet via rlogin, telnet, and ftp. In fact, on setup day several people from non-net-connected boths came by to use these systems to retrieve forgotten files from their home systems. This includes yours truly, embarrassingly enough. We needed to connect to the serial port on our PC64 system to use the ROM debug monitor, but the version of Linux/Alpha on the other system did not have kermit, cu, tip, or any other terminal emulation or serial connection program. No problem: I ftp'ed over to David Mosberger-Tang's archive at ftp.azstarnet.com and retrieved the Linux/Alpha version of kermit. We were in business in a few minutes.

In short, Linux/Alpha is starting to feel like a real Linux system!

Future Directions

Despite our great progress, much work remains to be done on Linux/Alpha:

- As mentioned above, we need to deploy some sort of BIOS-emulation facility so that we can execute the proprietary initialization code on some expansion cards. While initial code exists and works, it does not support the real-mode 32-bit instructions that are used in the BIOSes of some cards.
- We need to tackle the great unsolved problem of floating-point exception handling. Programs that are floating-point intensive are not likely to work until this is done.
- We need to write a character-cell driver and an X server for the TGA graphics adapter that is provided with Multia and several other Digital Alpha systems.
- We desperately need shared libraries! As of this writing, the statically-linked executables in Linux/Alpha are rather large (around 200Kb for a

typical utility, several megabytes for the X server). Shared libraries will decrease both disk space requirements and virtual-memory usage.

- We need to work on compiler optimizations. The Alpha support in gcc does very good optimizations in some places, not so good in others. In addition, the compiler does not yet take advantage of Alpha's multiple-instruction issue feature. This feature allows more than one instruction to be issued per clock cycle, but only certain combinations are allowed. By carefully rearranging the instructions in the executable, one can take advantage of this feature and achieve significant performance improvements.

All in all, we are excited about the future. Linux/Alpha, even in its relatively primitive state, feels like a real Linux system. Addressing the above areas can only make it better!

Jim Paradis works as a Principal Software Engineer for Digital Equipment Corporation as a member of the Alpha Migration Tools group. Ever since a mainframe system administrator yelled at him in college, he's wanted to have a multiuser, multitasking operating system on his own desktop system. To this end, he has tried nearly every UNIX variant ever produced for PCs, including PCNX, System V, Minix, BSD, and Linux. Needless to say, he likes Linux the best. Jim currently lives in Worcester, Massachusetts with his wife, eleven cats, and a house forever under renovation. He can be reached via e-mail at paradis@sousa.amt.tay1.dec.com and on the WWW at www.iii.net/users/jrp.html

Archive Index Issue Table of Contents

Advanced search