LINUX JOURNAL

Advanced search

Linux Journal Issue #23/March 1996

Features

Linux Distributions Compared
There are more Linux distributions than ever. Our editor takes you on a tour to help you decide which is best for you.
Linux Distributions Features Comparison Chart
Graphic Formats for Linux *by Gerald Graef*
What ARE all those graphics storage formats and which should you use? Gerald Graef explains the methods and benefits.

News & Articles

Geomview *by Tim Jones*
Want a way to view and animate 3-D objects? Then Geomview may help you.
Ext2tools for Linux *by Robert A. Dalrymple*
How to use Windows and Linux on the same PC while losing neither your mind nor your files.
Running Windows Applications NOW *by Ron Bardarson*
Need to run MS Windows Applications right away? Then DESQview/X offers a solution.

Columns

Kernel Korner Dynamic Kernels - Modularized Device Drivers
Letters to the Editor
Stop the Presses
Take Command The cpio command

Directories & References

Linux Distributions Compared

Linux Journal Staff

Issue #23, March 1996

Been considering installing or re-installing Linux on a PC? Confused by the wide range of distributions available? This article may not solve all your woes, but it should at least put you on the right track.

Why doesn't *Linux Journal* recommend one particular Linux distribution as the best? Why not do quarterly awards for the best current Linux distribution? There are at least two good reasons and one not-so-good reason. The good reasons are that different users have different needs, and that a “best” rating for any one distribution would unfairly penalize the other distributors. The not-so-good reason has been that *Linux Journal* hasn't had the resources to do comparative review.

While we still will not recommend one “best” distribution, we have recently acquired hardware specifically for testing distributions. While we can't begin to buy a full set of hardware that will allow us to test distributions on a wide range of hardware, we can assume that since all the distributions use the Linux kernel with few or no modifications, there are not likely to be too many differences based on hardware. By buying mainstream hardware (see The Test Platform sidebar), we can ignore the hardware problems that do not generally differentiate distributions and concentrate on the other characteristics of the distributions.

You Be the Judge

We are not even going to attempt to rank the distributions. We will introduce the distributions that are the most common in the U.S. as this is written; we will introduce other distributions in future articles. As we explore more distributions, we will update our table for feature comparison, and future articles will include the entire table for all the distributions tested so far for easy comparison.

Our comparisons will be designed to allow you to rank the distributions based on your own needs, rather than our perceptions of your needs. To do this, we will provide a description of each distribution, as well as comparison charts of features so that you can easily see the trade-offs that various Linux distributors have made.

Is this a cop-out? Are we shirking our duty? It has become clear to us that it would be hard for *Linux Journal* to rank the distributions; our staff members have different personal favorite distributions and defend their choices rationally—occasionally even argumentatively. At *Linux Journal*, we use Linux extensively—nearly exclusively—and we still don't agree which distribution is the best. We have different priorities, skills, and expectations, and we believe this is true of our readers as well.

We have reason to think that our readership is thoughtful and intelligent, and we have certainly been informed in *many* letters to the editor that our readers appreciate being given a chance to form their own opinions. So instead of attempting to make up your mind for you, we would like to give you as much material as possible to use to make up your own mind.

Version numbers

Many new Linux users confuse the version of the distribution they are using with the version number of the kernel they are running. As described in the What's a distribution? sidebar, the Linux kernel is just one of the many pieces of software needed to create an entire distribution. Each distribution uses version numbers of its own to keep track of the state of the entire distribution, which has more to do with the collection of programs than with the particular kernel involved. Indeed, many distributions have included two or more different kernel versions in one version of the distribution.

However, it's worth understanding the version numbers used for the Linux kernel itself, since the kernel is a key part of any Linux distribution. Kernel version numbers come in three parts: the major version number, the minor version number, and the patch-level. The Linux kernel is being constantly developed by a large team of developers, and while they add new features, they occasionally introduce new bugs. To keep this from causing a problem for Linux users, the developers periodically dedicate several months to fixing bugs and creating a very robust, stable kernel. When this is done, a stable version is released with an even-numbered minor version number. The developers then begin adding features (and temporarily breaking things sometimes) in development versions with odd-numbered minor version numbers.

Unless you want to live on the “bleeding edge” of Linux development, you will probably want to stick with the latest stable kernel version. As of this writing, the latest stable kernel version is 1.2.13; by the time you read this, preparations will probably be underway for 1.4.0.

Value Added

Several vendors are adding value to existing distributions in various ways. For example, Caldera is adding commercial components to Red Hat Commercial Linux to create their Caldera Network Desktop. WorkGroup Solutions used to enhance Slackware as the base for WGS Linux; they have now switched to basing their work on Red Hat Commercial Linux. Trans-Ameritech sells disks with several distributions, including Slackware and Debian; they base their own value-added work on Slackware, trying to make it easier to install.

Value-add distributions are worth serious consideration, but they aren't the subject of this review. As part of deciding which distribution to get, you may want to consider what you can get from value-add vendors as well as from the base distributions in question. Most value-add vendors (as well as distribution vendors) advertise in *Linux Journal*.

Binary File Formats

One of the most common sources of confusion in the Linux world today involves binary file formats (See What is a binary file format? sidebar). The Linux community is in a state of transition from the old “a.out” binary file format to the new “ELF” binary file format, which has many features that are completely missing in the a.out format.

ELF is the binary file format used by Unix System V Release 4, but that doesn't mean that a Linux binary in the ELF file format is compatible with SVR4, nor does it mean that SVR4 binaries can run on Linux. The capability for Linux to run some SVR3 and SVR4 binaries is provided by the iBCS2 compatibility package, which most distributions include.

One of ELF's features is extensibility; with ELF, it is possible for developers to add features that weren't thought of when the format was first designed. For instance, one Linux developer has noted that he could add icons to ELF executables without breaking any software. Icons weren't considered when ELF was developed, but the format is extensible enough that they can be easily added.

But perhaps you don't care if you can add an icon to your ELF binary, or even if anyone else can. What does ELF do for you? Fundamentally, it makes life much easier for Linux developers. It also has a few esoteric features which make it

practical to support some software under Linux that was previously impractical to support. So it gives you more and better software available for Linux.

Since the whole Linux community is moving to ELF, you don't want to get stuck with a distribution which does not support ELF. A year from now, it will be almost impossible to find a.out binaries for programs you want, and important bug fixes may only be available for ELF.

Because of this, we have only reviewed distributions which at least support ELF binaries. The only distributions that are reviewed here which are not **based** on the most current ELF libraries are currently being updated, and should be based on standard ELF libraries by the time you read this.

What do we mean by **based**?

ELF-based means the entire distribution, or at least, almost the entire distribution, consists of ELF binaries. a.out binaries are not provided with the system, or if they are provided, they aren't part of the “core” of the system or are not available in ELF format.

By contrast, “supports ELF” means that, while the distribution is partially or completely built of binaries in the a.out format, the ELF **programming libraries** are included so that ELF binaries will also run.

Media

Many distributions are available solely on CD-ROM. There are several reasons for this:

- So much software is available for Linux that it is impractical to provide it all on floppies.
- It is much easier to install software from CD-ROM than to change floppies once a minute.
- A cheap supported CD-ROM drive costs approximately the same as the stack of floppies needed to install a complete Linux distribution.

However, some distributions (including Debian, Red Hat, and Slackware) are available via FTP over the Internet in a form designed to fit on floppies, and Linux System Labs still offers a floppy distribution service, from which you can order an entire Linux distribution on floppies, though it is the only company left, to our knowledge, which does this.

Copyright Considerations

Most distributions are freely available over the Internet, although only some are actively distributed in a way that makes it feasible to install them directly from the Internet. You should be aware that while distributors can restrict you from running commercial software components on more than one machine, they cannot restrict you from installing the base Linux software on multiple machines. A Linux distribution which is packaged in a form which cannot be installed without installing proprietary software that is under copyright licensing terms more strict than the GNU General Public License is probably in violation of copyright law. [And should be completely avoided, in my strongly held opinion—ED]

To sum up, you should feel free to install any Linux distribution you buy on as many machines as you like, as long as you respect the licensing terms of any proprietary software that is included with the distribution. Conversely, the vendor needs to make it possible for you to do so without taking unreasonable steps.

If you encounter a vendor who makes it difficult or impossible to install free components without installing proprietary components that have restrictive licenses, first politely bring the issue to their attention—they may not have considered the issue. If the vendor refuses to resolve the issue, return the distribution, ask for your money back, and send a letter to the Editor of *Linux Journal*.

The Future

By the time you read this, some of the distributions will have made new releases, and so some of this description will be out of date. We hope all the bugs will have been fixed, for instance...

We would like to keep tracking distributions as they improve and, occasionally, publish descriptions and the feature comparison table in updated form.

Missing Bugs

One thing you may notice missing from this overview is a list of bugs in each distribution. In general, we assume that some noticeable bugs are inevitable in software that is evolving as quickly as Linux and that vendors will be responsive in tracking and eradicating bugs. If they aren't, word of mouth between users will be more effective than any sort of complaining on our part. We also want to avoid showing any bias, and if we start listing bugs we find, we will either show bias or appear to show bias simply by the “choice” of bugs we publish.

Since we can't do a fair comparison of the number and severity of bugs between distributions, we have tried to note only bugs that involved completely missing functionality or which were difficult to work around and hard for the user to avoid.

If working around simple bugs is a problem for you, we urge you to purchase a distribution from a vendor who offers technical support for the installation process. By providing installation technical support, these vendors not only help you with the problems, but force on themselves a vested interest in fixing those bugs. Providing support is expensive, and solving the same problem over and over again is a waste of their time and money.

At the same time, if you buy installation technical support, please understand that you aren't buying a lifetime warranty for every part of the system. None of the vendors here provide that as part of their base product. You can buy very comprehensive support packages from several vendors if you need them. Happy shopping!

Debian .93R6

The only distribution in this lineup which is not currently developed on a commercial basis (Slackware was originally also non-commercial), Debian has been a long time in the making. Unlike all the other Linux distributions, Debian is put together by a large team of volunteers all over the world. While a few people are in charge of a very small core of the distribution, almost all the decisions are made by consensus (which has given Debian a reputation for vaporware, since consensus can take some time to achieve), and nearly all of the packages are developed independently by members of a large development team.

By following a strict set of rules and using the most powerful packaging technology currently available for Linux, this large team has achieved a remarkably coherent Linux package. As this article is being written, Debian is still the **only** distribution which has complete *dependencies*: when you install one package, it checks to see if other needed packages are also installed. It also checks version numbers if requested; package **A** can insist that package **B** be installed, and that package **B** be version *x* or greater. This makes upgrading any package nearly foolproof.

These dependencies are a natural requirement of Debian's distributed development and are well-tested, because many of Debian's users and developers upgrade parts of their systems on an as-necessary basis, which exercises the dependency-checking features considerably.

The Debian installation procedure does not have as many of the user-friendly assumptions as many other distributions. It won't guess at which disks you wish to use or which partitions on your disks you wish to use for swap and root, for instance. It doesn't give you the option to skip the check for bad blocks while making a file system. It installs the entire “base” system from floppy (three floppies), and then you reboot into the installed system with enough programs installed to install any other packages you like. This way, all the base system has to support is floppies and hard drives, and then you can choose which kernel modules to load during the initial “base” installation; those determine what hardware is supported. Debian is highly modularized.

Debian is one of the strictest distributions about setting passwords. As soon as you re-boot the “base” system, you are prompted for the root password, with an explanation of how to choose a good password. Also, part of adding a new user under Debian is setting the user's password.

All of Debian's packaging tools run in text mode, so X is not required to use any of Debian's sophisticated upgrade capabilities. They can be used in an xterm or rxvt on the console, over a modem link or over the network. The package tools are also very fast.

The default Debian kernel is highly modularized. This means that you rarely need to recompile the kernel, and you don't have lots of unnecessary, unused drivers in the kernel taking up memory and causing possible conflicts. Where Red Hat has 72 possible boot disks in its standard set, Debian has 1—but Red Hat requires you to make only 3 disks to install, whereas Debian requires you to make 5 disks, and have an extra blank disk available, which is used to make another boot disk during the installation process (you can re-use the boot or root disk if you are brave, but we don't recommend it).

Debian is currently being transformed from an a.out-based distribution with optional ELF support to a fully ELF-based distribution. By the time you read this, a new release of Debian fully based on ELF should have been released; most of the Debian **packages** have already been made available in ELF format. Debian 1.1 will be fully ELF-based. (Due to a mis-communication, a version called Debian 1.0 was released on the November Infomagic CD. That version was incomplete and will not work if you attempt to install it. The real release has been renumbered 1.1 in order to avoid confusion between the official ELF-based release and the one accidentally included on the Infomagic CD.)

Debian's home page can be found on the Web at www.debian.org.

Craftworks Linux 2.0

This relative newcomer SoftCraft Linux, by SolutionS R Us) has relatively few packages, each of which contains many files. Craftworks Linux comes with a boot floppy, a CD-ROM, and a manual.

We ran up against one bit of confusion right away: as soon as we booted from the provided floppy, an unclear copyright message was presented that seemed to appropriate title to Linux and seemed to attempt to legally restrict the user from installing the product on more than one machine.

Following our own advice, we contacted Craftworks and asked their intention. They confirmed that their intention was only to protect their own proprietary software included with the distribution, and that they allow users to install the product on as many systems as they wish. They do claim compilation copyright on the CD-ROM and the floppy, which means that they restrict the user from making any verbatim copies of either the CD-ROM or the floppy except for backup purposes. However, they in no way intend to restrict re-distribution of any included free software, and they promised to resolve the issue by making their copyright licensing terms and notices much clearer in future versions of the product.

The installation was fairly simple; Craftworks provides three pre-selected sets of packages to install (other packages can be installed separately after the base installation has been completed). This makes it quite simple to install a simple but usable Linux system fairly quickly.

While Softcraft claims compliance with the Linux File System Standard (the “FSSTND”), the default locations that come with many free software packages have been accepted, and so they expressly violate the FSSTND by installing several packages in the /usr/local hierarchy. They don't include documentation (at least in the manual) on how and where they deviate from the FSSTND, either. However, most of the base system does appear to follow the FSSTND.

A Softcraft system with extra packages (more than the base system) installed feels something like a SunOS system with an active system administrator. Plenty of files are in /usr/local (including Emacs), and other files are tucked away in other various corners; for example, Postgres95 is in the /usr/local/postgres95/ directory. Softcraft responds that they do this so that users familiar with commercial versions of Unix will feel comfortable.

The X configuration was very simple—there were only two questions to answer (type of mouse and type of video board). However, this provides a 640x480 pixel X configuration; to get any better resolution, the user is instructed to edit the /etc/XF86Config file by hand.

Craftworks' home page is at www.craftwork.com.

Linux Universe

Linux Universe is a book with simple installation and configuration instructions and a small reference, which includes a CD with a Linux distribution on it. It's translated from German, and the distribution on it is also apparently translated from German, since some of the comments in the scripts are still in German.

While it is completely ELF-based, the version available at the time of testing used the version 4 ELF library, not the new version 5 ELF libraries used by all the other ELF-based distributions. When Linux Universe was produced, the version 5 libraries were still in alpha testing and not ready for release. However, we have been assured that the new version, which should be available in stores by the time you read this, will have the version 5 ELF libraries. If you wish to run Linux binaries other than those included with the system, make sure you get the newer version.

Also unlike any of the other Linux distributions here, it doesn't include LILO. Instead, it provides a full-screen boot manager somewhat like OS/2's boot manager, which is easy to configure and is able to read ext2fs file systems—so it is able to boot any kernel on an ext2fs file system, not just ones that have been specially configured, as with LILO. However, the full-screen loader is itself loaded by a very simple loader that can be confusing to the new user. It requires you to remember which partitions of which drives hold the operating systems you want to boot.

Linux Universe is intended to be a companion to *Linux—Unleashing the Workstation in your PC,* by the same authors, and you can purchase Linux Universe alone or in a kit with its companion volume. If you are not already familiar with Linux (or at least Unix), you will want to purchase the whole kit, not just Linux Universe.

Linux Universe is designed (like Yggdrasil) to be run with the system CD in the drive, so that even packages that are not installed on the hard drive can still be run. Linux Universe adds technology which causes files accessed on the CD to be automatically copied to the hard drive for future access. The same design is used to install and run it from an NFS server.

The graphical configuration utility is simple to use and seems to work well. It works quickly and intelligently. When filling out the networking configuration, for example, it guesses most of the information once you type in the IP address.

You can find Linux Universe on the Web at www.springer-ny.com/samples/linux/linux.html.

Red Hat Commercial Linux 2.1

The first commercial distribution to adopt an upgradable packaging scheme, Red Hat's 2.1 release includes a single-command upgrade facility. A single script was included to do the upgrade from Red Hat 2.0 or 2.0Beta to 2.1, and it worked very well. At the end of the upgrade, it notifies the user which configuration files have been changed and the names of the files where the originals have been saved.

Unfortunately, going from version 2.0 to 2.1, it replaced the /etc/passwd and /etc/group files (among others) with new ones that had new administrative users, but didn't include any users that had been added to the system since the original install of Red Hat 2.0. No tool was provided to merge the old and new password files, either. Fortunately, Red Hat considers this a bug and will fix it in future releases. It is also fortunate that the previous versions were preserved—Red Hat's RPM tool always makes a backup when changing configuration files—and the upgrade script warned about the change, making it a simple matter to move the original versions back into place.

With this small, easily-fixed exception, the Red Hat upgrade process ran smoothly and took only a few minutes to upgrade a rather fully configured system to the latest version of Red Hat with proper configuration files in place. Red Hat has plans to improve the upgrading process so that it is even smoother for more people; in nearly all cases in the future, a single command should be entirely sufficient.

In general, Red Hat provides a reasonably wide range of applications on multiple architectures. In December 1995, at DECUS/San Francisco, Red Hat announced Red Hat Linux for Alpha, which runs on several different Digital Alpha systems. Red Hat's packaging scheme is designed to support multiple architectures transparently, including building packages, and Red Hat has announced that they are considering other architectures as well.

Red Hat includes a graphical package management tool, as well as a command-line tool. However, their assumption is that people who don't want or need to edit configuration files directly will configure and use X, and their configuration tools are almost entirely X-based.

Red Hat provides your choice of graphical and text-based installations, and can install from CD-ROM, NFS, floppy, or FTP. The floppy and FTP installations only work in text-based mode at the moment. The installation asks as many

questions as necessary immediately and then installs **all** the applications, not requiring user interaction again until after all the packages have been installed.

Red Hat provides their own easy-to-use X configuration program called the Xconfigurator. It uses dialog boxes that ask fairly easy questions to write an XF86Config file to configure X.

The URL for Red Hat's home page is www.redhat.com.

Slackware 3.0

This veteran descendent of the original Linux distribution, SLS, is still easy to install, if you don't mind tending it while it asks questions. It has wide support for installation from different media—it even has experimental support for installing from tape drives—and it still has the best support for installing from floppies. It also has a very wide selection of available software. If you want to use TeX to typeset Klingon or Tengwar, it's built in; just make the right selections while installing. Slackware has a long history and has a rich assortment of packages. Most of the 1000-page tomes covering Linux which are available in any bookstore cover Slackware, so it is also well-documented.

However, Slackware has no real upgradability. To upgrade a package (which is simply a .tar.gz file), you can only remove the old package and install the new one. For a technically advanced user who remembers all the configuration files for the package, who knows exactly what files to back up before doing an upgrade process, and has time to do so, that works, except when the user makes mistakes.

Slackware uses the standard XFree86 xf86config program to configure X, which is not particularly user-friendly, though very thorough. Once you get X running, however, you can take advantage of Slackware's fairly wide variety of X applications.

If you don't mind re-installing regularly to upgrade and demand very fine-grained control over exactly which files get installed, Slackware provides some of the most precise controls over what gets installed and what does not. For instance, if hard drive space is limited, and you want to install **only** the TeX fonts you want, most of them are selected separately. Thai fonts, OCR fonts, and Gothic German fonts are perhaps not likely to be used by the same people; Slackware allows you to select each of those families (and many more) separately. If you want as many packages as possible provided with the distribution, you will appreciate Slackware's wide range of available software. We counted 326 packages in Slackware 3.0.

Slackware's Web site is at www.cdrom.com/titles/slackware.html.

Yggdrasil Plug and Play Linux, Fall '95

The first Linux distribution designed for CD, Yggdrasil has had continued popularity due in part to graphical configuration, bootable floppies included with the distribution, and multimedia support.

Yggdrasil has had a history of including kernels with patches they have tested, but which aren't included in the standard Linux kernel. Fall '95 continues this tradition.

After installing the “Suggested” configuration (at 250 MB, the only choice smaller than “Everything”, which takes 600MB), it is impossible to compile the kernel without either mounting the CD or installing some packages that aren't installed as a part of the “Suggested” configuration. The manual doesn't say what packages need to be installed to build the kernel with the CD removed, and since the control panel doesn't show which packages are already installed, it's not clear what packages need to be installed. Even after installing all the needed software components, we still had to figure out that we had to remove /usr/src/linux/include/linux/version.h in order to successfully build the kernel; we were not able to find that in the documentation. This would not have been an important issue except for the fact that the standard kernel didn't support the 3C509 Ethernet card, and we needed to build a kernel with 3C509 support in order to test the networking setup. Further testing showed that several functions of the basic control panel, such as printing, also did not function with the “Suggested” configuration unless the CD-ROM was mounted.

Although running Yggdrasil without the system CD mounted can take some work, Yggdrasil is one of the few distributions able to run entirely from the CD. This capability has been included with Yggdrasil for some time and is a great way to demonstrate Linux to skeptical friends. Not surprisingly, running multimedia applications under X entirely from the CD-ROM takes considerable memory; it doesn't work well with less than 16 MB of memory.

Yggdrasil doesn't set up networking as part of the installation, but it does provide a graphical tool for setting up basic networking once you get X set up. It also has a reasonably easy X setup program that gets invoked automatically when you start X, if you haven't already configured X. It also tells you how to re-run the configuration if you aren't satisfied with the first configuration you produce.

Yggdrasil doesn't come with some programs that are now considered standard with Linux, such as **rxvt**. On the other hand, it does seem to include a fairly full range of multimedia tools. The PC Speaker sound patches, which provide sound capabilities through the standard PC speaker, are already applied to the supplied kernel source tree and are one of the configuration options. Yggdrasil

also comes with XEmacs, a version of Emacs with a good graphical user interface.

You will find Yggdrasil on the Web at www.yggdrasil.com.

Reading the Feature Chart

First, a word of warning: while the feature chart is intended to help you compare the features offered by different distributions, it doesn't say anything about the quality of the features. The chart doesn't tell the whole story, nor does this article, because it is impossible for a chart or article to do so. The chart *can* give you an idea of what the vendors are trying to provide with their distributions.

A “no” isn't necessarily a strike against a distribution. Don't sit down and count up the number of boxes with “yes” in them and consider that a reasonable ranking of the distributions. Instead, consider the features that **you** particularly want and need. If you have further questions, contact the vendors and ask. Buy from vendors with a money-back guarantee if you are worried that you will get stuck with a distribution you don't want.

The labels on the feature chart (page FIXME) aren't entirely self-explanatory, as you may have already noticed. An explanation of the entire chart is in order here. If there is any part that you don't understand or are not interested in, feel free to skip it. Starting from the top...

Like everything else, the **vendor's list price** isn't necessarily the price you will pay. Nearly all of these distributions are re-sold. Also, like Linux itself, Debian has no official vendor and no official vendor price. The Debian distribution is made available for ftp from ftp.debian.org, and several vendors provide it, either as the sole distribution on a CD or as part of a collection.

Slackware, Debian, and Red Hat are all contained on archive CDs containing snapshots of ftp sites. They don't come with paper manuals when you buy them that way, but they are often less expensive. Technical support is not included with most archive CDs, so if you expect to need a helping hand with your installation, this may not be the easiest way for you to go.

Packaging is technology for easily and correctly installing, removing, and upgrading parts of the system. Each part would include several inter-related files. For example, all the standard include files would probably be included in the same part. The first line of the Packaging section indicates whether you can at least add and remove packages with the provided package management tools.

Package upgradability involves preserving (as much as possible) the configuration of the package correctly on the hard drive while installing a new version. This involves at the very least some method of distinguishing configuration files from other files, so that when you upgrade, a package's configuration files are not changed, and you do not have to re-configure the package. If the package management tools are unable to preserve the configuration untouched (for example, if the format of a configuration file changes with a new version), the package management tools need to notify the administrator of that fact.

Dependencies are a capability which allows one package to require that another package also be installed. For instance, LaTeX requires that TeX also be installed, so if LaTeX and TeX are provided in separate packages, the package containing LaTeX may require that the TeX package be installed first, since the functionality of LaTeX depends on it.

Dependencies are usually absolute; a package either requires or does not require another package in order to function. Debian allows fine shades of distinction: A package can absolutely depend on another package being installed, it can recommend that any sane user would want the additional package installed, or it can suggest that you will probably want to install the other package.

Format refers to the way that packages are maintained on the installation media. Debian and Red Hat each use special archives of their own design which contain the extra information needed to implement their extra features. Slackware installs standard ".tar.gz" or ".tgz" archives created by using tar to archive a set of files and gzip to compress the archive; those archives do not, therefore, have the extra information needed to remove or upgrade archives. Yggdrasil and Linux Universe install by copying files directly from a file system on the CD-ROM to the hard drive, so they can only use their package management tools to install from the installation CD.

Third-party packages are closely related to format; those distributions which provide single-file packages are capable of installing packages built by third parties, downloaded from the Internet or found on other CDs. As an example of what can be done with third-party packages, Caldera built the Caldera Network Desktop upon Red Hat Commercial Linux (RHCL) and added new packages in Red Hat's RPM format to create a new distribution that is essentially a superset of RHCL.

Source packages means the source code is also installable with the package management tools

Buildable Source means the package has the capability to build entire packages with one command (which may or may not be integrated with the package management tools).

Multi-architecture means support for more than one binary architecture. The Intel 80386 and above (denoted here as i86) is certainly the most common CPU architecture supported, but Linux also runs on other platforms, including some Amigas and Ataris, DEC's Alpha, the Acorn Archimedes, and Sparc. Some of the distributions are beginning to include support for more than one architecture. We expect that by the time you read this chart, support for more architectures will be, at least, announced by more distributions.

Graphical and text-based package management refer to the tools available for managing packages. **Graphical** refers to native X-based tools, and **text-based** refers to those that run in character more (even in an Xterm or rxvt session).

Series sub-selection refers to logically grouping packages together into intelligent groups, but also being able to select individual components. For instance, making a group or series which contains everything related to the TeX typesetting system, but being able to separately choose each individual part separately.

Over the past year, the Linux community has been slowly migrating from the a.out to the ELF **binary file format**. ELF has many advantages, but it has been necessary to move slowly to avoid causing more pain than necessary. However, ELF is now the standard, and we have purposely not included any distributions that do not at least support ELF binaries.

ELF-based distributions can choose to include a.out libraries, in order to provide support for legacy applications.

Most distributions (all in this lineup) provide iBCS2 support for binaries from many i86 Unix platforms.

Supported architectures are the types of machines supported by the distribution.

The standard PC architecture doesn't provide a standard way to boot from CD-ROMs, so in order to install Linux, it is not possible to simply boot a Linux CD-ROM. [No letters to the editor telling us that the Adaptec 2940 provides support for booting from a CD-ROM; that's not “support by the standard PC architecture”—ED] This means that the distribution needs to provide some other way of booting. There are essentially two ways of doing this. One is to provide a bootable floppy disk containing the Linux kernel, and the other is to

rely on already having DOS installed. Several distributions provide both options.

In order to boot from a floppy, either:

- a bootable floppy needs to be included with the distribution, or
- a way needs to be provided to make a floppy from files on the CD-ROM.

Unfortunately, it's hard to make a single floppy that is sufficient for booting on all platforms, but failing to supply a floppy with a kernel makes the user rely on having another operating system (DOS, Linux, or any version of Unix will do) to use to create the floppy. Fortunately, this is rarely a problem, but if you have no other operating system available (even on a friend's computer...) to use to write the necessary floppies, you will probably want to seriously consider purchasing a distribution which provides the necessary floppies as part of the package.

It is possible to boot Linux directly from DOS, and some distributions provide an option to use no boot floppies at all by first booting DOS and then booting Linux from the CD-ROM. In this collection, Linux Universe provides this choice, as well as the option of booting from a floppy.

Some distributions provide a wide variety of boot images, whereas others use one or only a few. If you are able to make floppies yourself, those with more boot floppy choices *may* prove easier to configure for your hardware. The **Boot Images** line doesn't count duplicates for different size drives; it only counts the choices for one size of floppy drive.

All the distributions provide at least the option of booting from floppy, and all of them provide a way to make your own boot floppies from the CD-ROM under DOS and/or Linux. Some also provide some assistance (under DOS, Linux, or both) to make choosing and/or creating the floppy or floppy set easier. For instance, Red Hat provides a perl script which asks questions about the configuration you are interested in and then creates disks appropriate for your hardware configuration.

The **number of floppies required** to install the system varies widely. Debian installs its entire “base system” from a set of five floppies and then requires one blank floppy after the install to create an appropriate boot disk. If you boot the Linux Universe CD-ROM from DOS, it can install without any floppies at all.

While most of this article has assumed that you are installing from CD-ROM (after all, that's what most of the vendors provide), that's not necessarily true. You may wish to copy packages to a subdirectory of a local hard drive and

install from there, or you may wish to install from floppy or via NFS from a local server or via anonymous FTP or from a tape.

In order to install on most notebooks, support for PCMCIA (also known as PC Card) is required, unless you are want to install one of the distributions that are available on floppies and are interested in shuffling floppies for hours. If you have a PCMCIA Ethernet or SCSI adaptor supported by the Linux PCMCIA card services package, some distributions make it possible for you to do a network installation over Ethernet, or a install from a SCSI CD-ROM connected to a SCSI adaptor. This can be a real time-saver.

If you don't want to partition your disk, some distributions provide the option of installing in a **LINUX** subdirectory of a DOS file system using a Linux file system called UMSDOS. While this imposes on Linux all the inefficiencies of the DOS file system, it is an easy way to get started using Linux without committing a partition to do so. Yggdrasil goes one step farther, allowing you to install the base Linux system without booting Linux, by using a DOS program to do the installation.

Most distributions now have customized documentation written about them, which is either included with the distribution by default or is available as an option. Some distributions have been covered in other books available in the bookstore; we have mentioned their presence under **Alternate Books**. Some distributions are covered by freely-available documentation; the **Installation HOWTO** and *Linux Installation and Getting Started* currently cover Slackware primarily, and Debian's manual is distributed freely via the Internet, including an HTML version on www.debian.org.

Some distributions provide the capability to run nearly or completely from CD-ROM, which can be an effective way to demonstrate or experiment with Linux.

One of the most difficult parts of configuring a Linux system is configuring the X Window System. The XFree86 project, Inc. provides a configuration program called xf86config, which has a fairly basic user interface and requires you to know a lot about your configuration, but is quite complete. Some distributions provide tools to make X configuration easier.

Some vendors provide a support program, and some provide several different support programs. This chart only mentions whether at least one program is available—check with the vendors if you have specific requirements that you need met.

While all of these distributions are available via CD-ROM, some are also sold in other formats. There is still one third-party vendor (Linux System Labs) which

provides Slackware on floppy, and several of the vendors provide their distribution over the Internet via anonymous FTP.

Some of the distributions include configuration tools, mostly graphical ones. If you are not familiar with administering a Linux or Unix system, you are likely to find most common tasks simplified by a user interface that handles common tasks easily. None of the configuration tools provided with any of the distributions prevents you from doing your configuration by hand, so there is no reason to complain if you prefer to configure your system by hand.

In particular, we looked for configuration tools for setting up and maintaining a network, adding and deleting users and groups, mounting and un-mounting file systems, setting up print services, and manipulating boot scripts (generally kept in /etc/init.d). Again, we did not try to rank them in usefulness, and some of the scripts were definitely more functional for our purposes. On the other hand, a script with more functionality than one user needs could include needless, confusing detail for another user. Configuration scripts should have a review of their own some day, but mentioning whether or not they exist is a first step.

Last, but not least, some distributions require more memory to install than others. Some distributions install with only 4MB of memory, but others require 8MB. As new versions of the distributions are released, their memory requirements may change. If you have less than 8MB of memory, it will probably be worth your while to find out how much memory is currently required by each of the distributions you are considering.

Summary

Linux distributions have changed a lot in the few years they have been around. The Linux File system Standard has improved their ability to inter-operate; binaries compiled for one distribution are now more likely to run on another distribution, because standard files are more likely to be in the same place from one distribution to another. As distributions have competed, bugs have been worked out of all of them; thus users have benefited from distributions they haven't even used.

Since no single distribution provides an optimal environment for everyone, each distribution contributes something to the pot. Furthermore, having distributions targeted for different kinds of users means that no distribution is pressured into complete mediocrity by the attempt to be all things to all people. It's a well-worn clich that to do all things equally well is to do nothing at all well...

Archive Index Issue Table of Contents

Advanced search

Linux Distributions Compared

Linux Journal Staff

Issue #23, March 1996

Been considering installing or re-installing Linux on a PC? Confused by the wide range of distributions available? This article may not solve all your woes, but it should at least put you on the right track.

Why doesn't *Linux Journal* recommend one particular Linux distribution as the best? Why not do quarterly awards for the best current Linux distribution? There are at least two good reasons and one not-so-good reason. The good reasons are that different users have different needs, and that a “best” rating for any one distribution would unfairly penalize the other distributors. The not-so-good reason has been that *Linux Journal* hasn't had the resources to do comparative review.

While we still will not recommend one “best” distribution, we have recently acquired hardware specifically for testing distributions. While we can't begin to buy a full set of hardware that will allow us to test distributions on a wide range of hardware, we can assume that since all the distributions use the Linux kernel with few or no modifications, there are not likely to be too many differences based on hardware. By buying mainstream hardware (see The Test Platform sidebar), we can ignore the hardware problems that do not generally differentiate distributions and concentrate on the other characteristics of the distributions.

You Be the Judge

We are not even going to attempt to rank the distributions. We will introduce the distributions that are the most common in the U.S. as this is written; we will introduce other distributions in future articles. As we explore more distributions, we will update our table for feature comparison, and future articles will include the entire table for all the distributions tested so far for easy comparison.

Our comparisons will be designed to allow you to rank the distributions based on your own needs, rather than our perceptions of your needs. To do this, we will provide a description of each distribution, as well as comparison charts of features so that you can easily see the trade-offs that various Linux distributors have made.

Is this a cop-out? Are we shirking our duty? It has become clear to us that it would be hard for *Linux Journal* to rank the distributions; our staff members have different personal favorite distributions and defend their choices rationally—occasionally even argumentatively. At *Linux Journal*, we use Linux extensively—nearly exclusively—and we still don't agree which distribution is the best. We have different priorities, skills, and expectations, and we believe this is true of our readers as well.

We have reason to think that our readership is thoughtful and intelligent, and we have certainly been informed in *many* letters to the editor that our readers appreciate being given a chance to form their own opinions. So instead of attempting to make up your mind for you, we would like to give you as much material as possible to use to make up your own mind.

Version numbers

Many new Linux users confuse the version of the distribution they are using with the version number of the kernel they are running. As described in the What's a distribution? sidebar, the Linux kernel is just one of the many pieces of software needed to create an entire distribution. Each distribution uses version numbers of its own to keep track of the state of the entire distribution, which has more to do with the collection of programs than with the particular kernel involved. Indeed, many distributions have included two or more different kernel versions in one version of the distribution.

However, it's worth understanding the version numbers used for the Linux kernel itself, since the kernel is a key part of any Linux distribution. Kernel version numbers come in three parts: the major version number, the minor version number, and the patch-level. The Linux kernel is being constantly developed by a large team of developers, and while they add new features, they occasionally introduce new bugs. To keep this from causing a problem for Linux users, the developers periodically dedicate several months to fixing bugs and creating a very robust, stable kernel. When this is done, a stable version is released with an even-numbered minor version number. The developers then begin adding features (and temporarily breaking things sometimes) in development versions with odd-numbered minor version numbers.

Unless you want to live on the “bleeding edge” of Linux development, you will probably want to stick with the latest stable kernel version. As of this writing, the latest stable kernel version is 1.2.13; by the time you read this, preparations will probably be underway for 1.4.0.

Value Added

Several vendors are adding value to existing distributions in various ways. For example, Caldera is adding commercial components to Red Hat Commercial Linux to create their Caldera Network Desktop. WorkGroup Solutions used to enhance Slackware as the base for WGS Linux; they have now switched to basing their work on Red Hat Commercial Linux. Trans-Ameritech sells disks with several distributions, including Slackware and Debian; they base their own value-added work on Slackware, trying to make it easier to install.

Value-add distributions are worth serious consideration, but they aren't the subject of this review. As part of deciding which distribution to get, you may want to consider what you can get from value-add vendors as well as from the base distributions in question. Most value-add vendors (as well as distribution vendors) advertise in *Linux Journal*.

Binary File Formats

One of the most common sources of confusion in the Linux world today involves binary file formats (See What is a binary file format? sidebar). The Linux community is in a state of transition from the old “a.out” binary file format to the new “ELF” binary file format, which has many features that are completely missing in the a.out format.

ELF is the binary file format used by Unix System V Release 4, but that doesn't mean that a Linux binary in the ELF file format is compatible with SVR4, nor does it mean that SVR4 binaries can run on Linux. The capability for Linux to run some SVR3 and SVR4 binaries is provided by the iBCS2 compatibility package, which most distributions include.

One of ELF's features is extensibility; with ELF, it is possible for developers to add features that weren't thought of when the format was first designed. For instance, one Linux developer has noted that he could add icons to ELF executables without breaking any software. Icons weren't considered when ELF was developed, but the format is extensible enough that they can be easily added.

But perhaps you don't care if you can add an icon to your ELF binary, or even if anyone else can. What does ELF do for you? Fundamentally, it makes life much easier for Linux developers. It also has a few esoteric features which make it

practical to support some software under Linux that was previously impractical to support. So it gives you more and better software available for Linux.

Since the whole Linux community is moving to ELF, you don't want to get stuck with a distribution which does not support ELF. A year from now, it will be almost impossible to find a.out binaries for programs you want, and important bug fixes may only be available for ELF.

Because of this, we have only reviewed distributions which at least support ELF binaries. The only distributions that are reviewed here which are not **based** on the most current ELF libraries are currently being updated, and should be based on standard ELF libraries by the time you read this.

What do we mean by **based**?

ELF-based means the entire distribution, or at least, almost the entire distribution, consists of ELF binaries. a.out binaries are not provided with the system, or if they are provided, they aren't part of the “core” of the system or are not available in ELF format.

By contrast, “supports ELF” means that, while the distribution is partially or completely built of binaries in the a.out format, the ELF **programming libraries** are included so that ELF binaries will also run.

Media

Many distributions are available solely on CD-ROM. There are several reasons for this:

- So much software is available for Linux that it is impractical to provide it all on floppies.
- It is much easier to install software from CD-ROM than to change floppies once a minute.
- A cheap supported CD-ROM drive costs approximately the same as the stack of floppies needed to install a complete Linux distribution.

However, some distributions (including Debian, Red Hat, and Slackware) are available via FTP over the Internet in a form designed to fit on floppies, and Linux System Labs still offers a floppy distribution service, from which you can order an entire Linux distribution on floppies, though it is the only company left, to our knowledge, which does this.

Copyright Considerations

Most distributions are freely available over the Internet, although only some are actively distributed in a way that makes it feasible to install them directly from the Internet. You should be aware that while distributors can restrict you from running commercial software components on more than one machine, they cannot restrict you from installing the base Linux software on multiple machines. A Linux distribution which is packaged in a form which cannot be installed without installing proprietary software that is under copyright licensing terms more strict than the GNU General Public License is probably in violation of copyright law. [And should be completely avoided, in my strongly held opinion—ED]

To sum up, you should feel free to install any Linux distribution you buy on as many machines as you like, as long as you respect the licensing terms of any proprietary software that is included with the distribution. Conversely, the vendor needs to make it possible for you to do so without taking unreasonable steps.

If you encounter a vendor who makes it difficult or impossible to install free components without installing proprietary components that have restrictive licenses, first politely bring the issue to their attention—they may not have considered the issue. If the vendor refuses to resolve the issue, return the distribution, ask for your money back, and send a letter to the Editor of *Linux Journal*.

The Future

By the time you read this, some of the distributions will have made new releases, and so some of this description will be out of date. We hope all the bugs will have been fixed, for instance...

We would like to keep tracking distributions as they improve and, occasionally, publish descriptions and the feature comparison table in updated form.

Missing Bugs

One thing you may notice missing from this overview is a list of bugs in each distribution. In general, we assume that some noticeable bugs are inevitable in software that is evolving as quickly as Linux and that vendors will be responsive in tracking and eradicating bugs. If they aren't, word of mouth between users will be more effective than any sort of complaining on our part. We also want to avoid showing any bias, and if we start listing bugs we find, we will either show bias or appear to show bias simply by the “choice” of bugs we publish.

Since we can't do a fair comparison of the number and severity of bugs between distributions, we have tried to note only bugs that involved completely missing functionality or which were difficult to work around and hard for the user to avoid.

If working around simple bugs is a problem for you, we urge you to purchase a distribution from a vendor who offers technical support for the installation process. By providing installation technical support, these vendors not only help you with the problems, but force on themselves a vested interest in fixing those bugs. Providing support is expensive, and solving the same problem over and over again is a waste of their time and money.

At the same time, if you buy installation technical support, please understand that you aren't buying a lifetime warranty for every part of the system. None of the vendors here provide that as part of their base product. You can buy very comprehensive support packages from several vendors if you need them. Happy shopping!

Debian .93R6

The only distribution in this lineup which is not currently developed on a commercial basis (Slackware was originally also non-commercial), Debian has been a long time in the making. Unlike all the other Linux distributions, Debian is put together by a large team of volunteers all over the world. While a few people are in charge of a very small core of the distribution, almost all the decisions are made by consensus (which has given Debian a reputation for vaporware, since consensus can take some time to achieve), and nearly all of the packages are developed independently by members of a large development team.

By following a strict set of rules and using the most powerful packaging technology currently available for Linux, this large team has achieved a remarkably coherent Linux package. As this article is being written, Debian is still the **only** distribution which has complete *dependencies*: when you install one package, it checks to see if other needed packages are also installed. It also checks version numbers if requested; package **A** can insist that package **B** be installed, and that package **B** be version *x* or greater. This makes upgrading any package nearly foolproof.

These dependencies are a natural requirement of Debian's distributed development and are well-tested, because many of Debian's users and developers upgrade parts of their systems on an as-necessary basis, which exercises the dependency-checking features considerably.

The Debian installation procedure does not have as many of the user-friendly assumptions as many other distributions. It won't guess at which disks you wish to use or which partitions on your disks you wish to use for swap and root, for instance. It doesn't give you the option to skip the check for bad blocks while making a file system. It installs the entire “base” system from floppy (three floppies), and then you reboot into the installed system with enough programs installed to install any other packages you like. This way, all the base system has to support is floppies and hard drives, and then you can choose which kernel modules to load during the initial “base” installation; those determine what hardware is supported. Debian is highly modularized.

Debian is one of the strictest distributions about setting passwords. As soon as you re-boot the “base” system, you are prompted for the root password, with an explanation of how to choose a good password. Also, part of adding a new user under Debian is setting the user's password.

All of Debian's packaging tools run in text mode, so X is not required to use any of Debian's sophisticated upgrade capabilities. They can be used in an xterm or rxvt on the console, over a modem link or over the network. The package tools are also very fast.

The default Debian kernel is highly modularized. This means that you rarely need to recompile the kernel, and you don't have lots of unnecessary, unused drivers in the kernel taking up memory and causing possible conflicts. Where Red Hat has 72 possible boot disks in its standard set, Debian has 1—but Red Hat requires you to make only 3 disks to install, whereas Debian requires you to make 5 disks, and have an extra blank disk available, which is used to make another boot disk during the installation process (you can re-use the boot or root disk if you are brave, but we don't recommend it).

Debian is currently being transformed from an a.out-based distribution with optional ELF support to a fully ELF-based distribution. By the time you read this, a new release of Debian fully based on ELF should have been released; most of the Debian **packages** have already been made available in ELF format. Debian 1.1 will be fully ELF-based. (Due to a mis-communication, a version called Debian 1.0 was released on the November Infomagic CD. That version was incomplete and will not work if you attempt to install it. The real release has been renumbered 1.1 in order to avoid confusion between the official ELF-based release and the one accidentally included on the Infomagic CD.)

Debian's home page can be found on the Web at www.debian.org.

Craftworks Linux 2.0

This relative newcomer SoftCraft Linux, by SolutionS R Us) has relatively few packages, each of which contains many files. Craftworks Linux comes with a boot floppy, a CD-ROM, and a manual.

We ran up against one bit of confusion right away: as soon as we booted from the provided floppy, an unclear copyright message was presented that seemed to appropriate title to Linux and seemed to attempt to legally restrict the user from installing the product on more than one machine.

Following our own advice, we contacted Craftworks and asked their intention. They confirmed that their intention was only to protect their own proprietary software included with the distribution, and that they allow users to install the product on as many systems as they wish. They do claim compilation copyright on the CD-ROM and the floppy, which means that they restrict the user from making any verbatim copies of either the CD-ROM or the floppy except for backup purposes. However, they in no way intend to restrict re-distribution of any included free software, and they promised to resolve the issue by making their copyright licensing terms and notices much clearer in future versions of the product.

The installation was fairly simple; Craftworks provides three pre-selected sets of packages to install (other packages can be installed separately after the base installation has been completed). This makes it quite simple to install a simple but usable Linux system fairly quickly.

While Softcraft claims compliance with the Linux File System Standard (the “FSSTND”), the default locations that come with many free software packages have been accepted, and so they expressly violate the FSSTND by installing several packages in the /usr/local hierarchy. They don't include documentation (at least in the manual) on how and where they deviate from the FSSTND, either. However, most of the base system does appear to follow the FSSTND.

A Softcraft system with extra packages (more than the base system) installed feels something like a SunOS system with an active system administrator. Plenty of files are in /usr/local (including Emacs), and other files are tucked away in other various corners; for example, Postgres95 is in the /usr/local/postgres95/ directory. Softcraft responds that they do this so that users familiar with commercial versions of Unix will feel comfortable.

The X configuration was very simple—there were only two questions to answer (type of mouse and type of video board). However, this provides a 640x480 pixel X configuration; to get any better resolution, the user is instructed to edit the /etc/XF86Config file by hand.

Craftworks' home page is at www.craftwork.com.

Linux Universe

Linux Universe is a book with simple installation and configuration instructions and a small reference, which includes a CD with a Linux distribution on it. It's translated from German, and the distribution on it is also apparently translated from German, since some of the comments in the scripts are still in German.

While it is completely ELF-based, the version available at the time of testing used the version 4 ELF library, not the new version 5 ELF libraries used by all the other ELF-based distributions. When Linux Universe was produced, the version 5 libraries were still in alpha testing and not ready for release. However, we have been assured that the new version, which should be available in stores by the time you read this, will have the version 5 ELF libraries. If you wish to run Linux binaries other than those included with the system, make sure you get the newer version.

Also unlike any of the other Linux distributions here, it doesn't include LILO. Instead, it provides a full-screen boot manager somewhat like OS/2's boot manager, which is easy to configure and is able to read ext2fs file systems—so it is able to boot any kernel on an ext2fs file system, not just ones that have been specially configured, as with LILO. However, the full-screen loader is itself loaded by a very simple loader that can be confusing to the new user. It requires you to remember which partitions of which drives hold the operating systems you want to boot.

Linux Universe is intended to be a companion to *Linux—Unleashing the Workstation in your PC,* by the same authors, and you can purchase Linux Universe alone or in a kit with its companion volume. If you are not already familiar with Linux (or at least Unix), you will want to purchase the whole kit, not just Linux Universe.

Linux Universe is designed (like Yggdrasil) to be run with the system CD in the drive, so that even packages that are not installed on the hard drive can still be run. Linux Universe adds technology which causes files accessed on the CD to be automatically copied to the hard drive for future access. The same design is used to install and run it from an NFS server.

The graphical configuration utility is simple to use and seems to work well. It works quickly and intelligently. When filling out the networking configuration, for example, it guesses most of the information once you type in the IP address.

You can find Linux Universe on the Web at www.springer-ny.com/samples/linux/linux.html.

Red Hat Commercial Linux 2.1

The first commercial distribution to adopt an upgradable packaging scheme, Red Hat's 2.1 release includes a single-command upgrade facility. A single script was included to do the upgrade from Red Hat 2.0 or 2.0Beta to 2.1, and it worked very well. At the end of the upgrade, it notifies the user which configuration files have been changed and the names of the files where the originals have been saved.

Unfortunately, going from version 2.0 to 2.1, it replaced the /etc/passwd and /etc/group files (among others) with new ones that had new administrative users, but didn't include any users that had been added to the system since the original install of Red Hat 2.0. No tool was provided to merge the old and new password files, either. Fortunately, Red Hat considers this a bug and will fix it in future releases. It is also fortunate that the previous versions were preserved—Red Hat's RPM tool always makes a backup when changing configuration files—and the upgrade script warned about the change, making it a simple matter to move the original versions back into place.

With this small, easily-fixed exception, the Red Hat upgrade process ran smoothly and took only a few minutes to upgrade a rather fully configured system to the latest version of Red Hat with proper configuration files in place. Red Hat has plans to improve the upgrading process so that it is even smoother for more people; in nearly all cases in the future, a single command should be entirely sufficient.

In general, Red Hat provides a reasonably wide range of applications on multiple architectures. In December 1995, at DECUS/San Francisco, Red Hat announced Red Hat Linux for Alpha, which runs on several different Digital Alpha systems. Red Hat's packaging scheme is designed to support multiple architectures transparently, including building packages, and Red Hat has announced that they are considering other architectures as well.

Red Hat includes a graphical package management tool, as well as a command-line tool. However, their assumption is that people who don't want or need to edit configuration files directly will configure and use X, and their configuration tools are almost entirely X-based.

Red Hat provides your choice of graphical and text-based installations, and can install from CD-ROM, NFS, floppy, or FTP. The floppy and FTP installations only work in text-based mode at the moment. The installation asks as many

questions as necessary immediately and then installs **all** the applications, not requiring user interaction again until after all the packages have been installed.

Red Hat provides their own easy-to-use X configuration program called the Xconfigurator. It uses dialog boxes that ask fairly easy questions to write an XF86Config file to configure X.

The URL for Red Hat's home page is www.redhat.com.

Slackware 3.0

This veteran descendent of the original Linux distribution, SLS, is still easy to install, if you don't mind tending it while it asks questions. It has wide support for installation from different media—it even has experimental support for installing from tape drives—and it still has the best support for installing from floppies. It also has a very wide selection of available software. If you want to use TeX to typeset Klingon or Tengwar, it's built in; just make the right selections while installing. Slackware has a long history and has a rich assortment of packages. Most of the 1000-page tomes covering Linux which are available in any bookstore cover Slackware, so it is also well-documented.

However, Slackware has no real upgradability. To upgrade a package (which is simply a .tar.gz file), you can only remove the old package and install the new one. For a technically advanced user who remembers all the configuration files for the package, who knows exactly what files to back up before doing an upgrade process, and has time to do so, that works, except when the user makes mistakes.

Slackware uses the standard XFree86 xf86config program to configure X, which is not particularly user-friendly, though very thorough. Once you get X running, however, you can take advantage of Slackware's fairly wide variety of X applications.

If you don't mind re-installing regularly to upgrade and demand very fine-grained control over exactly which files get installed, Slackware provides some of the most precise controls over what gets installed and what does not. For instance, if hard drive space is limited, and you want to install **only** the TeX fonts you want, most of them are selected separately. Thai fonts, OCR fonts, and Gothic German fonts are perhaps not likely to be used by the same people; Slackware allows you to select each of those families (and many more) separately. If you want as many packages as possible provided with the distribution, you will appreciate Slackware's wide range of available software. We counted 326 packages in Slackware 3.0.

Slackware's Web site is at www.cdrom.com/titles/slackware.html.

Yggdrasil Plug and Play Linux, Fall '95

The first Linux distribution designed for CD, Yggdrasil has had continued popularity due in part to graphical configuration, bootable floppies included with the distribution, and multimedia support.

Yggdrasil has had a history of including kernels with patches they have tested, but which aren't included in the standard Linux kernel. Fall '95 continues this tradition.

After installing the “Suggested” configuration (at 250 MB, the only choice smaller than “Everything”, which takes 600MB), it is impossible to compile the kernel without either mounting the CD or installing some packages that aren't installed as a part of the “Suggested” configuration. The manual doesn't say what packages need to be installed to build the kernel with the CD removed, and since the control panel doesn't show which packages are already installed, it's not clear what packages need to be installed. Even after installing all the needed software components, we still had to figure out that we had to remove /usr/src/linux/include/linux/version.h in order to successfully build the kernel; we were not able to find that in the documentation. This would not have been an important issue except for the fact that the standard kernel didn't support the 3C509 Ethernet card, and we needed to build a kernel with 3C509 support in order to test the networking setup. Further testing showed that several functions of the basic control panel, such as printing, also did not function with the “Suggested” configuration unless the CD-ROM was mounted.

Although running Yggdrasil without the system CD mounted can take some work, Yggdrasil is one of the few distributions able to run entirely from the CD. This capability has been included with Yggdrasil for some time and is a great way to demonstrate Linux to skeptical friends. Not surprisingly, running multimedia applications under X entirely from the CD-ROM takes considerable memory; it doesn't work well with less than 16 MB of memory.

Yggdrasil doesn't set up networking as part of the installation, but it does provide a graphical tool for setting up basic networking once you get X set up. It also has a reasonably easy X setup program that gets invoked automatically when you start X, if you haven't already configured X. It also tells you how to re-run the configuration if you aren't satisfied with the first configuration you produce.

Yggdrasil doesn't come with some programs that are now considered standard with Linux, such as **rxvt**. On the other hand, it does seem to include a fairly full range of multimedia tools. The PC Speaker sound patches, which provide sound capabilities through the standard PC speaker, are already applied to the supplied kernel source tree and are one of the configuration options. Yggdrasil

also comes with XEmacs, a version of Emacs with a good graphical user interface.

You will find Yggdrasil on the Web at www.yggdrasil.com.

Reading the Feature Chart

First, a word of warning: while the feature chart is intended to help you compare the features offered by different distributions, it doesn't say anything about the quality of the features. The chart doesn't tell the whole story, nor does this article, because it is impossible for a chart or article to do so. The chart *can* give you an idea of what the vendors are trying to provide with their distributions.

A “no” isn't necessarily a strike against a distribution. Don't sit down and count up the number of boxes with “yes” in them and consider that a reasonable ranking of the distributions. Instead, consider the features that **you** particularly want and need. If you have further questions, contact the vendors and ask. Buy from vendors with a money-back guarantee if you are worried that you will get stuck with a distribution you don't want.

The labels on the feature chart (page FIXME) aren't entirely self-explanatory, as you may have already noticed. An explanation of the entire chart is in order here. If there is any part that you don't understand or are not interested in, feel free to skip it. Starting from the top...

Like everything else, the **vendor's list price** isn't necessarily the price you will pay. Nearly all of these distributions are re-sold. Also, like Linux itself, Debian has no official vendor and no official vendor price. The Debian distribution is made available for ftp from ftp.debian.org, and several vendors provide it, either as the sole distribution on a CD or as part of a collection.

Slackware, Debian, and Red Hat are all contained on archive CDs containing snapshots of ftp sites. They don't come with paper manuals when you buy them that way, but they are often less expensive. Technical support is not included with most archive CDs, so if you expect to need a helping hand with your installation, this may not be the easiest way for you to go.

Packaging is technology for easily and correctly installing, removing, and upgrading parts of the system. Each part would include several inter-related files. For example, all the standard include files would probably be included in the same part. The first line of the Packaging section indicates whether you can at least add and remove packages with the provided package management tools.

Package upgradability involves preserving (as much as possible) the configuration of the package correctly on the hard drive while installing a new version. This involves at the very least some method of distinguishing configuration files from other files, so that when you upgrade, a package's configuration files are not changed, and you do not have to re-configure the package. If the package management tools are unable to preserve the configuration untouched (for example, if the format of a configuration file changes with a new version), the package management tools need to notify the administrator of that fact.

Dependencies are a capability which allows one package to require that another package also be installed. For instance, LaTeX requires that TeX also be installed, so if LaTeX and TeX are provided in separate packages, the package containing LaTeX may require that the TeX package be installed first, since the functionality of LaTeX depends on it.

Dependencies are usually absolute; a package either requires or does not require another package in order to function. Debian allows fine shades of distinction: A package can absolutely depend on another package being installed, it can recommend that any sane user would want the additional package installed, or it can suggest that you will probably want to install the other package.

Format refers to the way that packages are maintained on the installation media. Debian and Red Hat each use special archives of their own design which contain the extra information needed to implement their extra features. Slackware installs standard ".tar.gz" or ".tgz" archives created by using tar to archive a set of files and gzip to compress the archive; those archives do not, therefore, have the extra information needed to remove or upgrade archives. Yggdrasil and Linux Universe install by copying files directly from a file system on the CD-ROM to the hard drive, so they can only use their package management tools to install from the installation CD.

Third-party packages are closely related to format; those distributions which provide single-file packages are capable of installing packages built by third parties, downloaded from the Internet or found on other CDs. As an example of what can be done with third-party packages, Caldera built the Caldera Network Desktop upon Red Hat Commercial Linux (RHCL) and added new packages in Red Hat's RPM format to create a new distribution that is essentially a superset of RHCL.

Source packages means the source code is also installable with the package management tools

Buildable Source means the package has the capability to build entire packages with one command (which may or may not be integrated with the package management tools).

Multi-architecture means support for more than one binary architecture. The Intel 80386 and above (denoted here as i86) is certainly the most common CPU architecture supported, but Linux also runs on other platforms, including some Amigas and Ataris, DEC's Alpha, the Acorn Archimedes, and Sparc. Some of the distributions are beginning to include support for more than one architecture. We expect that by the time you read this chart, support for more architectures will be, at least, announced by more distributions.

Graphical and text-based package management refer to the tools available for managing packages. **Graphical** refers to native X-based tools, and **text-based** refers to those that run in character more (even in an Xterm or rxvt session).

Series sub-selection refers to logically grouping packages together into intelligent groups, but also being able to select individual components. For instance, making a group or series which contains everything related to the TeX typesetting system, but being able to separately choose each individual part separately.

Over the past year, the Linux community has been slowly migrating from the a.out to the ELF **binary file format**. ELF has many advantages, but it has been necessary to move slowly to avoid causing more pain than necessary. However, ELF is now the standard, and we have purposely not included any distributions that do not at least support ELF binaries.

ELF-based distributions can choose to include a.out libraries, in order to provide support for legacy applications.

Most distributions (all in this lineup) provide iBCS2 support for binaries from many i86 Unix platforms.

Supported architectures are the types of machines supported by the distribution.

The standard PC architecture doesn't provide a standard way to boot from CD-ROMs, so in order to install Linux, it is not possible to simply boot a Linux CD-ROM. [No letters to the editor telling us that the Adaptec 2940 provides support for booting from a CD-ROM; that's not “support by the standard PC architecture”—ED] This means that the distribution needs to provide some other way of booting. There are essentially two ways of doing this. One is to provide a bootable floppy disk containing the Linux kernel, and the other is to

rely on already having DOS installed. Several distributions provide both options.

In order to boot from a floppy, either:

- a bootable floppy needs to be included with the distribution, or
- a way needs to be provided to make a floppy from files on the CD-ROM.

Unfortunately, it's hard to make a single floppy that is sufficient for booting on all platforms, but failing to supply a floppy with a kernel makes the user rely on having another operating system (DOS, Linux, or any version of Unix will do) to use to create the floppy. Fortunately, this is rarely a problem, but if you have no other operating system available (even on a friend's computer...) to use to write the necessary floppies, you will probably want to seriously consider purchasing a distribution which provides the necessary floppies as part of the package.

It is possible to boot Linux directly from DOS, and some distributions provide an option to use no boot floppies at all by first booting DOS and then booting Linux from the CD-ROM. In this collection, Linux Universe provides this choice, as well as the option of booting from a floppy.

Some distributions provide a wide variety of boot images, whereas others use one or only a few. If you are able to make floppies yourself, those with more boot floppy choices *may* prove easier to configure for your hardware. The **Boot Images** line doesn't count duplicates for different size drives; it only counts the choices for one size of floppy drive.

All the distributions provide at least the option of booting from floppy, and all of them provide a way to make your own boot floppies from the CD-ROM under DOS and/or Linux. Some also provide some assistance (under DOS, Linux, or both) to make choosing and/or creating the floppy or floppy set easier. For instance, Red Hat provides a perl script which asks questions about the configuration you are interested in and then creates disks appropriate for your hardware configuration.

The **number of floppies required** to install the system varies widely. Debian installs its entire “base system” from a set of five floppies and then requires one blank floppy after the install to create an appropriate boot disk. If you boot the Linux Universe CD-ROM from DOS, it can install without any floppies at all.

While most of this article has assumed that you are installing from CD-ROM (after all, that's what most of the vendors provide), that's not necessarily true. You may wish to copy packages to a subdirectory of a local hard drive and

install from there, or you may wish to install from floppy or via NFS from a local server or via anonymous FTP or from a tape.

In order to install on most notebooks, support for PCMCIA (also known as PC Card) is required, unless you are want to install one of the distributions that are available on floppies and are interested in shuffling floppies for hours. If you have a PCMCIA Ethernet or SCSI adaptor supported by the Linux PCMCIA card services package, some distributions make it possible for you to do a network installation over Ethernet, or a install from a SCSI CD-ROM connected to a SCSI adaptor. This can be a real time-saver.

If you don't want to partition your disk, some distributions provide the option of installing in a **LINUX** subdirectory of a DOS file system using a Linux file system called UMSDOS. While this imposes on Linux all the inefficiencies of the DOS file system, it is an easy way to get started using Linux without committing a partition to do so. Yggdrasil goes one step farther, allowing you to install the base Linux system without booting Linux, by using a DOS program to do the installation.

Most distributions now have customized documentation written about them, which is either included with the distribution by default or is available as an option. Some distributions have been covered in other books available in the bookstore; we have mentioned their presence under **Alternate Books**. Some distributions are covered by freely-available documentation; the **Installation HOWTO** and *Linux Installation and Getting Started* currently cover Slackware primarily, and Debian's manual is distributed freely via the Internet, including an HTML version on www.debian.org.

Some distributions provide the capability to run nearly or completely from CD-ROM, which can be an effective way to demonstrate or experiment with Linux.

One of the most difficult parts of configuring a Linux system is configuring the X Window System. The XFree86 project, Inc. provides a configuration program called xf86config, which has a fairly basic user interface and requires you to know a lot about your configuration, but is quite complete. Some distributions provide tools to make X configuration easier.

Some vendors provide a support program, and some provide several different support programs. This chart only mentions whether at least one program is available—check with the vendors if you have specific requirements that you need met.

While all of these distributions are available via CD-ROM, some are also sold in other formats. There is still one third-party vendor (Linux System Labs) which

provides Slackware on floppy, and several of the vendors provide their distribution over the Internet via anonymous FTP.

Some of the distributions include configuration tools, mostly graphical ones. If you are not familiar with administering a Linux or Unix system, you are likely to find most common tasks simplified by a user interface that handles common tasks easily. None of the configuration tools provided with any of the distributions prevents you from doing your configuration by hand, so there is no reason to complain if you prefer to configure your system by hand.

In particular, we looked for configuration tools for setting up and maintaining a network, adding and deleting users and groups, mounting and un-mounting file systems, setting up print services, and manipulating boot scripts (generally kept in /etc/init.d). Again, we did not try to rank them in usefulness, and some of the scripts were definitely more functional for our purposes. On the other hand, a script with more functionality than one user needs could include needless, confusing detail for another user. Configuration scripts should have a review of their own some day, but mentioning whether or not they exist is a first step.

Last, but not least, some distributions require more memory to install than others. Some distributions install with only 4MB of memory, but others require 8MB. As new versions of the distributions are released, their memory requirements may change. If you have less than 8MB of memory, it will probably be worth your while to find out how much memory is currently required by each of the distributions you are considering.

Summary

Linux distributions have changed a lot in the few years they have been around. The Linux File system Standard has improved their ability to inter-operate; binaries compiled for one distribution are now more likely to run on another distribution, because standard files are more likely to be in the same place from one distribution to another. As distributions have competed, bugs have been worked out of all of them; thus users have benefited from distributions they haven't even used.

Since no single distribution provides an optimal environment for everyone, each distribution contributes something to the pot. Furthermore, having distributions targeted for different kinds of users means that no distribution is pressured into complete mediocrity by the attempt to be all things to all people. It's a well-worn clich that to do all things equally well is to do nothing at all well...

Archive Index Issue Table of Contents

Advanced search

Graphics Formats for Linux

Gerald Graef

Issue #23, March 1996

Gerald Graefs provides a comprehensive guide through the jumble of graphics formats currently available.

If there is a single unifying feature of the computer world, it is non-conformity. In no place is this better illustrated than computer graphics. Literally dozens of different formats exist to do the same thing: store graphic images. Proprietary formats, independent free standards, and international standards all vie for attention. Fortunately, the computer community has settled on just a handful of formats. Some, like TIFF, are the jacks-of-all-trades and are widely used. Others, like GIF, have a niche that they dominate. But whatever the context, there is always more than one way to store images.

This article first provides a brief introduction to graphics on Linux, including format and type, compression, file structure, and various features. It then continues with descriptions of the common graphics formats found in the Linux world and programs with which to manipulate them.

Format

Graphics formats come in two basic varieties: **raster** and **vector**. Raster formats store the entire image pixel by pixel and are the most commonly-used formats. Vector formats, also called descriptive formats, do not contain pixel image data; instead, they contain a series of commands telling a display program how to draw the image.

The most common use of vector formats is in computer aided drafting (CAD). Describing a line, for example, as a starting point, an ending point, and a color, as opposed to many colored points in a field of points, allows a program to manipulate the line as an object, making it easy to change color, length, and position. Another common use is in raytracing programs. These programs take geometric objects and trace the path of light rays to find shadows and

highlights. Since the rendered objects are geometric shapes, it is much easier to store an image as “dodecahedron at (3,4,3) with size 2 and color 27” than as an array of pixel values, especially since the scene to be rendered is three-dimensional. Of course, once the scene is raytraced, the resulting image is stored in a raster format.

Type

Images come in four basic types: monochrome (black and white), grayscale, color palette, and true color.

Grayscale and color palette images typically have 2, 4, or 8 bits per pixel. They use lookup tables that must be included in the image file; each entry in the color lookup table gives the red, green, and blue values for a color. Grayscale images have a similar lookup table for gray values. Each color or gray value is in a range of 0-3 for 2-bit images, 0-15 for 4-bit images, and 0-255 for 8-bit images. The actual image pixel data is a series of color or gray values corresponding to table entries which give the actual color of each pixel.

True color images are 24-bit images and are usually stored as 8 bits of red, 8 bits of green, and 8 bits of blue per pixel. Some formats may use different color models than RGB. The JPEG format, for example, uses a colorspace based on the luminance and two chrominance values (YCbCr). True color images do not need a color lookup table or palette.

Compression

Compression is integral to most graphics formats. An uncompressed 640x400x256 color image requires just over 256,000 bytes, while a true color image of the same size requires three-quarters of a megabyte. Although many compression schemes exist, only a few are common in graphics applications. LZW, LZ77, Huffman, and run-length encoding are lossless, meaning that the image is reproduced exactly when uncompressed. The cosine transform is the basis for the “lossy” JPEG format. We aren't concerned with the details of the compression schemes here.

JPEG gives by far the best compression, but at the expense of exact image reproduction and speed. Acceptable results are common with compression ratios of 25 to 1, that is, the compressed file is 25 times smaller than the original. Ratios of 100 to 1 may be possible if image quality is not a primary concern.

[FOOTNOTE:A better algorithm does exist, however: fractal image compression. This method gives good results at 100 to 1 and is finding many applications,

though it has not yet been widely implemented as a general purpose graphics format.]

LZW and LZ77 are the best lossless schemes and produce similar results: usually around 2 to 1 compression. The modified LZ77 used in the PNG graphics format is slightly slower at compressing images than LZW, but it creates smaller files and is faster at decompressing. LZ77 is also used by the general purpose compression programs ZIP, GZIP, and PKZIP. Until recently, the use of LZW was much more common than LZ77. Its use is falling, however, because it is a patented algorithm, and UNISYS requires royalties for its use. Nonetheless, both GIF and TIFF, two of the three most common formats, use it.

Run length encoding (RLE) is the most common scheme and the least effective. It is, however, very fast compared to other compression algorithms. Its basic mechanism is very simple: when a string of pixels with the same value is encountered, RLE outputs the pixel value and the number of consecutive pixels with that value. Complex images do not compress well with this scheme, although monochrome images often compress very well.

Structure

Simple formats consist of nothing more than the image data and a short header giving the minimum amount of information necessary to display the image: size and number of colors. More robust formats often employ a tag or chunk structure. After an initial header, the image is stored in a series of chunks, each prefixed by a code telling the decoding software what the chunk contains. The biggest chunk is the actual image data. Other chunks may include comment blocks, palette information, or other special image information. Software may only support some chunks of a given format: if an unknown chunk is encountered, it is simply skipped.

Finally, three special features bear closer examination: magic numbers for format identification, gamma correction, and alpha transparency masks. In some cases it is possible to identify the format of a file by looking at the file itself. Many formats use magic numbers—special (unique, it is hoped!) codes—to identify the format. GIF files, for example, have "GIF87a" or "GIF89a" at the start of the file. Because these magic numbers are ASCII coded, using **less**, **strings** or even **cat** (though that can accidentally change your character set) to display them on the screen is good enough for identifying them. For example, to look for GIF files, you can use **strings *filename* | grep GIF**.

Gamma correction is an important part of the proper display of an image. Whenever an image is displayed on a monitor, the monitor's characteristics affect the image. How the monitor handles varying degrees of brightness is especially important. In general, the brightness on the screen is related to the

brightness levels of the original image by a simple formula: screen brightness equals image brightness raised to the gamma power. The value of gamma for most monitors is around 2.5. To compensate for this, image capture hardware and software may make an “opposite” operation on the image. The result is an image with a gamma of 0.45. When this image is displayed, the gamma of 0.45 and 2.5 cancel each other out, producing an apparent gamma of about 1 (called linear brightness scale).

But what does gamma mean visually? An image displayed with a gamma less than one uses more of its pixel codes in the darker areas, that is, darker areas have better color resolution. A gamma value greater than one uses more of its pixel codes in the lighter regions. A murky image, or an image with too much contrast, may need to have its gamma corrected. Correcting the gamma of an image is inherently lossy because of round-off error during the taking of powers; hence, gamma correction should be performed when the image is displayed, keeping the original image file intact without loss.

Alpha transparency masks are a way of hiding, or masking, parts of an image. In addition to image data, a value is included in the image file for every pixel. A value of 2bitdepth-1 is opaque and the pixel is displayed normally. A value of 0 is transparent, allowing the background color of the screen to show through—the image is not visible at all. The simplest application is to make a non-rectangular image. Define an array the size of the image and then draw a line between two opposite corners. Set the value of each point above the line to one, and the values below to zero. The result will be a triangular image.

The Formats

BMP

BMP is the native bitmapped format used by Microsoft Windows. It is a minimal format that has few features and uses simple run length encoding for data compression. With the widespread use of Windows, BMP is the most common format. In many ways BMP is very similar to the PCX format, and has assumed the role that PCX once held.

BMP encodes 1, 2, 4, 8, and 24 bit images. OS/2 uses a similar BMP format, the only difference being a slightly simpler header. The first two characters of the BMP header are always “BM”.

WMF

Another format native to Microsoft Windows is the Windows MetaFile (WMF). WMF uses Windows' Graphical Device Interface (GDI) function calls to store images that appear repeatedly in applications. The GDI calls provide support

for setting up the screen, defining regions, colors, text, pixels, lines, polygons and bitmaps. WMF supports uncompressed monochrome, color palette, and true color images.

GIF

The Graphics Interchange Format (pronounced “jiff”) was originally developed by the Compuserve Information Service as a color replacement for the earlier RLE format. Since the 1987 release, GIF has become the standard for graphics interchange, especially on networks. A second release in 1989 added several new features to the format. GIF is a lossless format based on the LZW compression scheme. It uses a tag system to identify extension blocks in the file, although only a few tags have been defined. The biggest assets of the format are its relative simplicity, excellent compression, and widespread availability.

However, the format has two major drawbacks. First, it is based on a copyrighted compression scheme and commercial software using it must pay royalties to Unisys, the patent holder. Second, GIF can be used only for images with 256 or fewer colors. (Strictly speaking, that is. Since GIF supports local color maps, the palette can be changed in the middle of an image allowing for more than 256 colors. Unfortunately, much of the available software does not support, or supports poorly, this feature, and it is, at best, clumsy to use.) GIF files can be identified by the string **GIF87a** or **GIF89a** at the start of the file.

JPEG/JFIF

JPEG (pronounced “jay-peg”) is a standard for lossy compression of images. JPEG achieves compression by breaking the image into 8x8 boxes of pixels, performing a mathematical operation called a cosine transform on each, and throwing out the high frequency/small detail components; the more components thrown out, the greater the compression and the poorer the image quality. The remaining frequency components are then run length encoded and compressed with the Huffman algorithm. To view an image, a JPEG file must be uncompressed, decoded, and an inverse cosine transform performed. This three-step process makes displaying JPEGs very slow. Fortunately, JPEG produces excellent compression with little or no visible image loss.

JPEG itself is not actually an image format but a set of standards for compressing image data. JFIF, the JPEG File Interchange Format, is the format commonly referred to as JPEG. JFIF does not support all of the features of JPEG, but is intended as a minimal implementation for image transfer. A full featured implementation of JPEG is included in version 6.0 of the TIFF format. Designers

hope the two implementations of JPEG, one minimal and one full featured, will deter software vendors from defining proprietary formats based on JPEG (early Macintosh versions were especially notorious for incompatibilities).

JPEG works best on real world images; line drawings and cartoons will not compress nearly as well as scanned images. Unlike most other formats, JPEG does not store a pixel as red, green, and blue values. Instead it uses a format called YCbCr. Since most display hardware uses RGB values, the YCbCr values must be converted—yet another step slowing decompression. The JFIF header includes the ASCII characters “JFIF” for format identification.

PBM/PGM/PPM

These three formats are intermediate formats used by the PBMPLUS utilities. The acronyms stand for Portable BitMap/GrayMap/PixMap. PBM is for monochrome images, PGM for grayscale images with up to 256 shades of gray, and PPM for color images using up to 24-bits of true color. A fourth “format” is the Portable AnyMap, PNM. PNM is not actually a format itself. A program that uses PNM can read and write PBM, PGM, and PPM files. PNM is used for utility programs that support multiple image types. For instance, since the image type of a TIFF file may not be known, PNM reads the TIFF file and writes the appropriate file type.

Each of the four formats can read the other ones that carry less information. That is, a PGM utility reads PGM and PBM, a PPM utility reads PPM, PGM, and PBM. PBM, PGM, and PPM utilities always write in their own format, while PNM utilities generally write whatever format they have read. The formats store data either as ASCII or binary data and are otherwise basic formats consisting of a header and image data. The header consists of a magic number to identify the format, image size, and (except for PBM) the number of colors/gray shades. The magic strings are **PBM P1(P4)**, **PGM P2(P5)**, and **PPM P3(P6)**, where the first code is the code for ASCII data, and the code in parentheses is for binary data. True color images store pixel data as a triplet of numbers for RGB data. For more information on the PBMPLUS package, see the section on software below.

PCX

The PCX format is the native format of Z-Soft's PC Paintbrush program and uses run length encoding. As one of the first general purpose formats, its use has been very widespread: few programs exist that do not recognize it. PCX is a basic format consisting of a 128 byte header followed by image data. Monochrome as well as 4, 8, and 24-bit color images are supported. The palette for 4-bit images is included in the header while the 8-bit palette is appended after the image data. This non-uniformity is the result of an older format being

updated for newer hardware. For this reason, the use of PCX has dwindled in favor of the more coherent and unified BMP.

PNG

Amazingly, with all the other formats available, there has been an important niche left unfilled: a portable, relatively simple, free standard for the lossless exchange of true color images—in short, a 24-bit version of GIF with a non-patented compression algorithm. There are existing formats that can be used, but only TIFF has the necessary features, and it suffers from over-completeness: very few implementations make use of the entire TIFF specification. What is needed is a format that is simple enough that any image saved under it can be read by any viewer supporting it.

Compuserve called their development specification GIF24, but when a group of Internet graphics experts developed PNG (Portable Network Graphics, pronounced “ping”), Compuserve adopted it as the successor to GIF. PNG is a natural extension to GIF, although it is not backwards compatible because of a change in compression scheme. In addition to the original GIF features, PNG supports true color images up to 48 bits and grayscale images to 16 bits, as well as full alpha-channel, gamma correction, and detection of file corruption. On 8 bit and larger data, PNG can use a preprocessor on the image data prior to compressing. In many cases this processing improves the compression efficiency and results in smaller file sizes. Expect to see PNG files appearing at an archive near you soon.

PS/EPS

PostScript (PS) is a page description language developed by Adobe Systems that is both a descriptive and raster format and has become one of the most common printer languages. PostScript files are created by many application programs as a device-independent output format. Encapsulated PostScript (EPS) is a limited version of postscript for single pictures and is used for images to be included, or “encapsulated”, in programs or postscript files. The first line of a postscript file is a line of the form:

```
%!PS-Adobe-3.0
```

where the number refers to the PostScript version the file was created under. EPS files append **EPSF-3.0** to this line. PostScript is a large and versatile language. Display PostScript is an implementation of PostScript for controlling video hardware and is used by NeXT computers and software.

TGA

The Targa Truevision graphics format was developed for use in Truevision's product line. TGA uses run length encoding to store grayscale, color table, and true color images, and can include comments, gamma, alpha, color corrections, and a “postage stamp” version of the image. TGA was one of the first true color formats and is still used by some applications like the Persistence of Vision Ray Tracer. TGA files may contain a block at the end of the file that includes the text **TRUEVISION-XFILE**.

TIFF

The Tag Image File Format is the most robust format. As its name suggests, TIFF makes liberal use of the tag concept. TIFF files are typically read using random access because the tag fields can come in any order; an image file directory provides offsets to the location of data in the file. (Even the directory can be anywhere in the file! A short header gives the directory location in the file.) The TIFF format defines several classes of image data: bilevel (monochrome), 4 to 8-bit grayscale or color palette, 24-bit RGB, and 24-bit YCbCr. It supports run length, Huffman, LZW, and JPEG compression. Most implementations do not support all TIFF features, making TIFF a potentially aggravating format. However, TIFF has a huge number of features (implemented in tags) making it unique. Originally intended for desktop publishing, TIFF has spread to video, fax, and document storage, as well as medical and scientific imaging. Because of its complexity, it is not commonly used for home applications.

XBM/XPM

X-Windows defines several formats for internal use. X Bitmap (XBM) is an ASCII format for including 1 bit images in the C source of a program: XBM images are integral to the code and included at compile time, not run time. XBM data is usually included in header files and includes two define statements for the width and height of the image followed by a static unsigned character array. XBM images are often used for icons and cursor bitmaps in X. X Pixmap (XPM) is the equivalent format for color palette images. Its use is identical to XBM except for the addition of the color table and three extra define statements for version number, color table length, and the number of bytes per pixel. X does define a general format for images, the X Window Dump (XWD or WD) format. XWD supports uncompressed raster data of all types.

Programs and Libraries

Having looked at some of the common graphics formats, we finish up with a quick look at some available programs and libraries for converting images.

xv and ImageMagick

These two X programs display, convert, and manipulate images. **xv** can read and write JPEG, GIF, TIFF, PBM/PGM/PPM, and X formats, although it cannot display 24bit images as 24bit. In addition to format conversions, xv can perform simple image manipulations, e.g., rotate, flip, crop, magnify, and gamma correction. ImageMagick supports the formats xv does, plus TGA. It can send output directly to a postscript printer and has more image manipulation capabilities than xv: cut, copy, paste, resize, flip, flop, rotate, invert, emboss, and more, on the whole or on part of the image.

ghostscript

The canonical way to view and convert from PostScript (including EPS) is ghostscript. Ghostscript operates both from the command line options and interactively. Input is always a PostScript file and output may be any of several different file formats, printer languages, or screen types. By default, ghostscript sends its output to an X terminal, but it can also save images to file. Ghostview is a popular front end for ghostscript that improves the screen handling.

PBMPLUS

The PBMPLUS utilities are a set of 120 programs for image conversion and manipulation. PBMPLUS defines three intermediate formats: PBM, PGM, and PPM (see listing above). The basic philosophy is that if there are twenty formats, you need twenty squared, or 400, programs/subroutines to convert between them. The use of intermediate formats reduces that number to two times twenty, or forty. In addition to the conversion programs, there are a number of simple image processing programs: scaling, rotating, smoothing, convolution, gamma, cropping and more. NETPLUS is a newer version of PBMPLUS, but some versions suffer from serious bugs.

C Library support

Substantial support exists for C programmers working with graphics. Libraries for JPEG/JFIF, TIFF, and PNG are now available. In addition, code fragments for many of the other formats are readily available from Internet archive sites.

IJG JPEG/JFIF library

The easiest way to do JPEG programming is to use the Independent JPEG Group's (IJG) library. This library is based on the JFIF specification and includes two common programs found on many Unix systems for storage and retrieval of JFIF images: cjpeg and djpeg. The library is available as source and compiled code.

SGI TIFF library

Like the JPEG library, a full set of routines to implement TIFF is available. Written by Sam Leffler and SGI, this library is also available as source and compiled code. The library includes programs for converting, dithering, splitting, and displaying information on TIFF files.

PNG toolbox

With the recent announcement of support for the PNG format, Compuserve also announced a PNG toolbox. The toolbox uses the zlib library for the LZ77 code and is intended to speed the acceptance of the new format. Its use will be free of royalties. A beta version is available on Compuserve.

Graphics Resources

There are several archive sites on the Internet where programs and further information can be found:

anonymous ftp sites:ftp://sgi.com/graphics/tiff TIFF version 6.0 specification and the source for the SGI TIFF libraryftp://sunsite.unc.edu/pub/Linux/apps/graphics/convert PBMPLUSftp://sunsite.unc.edu/pub/Linux/X11/xapps/graphics ImageMagickftp://sunsite.unc.edu/pub/Linux/libs/graphics Compiled version of JPEG and TIFF libraries for Linuxftp://sunsite.unc.edu/pub/Linux/apps/graphics/viewers Compiled version of ghostscript for Linuxftp://ftp.wuarchive.edu/ Numerous code fragments for image formats are scattered throughout this large archivecompuserve.com GRAPHSUPPORT forum, library 20, LP071.zip, the beta version of the PNG toolboxwww.uwm.edu/~ggraef. List of links and other direct links to format information

Gerald Graef (ggraef@csd.uwm.edu) is a doctoral candidate in theoretical physics who lives on Alpha, Sparc, and Linux computers. His home page is at: www.uwm.edu/~ggraef.

Archive Index Issue Table of Contents

Advanced search

Geomview

Tim Jones

Issue #23, March 1996

Looking for a powerful visualization tool? Geomview, designed to serve as both a general purpose viewer and a visualization tool for mathematics research, may fit the bill. Tim Jones demonstrates Geomview's virtually unlimited applications.

At first glance, Geomview appears to be a neat piece of 3-D software—a toy. In reality, an hour's study of the program and the documentation will convince you that it is an extremely powerful visualization tool with virtually unlimited applications. With the mouse, you can easily rotate, scale, translate, and change the lighting of the objects. You can even “fly” through the scene using a built-in “flight simulator”.

These tools alone are enough to make Geomview worth loading onto your machine, but there is more. Incorporated into Geomview is a Graphical Command Language (GCL) which you can use to manipulate the 3-D objects in more ways than with the mouse alone. Additionally, you can pipe GCL commands into Geomview from other programs, allowing use of the package as a powerful interactive display for your own code.

Geomview was created at the University of Minnesota Geometry Center and was designed to serve as both a general purpose viewer and a visualization tool for mathematics research. Thus, the package also comes with the capability to display representations of 3-D objects as well as the ability to display objects in spherical and hyperbolic spaces. The package also includes code which allows you to use Geomview as the default 3-D viewer for Mathematica.

This article demonstrates how to create and animate objects in Geomview and tells you how to obtain a copy for yourself.

Running Geomview

Geomview comes packed with loads of sample images in the Geomview/data directory. Getting started is as easy as typing **geomview** followed by the path and name of one of the image files. For example, Figure 1 shows the file

/usr/local/Geomview/data/geom/office.oogl

as well as the two other widgets that will appear on the screen when you first start Geomview. If you are running Geomview on a slower machine, you may notice that manipulating the somewhat complex scene of Figure 1 may be a little grueling. You may therefore want to start out with something a little simpler, like:

Geomview/data/things/dragon.off

or the trefoil knot

Geomview/data/things/trefoil_knot.mesh

which are shown in Figure 2.

Performing rotations, translations, and engaging the flight simulator are all pretty straight forward. For example, to rotate a scene displayed by the camera widget, just press the **rotate** button on the tools widget, and drag the mouse in the camera widget while holding down the left mouse button. The scene will rotate about an axis parallel to the camera window. If you let go of the left mouse button, the graphic will continue to spin with a speed proportional to the speed of the mouse when you released the button. (This option can be turned off in the motion menu by toggling to the inertia option if you desire a static view of the scene.) Translating and zooming are just as easy.

Rotating the scene may be a little awkward when you first start using Geomview; however, with a little practice, you will quickly get the hang of manipulating objects. It helps if you imagine that the objects displayed by the camera are contained in a giant imaginary sphere. The mouse is then a "gripper" which will latch onto the sphere when the left mouse button is clicked. Moving the mouse with the gripper engaged will cause the sphere to rotate about an axis parallel to the camera plane. To get another view of the scene, you can rotate about an axis perpendicular to the camera plane by using the middle mouse button to activate the middle gripper instead of the left one. The same applies for translations. To translate along an axis perpendicular to the screen, you use the center mouse button.

You can activate Geomview's flight simulator by pressing the **fly** button in the tools menu, then dragging the mouse from the bottom of the camera widget to the top while holding down the middle mouse button. Again, the speed of your flight will depend on how fast you move the mouse. You can steer through the scene by dragging the mouse on the camera widget and clicking on the left or middle mouse buttons. If you want to stop the motion and take a serious look at something, you can press the halt button, located on the tools widget.

Browsing through the **inspect** menu on the Geomview widget, you will see a sample of the many attributes of the scene that can be selected and changed. Geomview does not limit you to just changing the color of the surfaces or the edges of the objects. It also allows you to control the color, the placement, and the intensity of the lights which illuminate the scene. There are a myriad other useful options which are fairly well described in the manual contained in the Geomview/doc directory.

The OOGL

The machinery of Geomview is contained within the Object Oriented Graphics Library (OOGL) which was also developed by the Geometry Center at the University of Minnesota. All of the scenes that are displayed by Geomview are described in terms of OOGL objects. For example, a sphere in Geomview can be described in terms of the OOGL object called **SPHERE**. The type of object, as well as the physical description, can be passed to Geomview in an OOGL object file which, in the case of a simple sphere, would contain the following text:

```
SPHERE       # name of the OOGL object
1.0          # radius of the sphere
0 0 0        # x,y,z coordinate of the center
```

If you open this file from within Geomview, a unit sphere will appear in the camera widget.

OOGL object files can also be composed of multiple OOGL objects. For example, a file describing a triangle slicing through the center of the above sphere would look like:

```
LIST            # More than 1 OOGL object
{               # Separates the Objects
SPHERE          # A OOGL Object
1.0             # Radius
0 0 0           # Center, x, y, z
}
{ OFF           # Format for describing polyhedra
3 1 3           # Number of vertices, faces, edges
-2.0  2.0  0.0  # vertex 0
 4.0  0.0  0.0  # vertex 1
-2.0 -2.0  0.0  # vertex 2
3 0 1 2         # number of vertices, and order to
                # connect them
}
```

OOGL appears to be fairly complete in the sense that you can describe a variety of objects and their attributes within the context of the library. Other OOGL objects include meshes, Bezier surfaces, lines, and quadrilaterals. Each of these objects has appearance attributes such as surface color, surface reflectance properties, and edge colors that can be specified either from Geomview menus or within the OOGL object files. However, the Linux port (like several others) does not yet support transparency attributes.

The Graphical Command Language

One of the powerful features of Geomview is its graphical command language (GCL), which is rich enough to allow users to not only manipulate the displayed scene, but to change the appearance attributes and animate objects in it. These commands can be either typed directly into Geomview via a command widget or piped from an external program. To illustrate just how easy this is, consider the following example: Say you want to animate a pendulum in the viewer. You will first need to describe the pendulum in terms of OOGL objects and then swing the pendulum by rotating it about an axis that passes through the end of the string from which the bob hangs.

A simple pendulum can be described as two objects, a line for the string and a sphere for the bob. Or, in terms of a list object file which I call pendulum.list:

```
LIST            # More than 1 OOGL object
{
SPHERE          # A OOGL Object
1.0             # radius of the sphere
0.0 0.0 -5.0    # center of the sphere
}
{
VECT            # A vector object
1 2 1           # lines, vertices, colors
2               # vertices in each line
1               # colors in each line
0.0 0.0 0.0     # vertex 0
0.0 0.0 -4.0    # vertex 1
1.0 0.0 0.0 1.0 # color of line
                # (red, green, blue, alpha)
}
```

The **VECT** object describes the string of the pendulum which goes from the origin (vertex 0) to the top of the sphere (vertex 1). The last line describes the color of the string.

For illustrative purposes, I will also put a square 8 units long above the pendulum. This is described in terms of the quadrilateral object (**QUAD**) which I place in a file called ceiling.quad.

```
CQUAD   # QUAD object with color
# vertex descriptions are
# (x, y, z, red, green, blue, alpha)
-4.0 -4.0 0.0  0.0 0.0 1.0 1.0  # vertex 0
 4.0 -4.0 0.0  0.0 0.0 1.0 1.0  # vertex 1
```

```
 4.0  4.0 0.0  0.0 0.0 1.0 1.0  # vertex 2
-4.0  4.0 0.0  0.0 0.0 1.0 1.0  # vertex 3
```

The prefix **C** in front of **QUAD** indicates that the vertex colors are included next to the location of the vertices. That is, the first three numbers on each line describe a vertex of the ceiling, and the last four give the color and the reflection coefficient. All colors are expressed in terms of fractions of red, green, and blue and are usually followed by a reflection coefficient, which is ignored for the Linux port.

To animate the pendulum, a small C program is used to send GCL commands along a pipe connected between Geomview and the program's standard input and output. These commands tell Geomview to not only load the objects into the viewer, but also to rotate the pendulum about the x axis between -30 and +30 degrees in 1 degree increments.

```
#include <stdio.h>
main()
{
  int i,j;
  /* Select no normalization */
  printf("(normalization g0 none)\n");
  /* No bounding boxes */
  printf("(bbox-draw g0 no)\n");
  /* A 8X8 Plane */
  printf("(load ceiling.quad)\n");
  /* A sphere with a string */
  printf("(load pendulum.list)\n");
  /* Flush pipe to Geomview */
  fflush(stdout);
  /* Swing to +30 degrees */
  for (i = 0; i < 30; i++)
    printf("(transform \"pendulum.list\""
           "g0 focus rotate 0.017 0.0 0.0)\n");
  /* Swing to -30 degrees */
  for (i = 0; i < 60; i++)
    printf("(transform \"pendulum.list\""
           "g0 focus rotate -0.017 0.0 0.0)\n");
  /* Swing back to 0 */
  for (i = 0; i < 30; i++)
    printf("(transform \"pendulum.list\""
           "g0 focus rotate 0.017 0.0 0.0)\n");
    /* Flush pipe to Geomview */
    fflush(stdout);
}
```

To see the results, compile the program with **gcc pendulum.c**. Next, start Geomview from this same directory and access the inspect menu and select commands. The command widget will pop up and you need to type **(emodule-define pend a.out)** to instruct Geomview to load the executable a.out as an external module named pend. If everything is working right, the module pend should appear in the external modules list and can be activated by clicking on the word **pend**. The pendulum and ceiling will appear in the camera widget and the pendulum will swing back and forth. You will have to use Geomview's tools to rotate the scene to the view shown in Figure 3.

The loading of the OOGL object files and animation of the above example is accomplished in just a few lines of C code and a total of 4 GCL commands. The

first two GCL commands describe some characteristics of the "World" (**g0**) object which, in turn, are adopted by all objects that are loaded following these commands. By default, when an object is loaded into Geomview, the coordinates of the object are normalized so that the object fits within a unit sphere centered on the origin. The **(normalization g0 none)** command turns this option off. The second command tells Geomview not to draw black bounding boxes around each object that it loads. The third command, **(load filename)**, tells Geomview to load the OOGL object file filename.

Most of the work is actually done by the fourth and final GCL command. **transform** tells Geomview to rotate the object **pendulum.list** by 0.017 radians (about 1 degree) around the x axis and 0 radians about the y and z axes. The other two arguments to **transform**, **g0** and **focus**, specify that rotations are with respect to the center of the World object (**g0**) and are to be applied in the frame of reference of the camera. There are many other GCL commands described in the manual that can be used to animate and manipulate OOGL objects.

Although the examples illustrated here have been rather simple, it should be apparent that Geomview is a package with numerous applications. Geomview, in combination with powerful graphical user interface development tools, such as Tcl and Wish, can be used to create interactive 3-D applications in relatively short amounts of time. This is quite a powerful package that should not be ignored by anyone developing applications involving 3-D images or anyone needing an easy-to-use viewer to better visualize objects, whether he is viewing complex mathematical output from Mathematica or simple geometric relationships between lines and planes.

Getting Geomview

You can find Geomview in several locations, including Sunsite. However, the home of Geomview is the anonymous ftp site ftp.geom.umn.edu. The Linux port can be found at pub/software/geomview/geomview-linux.tar.gz (3.1 MB). Source code, geomview-src.tar.gz, is available in the same directory.

If you have Netscape or Mosaic, I strongly recommend that you visit Geomview's World Wide Web site, which is located at www.geom.umn.edu/software/download/geomview.html. At this site you will find not only executable code, but also an on-line tutorial for OOGL file formats, an FAQ, and introductions to other software. (While you are there, be sure to check out their Interactive Web Applications at www.geom.umn.edu/apps/.)

After Geomview is unpacked, it will occupy approximately 8.3 MB on your disk, including about 1 MB of documentation and 2.2 MB of examples. Unzipping and untarring will result in a new directory called Geomview. Change to this directory, read the README, and type **install**. The install script will ask you

questions about where you want to install the executable, the manual pages, etc. At the end of the script, it will ask if you mind that the install script mails a registration to the Geometry Center. If you do not mind and have a mail facility set up on your machine, install will register you and place you on a very quiet mailing list (So far I have received only the registration confirmation).

Tim Jones (jones@uinpla.npl.uiuc.edu) is a graduate student at the University of Illinois Nuclear Physics Laboratory where he is working on analyzing data from his thesis experiment which was recently completed at CERNs Low Energy Anti-proton Ring.

Archive Index Issue Table of Contents

Advanced search

Ext2tools—Reading Linux files from DOS

Robert Dalrymple

Issue #23, March 1996

Robert admits he uses Windows on his PC along with Linux. Here's how he survives his divided existence.

I don't know about you, but I will admit to using Windows on my PC along with Linux. It's just that some Windows applications are still better than those available for the X windows system. (No flames, please. I didn't mention any names!)

Unfortunately, using both Windows and Linux on the same machine means I sometimes have important files on the MS-DOS partition, which I need while using Linux. I can access files on the MS-DOS partition with Linux, but often the opposite is true—I need to access Linux files from DOS or Windows 95. This used to involve getting out of Windows, booting Linux, moving a file to the DOS partition, and then rebooting Windows. Not a particularly convenient method!

NB: Be sure to have **Loadlin** working or a working Linux boot disk before you try loading Windows 95 if you use LILO to start Linux, as the installation of Windows will overwrite the Master Boot Record.

But it isn't that complicated any more, thanks to ext2tools, a collection of DOS programs written by Claus Tondering (ct@loglin.dknet.ck). ext2tools allows me to read Linux (ext2) files directly from DOS or Windows. Five of ext2tools' DOS programs mimic Unix commands, and a sixth is used for set-up. The programs are E2CAT, E2CD, E2CP, E2LS, E2PART and E2PWD.

Using these commands is straightforward. From DOS, or a DOS window, you get a listing of the files in your Linux root directory with the command **e2ls**. If you want a listing of another directory, just add the directory name.

To change to another Linux directory, say, home, you enter **e2cd home.** Another **e2ls** then lists the files in home. If you find the file you need in the

home directory, say it is called stuff.ext, typing **e2cp stuff.ext stuff.dos** will copy the file into stuff.dos in your present DOS directory. To just look at the file instead, **e2cat stuff.ext** will list it on your DOS screen.

The other DOS programs included in ext2tools are E2PWD, which gives the Linux directory you are currently in, and E2PRT, which is used to determine the partitions on a hard disk, which is to be signified by the disk number discussed below.

The set-up of ext2tools is really simple. It just requires you to add one line to your DOS autoexec.bat file to define the environmental variable e2cwd. The value of this variable is set to a two (or three) part number. I use:

set e2cwd=128:2

The 128 indicates the first hard disk (129 would be the second.). The 2 denotes the partition number of the ext2 file system on that disk. A third number, which I omit, is the inode number of the current working directory. Leaving it blank results in the default value of 2, which identifies the root directory. After re-running autoexec.bat in a DOS window, typing that command on the command line, or simply rebooting DOS/Windows, you should be ready to use the tool program.

Claus Tondering, the author of the program and a Unix programmer/electrical engineer at Olicom in Denmark, says there are two limitations of ext2tools that he knows of. If there are three or more partitions inside an extended partition table, the table is read incorrectly. Secondly, if you have more than two hard disks, there is no way to get to the third, due to limitations on the use of BIOS for disk information. He hopes that the next version should take care of both these problems and may even support wild cards, but no promises.

The access to the Linux file system by ext2tools is read-only. To write Linux files you must be in Linux. In fact, Tondering does not plan to add write capabilities to these DOS programs as he fears that it could be dangerous for a variety of reasons.

Finally, a **big warning**. ext2tools gives superuser status to the DOS user, as it does not respect the access permissions of the Linux file system. This was by design, since having DOS on a Linux system is an inherent security risk anyway. Be careful if you install it on a multi-user system.

See Getting ext2tools sidebar for more information.

Robert A. Dalrymple (rad@udel.edu) works at the University of Delaware and tends two machines that are half-and- half (Win95/Linux). He thanks Claus Tondering for helpful comments.

Running Windows Applications NOW

Ron Bardarson

Issue #23, March 1996

If you need to run MS Windows applications under Linux and haven't been satisfied with the current solutions, here's the road less traveled...

There are three methods of obtaining use of Microsoft (MS) Windows applications under Linux. They are:

Wine, available from ftp://tsx-11.mit.edu:/pub/linux/ALPHA/Wine/development/
DOSEMU, available from ftp://tsx-11.mit.edu:/pub/linux/ALPHA/dosemu/
DESQview/X (DV/X), available from QuarterDeck software

Wine is rather well-known, as is the excellent DOSEMU. While researching these methods, however, I discovered that there exists a wide misunderstanding of their current capabilities and status among Usenet readers. I certainly support the efforts of the Wine and DOSEMU crews to bring MS Windows applications to Linux, but the impression that either is “ready for prime time” only impedes their efforts. Annoyed users complain that the products are, as their developers themselves point out, very incomplete.

For instance, the file dosemu-0.60.2/dos/README.Windows contains this warning under the heading **Windows 3.1 Protected Mode**: WARNING!!! WARNING!!! WARNING!!! WARNING!!! WARNING!!! Danger Will Robinson!!! This is not yet fully supported and there are many known bugs! Large programs will almost certainly NOT WORK!!! BE PREPARED FOR SYSTEM CRASHES IF YOU TRY THIS!!!

The document goes on to list some of the problems; in short, it's not yet ready for serious use, such as most work environments. Take a moment to read that README completely before experimenting. I found it to be accurate.

Wine has a similar status: the release notes proclaim that Wine is currently for developers only, and you should not yet expect your applications to work. Thus,

a Linux user is left with two choices: reboot the system with DOS or use the slower DV/X. Linux users isolated from a network tend to think that rebooting with DOS is Okay, but those needing network services or the power of Linux on a regular basis should consider using DV/X.

DV/X is probably unknown to most *LJ* readers, and the thrust of this article is to introduce you to it. The three possibilities above are listed in order of speed—Wine will certainly be the fastest when it'ss finished. They are listed in reverse order of "success"—DV/X is capable of running nearly every large MS Windows application now. Because I am willing to pay a speed penalty in exchange for access to other Linux services while running MS Word or MS Excel, I have been happily utilizing DV/X for months. I look forward to the day that Wine (or DOSEMU) allows me to run MS Word without the speed penalty.

DV/X is QuarterDeck's solution to bringing DOS to network computing (see ftp.qdeck.com:/pub or www.qdeck.com). QuarterDecki's DOS software line includes the well-known QEMM memory manager and DESQview, the best DOS multi-tasker. DESQview was expanded into a network program several years ago as DESQview/X. This software product brought Unix programs to DOS via an X Window interface and provided DOS programs to networked Unix hosts with X. One of the most interesting features of DV/X is its ability to provide MS Windows applications to networked X boxes. The software had some delays coming to market, probably due to the difficulty of creating a program to translate MS Windows and X. It is now available as version 2.0 and is much improved.

Setup

To run MS Windows applications such as Word 6.0, Excel 5.0, and PowerPoint with DV/X, you need a dedicated DOS computer in addition to your Linux box. I used an old system, a 386DX with 12 Megabytes of RAM, at home and a spare 486DX2 machine at work. A network connection is necessary; I had pairs of SMC Ultra and 3COM 3C503 Ethernet cards. The DOS system doesn't need a fast or powerful video card, as the X display on the Linux box will be where you control MS Windows. The DOS box needs at least 8 MB of RAM, since DV/X can easily use up 5 MB and MS Windows needs several more. QuarterDeck supplies networking software with DV/X as part of its base cost, but it'ss not TCP/IP-based. A Novell TCP/IP stack is available for an extra fee and has been offered several times at no cost as a purchase inducement.

Get your Linux system up with networking active (see the NET-2 HOWTO document). Install DV/X on the DOS box after installing the TCP/IP stack (you will need an ODI driver on the DOS side for a TCP/IP stack). Just follow the prompts (and DV/X manuals) and adjust per your network. On an isolated

network, you may wish to allow access without passwords for easier use. Once Linux has networking running and DV/X is installed, run some ping checks to be sure that communications are working correctly. I get 1 to 2 millisecond ping times on either a dedicated two computer network or a crowded 10-BaseT hub. Try some large FTP transfers to accurately test the network; isolated networks should run hundreds of kilobytes per second when properly configured.

Once you think your “network computer” is ready, launch DV/X on the DOS box and bring up an Xterm on the Linux side. You will need a username besides root on the Linux side in order to simplify logins. Any username can have root's “power” by setting the UID and GID to 0—something that might be convenient for overriding protections during file transfers. Obviously, don't consider doing this if your network is connected in any way to the outside world. Also, if you do this, you are discarding all the file system and memory protections that protect you from damage caused by program bugs, viruses (none are documented to exist under Linux), and simple mistakes.

In the Linux Xterm, type:

```
$ xhost +dosboxname
$ rsh dosboxname fileman
```

where **dosboxname** is the alias for the IP address for the DV/X DOS computer (you will need to have it in /etc/hosts on the Linux computer). After a few moments, you should get a new X-Window with the very useful DV/X File Manager. It's similar to the old Norton Commander, an interface whose look and feel has been duplicated countless times. With the DV/X File Manager, you can log into networked hosts, initiate multiple file transfers, or perform routine disk housekeeping tasks. If the File Manager doesn't work, take a look at whatever messages were displayed and check manuals, Usenet, friends, support lines, and maybe even the author...

Log into the Linux computer from the File Manager (under the Navigate radio button) as a user other than root. Work through the file system until you get to one of the directories in the font path for the X server (I used /usr/X386/lib/X11/fonts/misc). Note that File Manager doesn't show symbolic links, so you need the actual pathname. On the DV/X side, maneuver to the BDF subdirectory under the DVX directory. Copy all the font files over to the Linux side (unless you **never** want to run DOS remotely) with a binary transfer. Check that the file sizes come out identical on both sides of the File Manager dialog box. Once that's complete, you need to run **mkfontdir** in the Linux font directory you chose and restart the X server.

Now you can run remote DOS with:

```
$ rsh dosboxname dos
```

You might need to set up the xhost access list again. Without the transferred fonts, a remote DOS process might crash your X server. Now that you have these preliminary steps accomplished, you know that your “network computer” is working. Time for MS Windows. DV/X has a lot of interesting features, such as the Remote Program Launcher; spend some time exploring them.

MS Windows

Initially, you should start with MS Windows configured to use just a simple VGA Display driver. Getting MS Windows running if you have difficulties is a problem exclusively for their product support personnel. Naturally, if MS Windows doesn't run on the DOS computer, it won't work under DV/X. After getting the networked MS Windows running, you can get screen sizes of 1024x768 if your X display has room. Since DV/X is converting the video commands, you don't need the actual video card drivers. Instead, you use the Windows “Super VGA 1024x768 256 colors Large Fonts” driver. Read the DV/X documentation on setting up WinX for best results. To run MS Windows under Linux X, you can type the following in an xterm:

```
$ rsh dosboxname winx
```

In a few moments, you should be able to launch applications other than Solitaire from MS Windows. You can even start another MS Windows process (if you have sufficient memory) that is completely independent of the first (be careful). You're limited to Standard mode, but so far I haven't come across an application I need that truly requires Enhanced mode. One of the first things you'll need to do is change the MS Windows color scheme with the Control Panel Color tool (In the “Main” window). Otherwise, you may have invisible fonts! I suggest starting with the “Arizona” color scheme instead of the default.

Gotchas

TANSTAAFL: There Ain't No Such Thing As A Free Lunch. As mentioned above, there is a speed penalty for using DV/X to get MS Windows applications under Linux. It's a direct result of the network connection between the involved computers (that's likely to be the slowest part of the pipeline). However, for many uses, that speed penalty may be negligible. Ten Mbit/s Ethernet is not very fast when compared to ISA bus speeds: ISA is 5 times faster than the original Ethernet. There are one hundred Mbit Ethernet cards appearing, for a cost about 2 times the 10 Mbit cards. One hundred Mbit cards should give DV/X MS Windows performance equivalent to an ISA video card, which is quite acceptable to many users. This is the most serious drawback, but it needs to be balanced against the alternatives—in short, your mileage may vary.

As I mentioned, the speed difference may be negligible—you may not even notice it. I still can't type at even one-thousandth Ethernet speed. I find the major use of DV/X MS Windows to be typing large amounts of text without text layout or formatting concerns. The other major use is for tasks of under 30 minutes; in both cases, the speed penalty is not apparent or traded off against the time to change operating systems. The remaining gotchas are somewhat minor.

The DOS ODI driver may have some problems. I notice that the 16-bit Ultra cards don't outperform the 8-bit 3C503 cards. I believe the Ultra ODI driver might be the cause. The network stack you use on the DOS computer can also make a difference—I've seen the best results from the Novell TCP stack. It's RTFM time here.

You may see some color changes. I use fvwm as an X Window manager, and sometimes the MS Window picks up the colors of the fvwm pager when switching between pages. Usually the next dialog box clears the colors up. There may also be some rearranging of your X workplace in order to fit MS Windows in at higher resolutions. The Alt-F4 keystroke will no longer close the MS Windows application. Instead, under the standard fvwm setup, it will iconify the window. Double-clicking the corner of the menu bar still works nicely to terminate applications. The usual problems between DOS filename limitations and Unix file names exist during file transfer.

Conclusion

I use DV/X regularly after months of experimenting—in fact, I booted DOS last month and realized it had been 6 weeks since I had last ran DOS. DV/X is as stable as MS Windows (actually, a little better, since DESQview isolates tasks in a fashion similar to Linux). There are some silver linings in the slow speed cloud. DV/X File Manager is a very useful tool; we have an Apollo workstation that exercises it frequently. DV/X can provide remote printing services, making the Linux end a snap to configure. Oh, and yes, I used my network to assist in preparing this article. See Figure 1.

I've set up two networks with DV/X and Linux; most of it runs “right out of the box”. I've been using Linux at work for 10 months now and have only rarely been forced back into DOS. In fact, Linux applications have provided better answers to tasks than the less stable DOS equivalents on several occasions. Source code availability and group programming naturally lead to superior programs!

Ron Bardarson (ronb@lenin.svl.trw.com) plays with Linux nowadays; eleven years ago, he was hacking ZCPR3 (he wrote its Make utility, MAKE.RCP).

Network computing and studying interesting applications of numbers divert him from work, kids, and dancing.

Archive Index Issue Table of Contents

Advanced search

Dynamic Kernels: Modularized Device Drivers

Alessandro Rubini

Issue #23, March 1996

This is the first in a series of four articles co-authored by Alessandro Rubini and Georg Zezchwitz which present a practical approach to writing Linux device drivers as kernel loadable modules. This installment presents and introduction to thte topic, preparing the reader to understand next month's installment.

Kernel modules are a great feature of recent Linux kernels. Although most users feel modules are only a way to free some memory by keeping the floppy driver out of the kernel most of the time, the real benefit of using modules is support for adding additional devices without patching the kernel source. In the next few Kernel Korners Georg Zezschwitz and I will try to introduce the “art” of writing a powerful module—while avoiding common design errors.

What is a device?

A device driver is the lowest level of the software that runs on a computer, as it is directly bound to the hardware features of the device.

The concept of “device driver” is quite abstract, actually, and the kernel can be considered like it was a big device driver for a device called “computer”. Usually, however, you don't consider your computer a monolithic entity, but rather a CPU equipped with peripherals. The kernel can thus be considered as an application running on top of the device drivers: each driver manages a single part of the computer, while the kernel-proper builds process scheduling and file-system access on top of the available devices.

See Figure 1.

A few mandatory drivers are “hardwired” in the kernel, such as the processor driver and the the memory driver; the others are optional, and the computer is usable both with and without them—although a kernel with neither the console driver nor the network driver is pointless for a conventional user.

The description above is somehow a simplistic one, and slightly philosophical too. Real drivers interact in a complex way and a clean distinction among them is sometimes difficult to achieve.

In the Unix world things like the network driver and a few other complex drivers belong to the kernel, and the name of *device driver* is reserved to the low-level software interface to devices belonging to the following three groups:

character devices
: Those which can be considered files, in that they can be read-from and/or written-to. The console (i.e. monitor and keyboard) and the serial/parallel ports are examples of character devices. Files like /dev/tty0 and /dev/cua0 provide user access to the device. A char device usually can only be accessed sequentially.

block devices
: Historically: devices which can be read and written only in multiples of the block-size, often 512 or 1024 bytes. These are devices on which you can mount a filesystem, most notably disks. Files like /dev/hda1 provide access to the devices. Blocks of block devices are cached by the **buffer cache**. Unix provides uncached character devices corresponding to block devices, but Linux does not.

network interfaces
: Network interfaces don't fall in the device-file abstraction. Network interfaces are identified by means of a name (such as eth0 or plip1) but they are not mapped to the filesystem. It would be theoretically possible, but it is impractical from a programming and performance standpoint; a network interface can only transfer packets, and the file abstraction does not efficiently manage structured data like packets.

The description above is rather sketchy, and each flavour of Unix differs in some details about what is a block device. It doesn't make too much a difference, actually, because the distinction is only relevant inside the kernel, and we aren't going to talk about block drivers in detail.

What is missing in the previous representation is that the kernel also acts as a library for device drivers; drivers request services from the kernel. Your module will be able to call functions to perform memory allocation, filesystem access, and so on.

As far as loadable modules are concerned, any of the three driver-types can be constructed as a module. You can also build modules to implement filesystems, but this is outside of our scope.

These columns will concentrate on character device drivers, because special (or home-built) hardware fits the character device abstraction most of the time. There are only a few differences between the three types, and so to avoid confusion, we'll only cover the most common type.

You can find an introduction to block drivers in issues 9, 10, and 11 of *Linux Journal*, as well as in the *Linux Kernel Hackers' Guide*. Although both are slightly outdated, taken together with these columns, they should give you enough information to get started.

What is a module?

A module is a code segment which registers itself with the kernel as a device driver, is called by the kernel in order to communicate with the device, and in turn invokes other kernel functions to accomplish its tasks. Modules utilize a clean interface between the “kernel proper” and the device, which both makes the modules easy to write and keeps the kernel source code from being cluttered.

The module must be compiled to object code (no linking; leave the compiled code in .o files), and then loaded into the running kernel with **insmod**. The insmod program is a **runtime linker**, which resolves any undefined symbols in the module to addresses in the running kernel by means of the kernel symbol table.

This means that you can write a module much like a conventional C-language program, and you can call functions you don't define, in the same way you usually call **printf()** and **fopen()** in your application. However, you can count only on a minimal set of external functions, which are the *public* functions provided by the kernel. **insmod** will put the right kernel-space addresses in your compiled module wherever your code calls a kernel function, then insert the module into the running Linux kernel.

If you are in doubt whether a kernel-function is public or not, you can look for its name either in the source file /usr/src/linux/kernel/ksyms.c or in the run-time table /proc/ksyms.

To use make to compile your module, you'll need a Makefile as simple as the following one:

```
TARGET = myname

ifdef DEBUG
  # -O is needed, because of "extern inline"
  # Add -g if your gdp is patched and can use it
  CFLAGS = -O -DDEBUG_$(TARGET) -D__KERNEL__ -Wall
else
```

```
    CFLAGS = -O3 -D__KERNEL__ -fomit-frame-pointer
endif

all: $(TARGET).o
```

As you see, no special rule is needed to build a module, only the correct value for **CFLAGS**. I recommend that you include debugging support in your code, because without patches, gdb isn't able to take advantage of the symbol information provided by the **-g** flag to a module while it is part of the running kernel.

Debugging support will usually mean extra code to print messages from within the driver. Using **printk()** for debugging is powerful, and the alternatives are running a debugger on the kernel, peeking in /dev/mem, and other extremely low-level techniques. There are a few tools available on the Internet to help use these other techniques, but you need to be conversant with gdb and be able to read real kernel code in order to benefit from them. The most interesting tool at time of writing is kdebug-1.1, which lets you use gdb on a running kernel, examining or even changing kernel data structures (including those in loaded kernel modules) while the kernel is running. *Kdebug* is available for ftp from sunsite.unc.edu and its mirrors under /pub/Linux/kernel.

Just to make things a little harder, the kernel equivalent of the standard **printf()** function is called **printk()**, because it does not work exactly the same as **printf()**. Before 1.3.37, conventional **printk()**'s generated lines in **/var/adm/messages**, while later kernels will dump them to the console. If you want quiet logging (only within the messages file, via syslogd) you must prepend the symbol **KERN_DEBUG** to the format string. **KERN_DEBUG** and similar symbols are simply strings, which get concatenated to your format string by the compiler. This means that you must *not* put a comma between **KERN_DEBUG** and the format string. These symbols can be found in **<linux/kernel.h>**, and are documented there. Also, **printk()** does not support floating point formats.

Remember, that syslog write to the messages file as soon as possible, in order to save all messages on disk in case of a system crash. This means that an over-printk-ed module will slow down perceptibly, and will fill your disk in a short time.

Almost any module misbehaviour will generate an [cw]Oops[ecw] message from the kernel. An **Oops** is what happens when the kernel gets an exception from kernel code. In other words, **Oops**es are the equivalent of segmentation faults in user space, though no core file is generated. The result is usually the sudden destruction of the responsible process, and a few lines of low-level information in the messages file. Most **Oops** messages are the result of dereferencing a **NULL** pointer.

This way to handle disasters is a friendly one, and you'll enjoy it whenever your code is faulty: most other Unices produce a kernel panic instead. This does not mean that Linux never panics. You must be prepared to generate panics whenever you write functions that operate outside of a process context, such as within interrupt handlers and timer callbacks.

The scarce, nearly unintelligible information included with the [cw]Oops[ecw] message represents the processor state when the code faulted, and can be used to understand where the error is. A tool called ksymoops is able to print more readable information out of the oops, provided you have a kernel map handy. The map is what is left in /usr/src/linux/System.map after a kernel compilation. Ksymoops was distributed within util-linux-2.4, but was removed in 2.5 because it has been included in the kernel distribution during the linux-1.3 development.

If you really understand the **Oops** message, you can use it as you want, like invoking gdb off-line to disassemble the whole responsible function. if you understand neither the **Oops** nor the ksymoops output, you'd better add some more debugging **printk()** code, recompile, and reproduce the bug.

The following code can ease management of debugging messages. It must reside in the module's public include file, and will work for both kernel code (the module) and user code (applications). Note however that this code is gcc-specific. Not too big a problem for a kernel module, which is gcc-dependent anyway. This code was suggested by Linus Torvalds, as an enhancement over my previous ansi-compliant approach.

```
#ifndef PDEBUG
#  ifdef DEBUG_modulename
#    ifdef __KERNEL__
#      define PDEBUG(fmt, args...) printk (KERN_DEBUG fmt , ## args)
#    else
#      define PDEBUG(fmt, args...) fprintf (stderr, fmt , ## args)
#    endif
#  else
#    define PDEBUG(fmt, args...)
#  endif
#endif

#ifndef PDEBUGG
#  define PDEBUGG(fmt, args...)
#endif
```

After this code, every **PDEBUG("any %i or %s...\n", i, s);** in the module will result in a printed message only if the code is compiled with **-DDEBUG_*modulename***, while **PDEBUGG()** with the same arguments will expand to nothing. In user mode applications, it works the same, except that the message is printed to **stderr** instead of the messages file.

Using this code, you can enable or disable any message by removing or adding a single **G** character.

Writing Code

Let's look at what kind of code must go inside the module. The simple answer is “whatever you need”. In practice, you must remember that the module is kernel code, and must fit a well-defined interface with the rest of Linux.

Usually, you start with header inclusion. And you begin to have contraints: you must always define the **__KERNEL__** symbol before including any header unless it is defined in your makefile, and you must *only* include files pertaining to the **<linux/*>** and **<asm/*>** hierarchies. Sure, you can include your module-specific header, but never, ever, include library specific files, such as **<stdio.h>** or **<sys/time.h>**.

The code fragment in Listing 1 represents the first lines of source of a typical character driver. If you are going to write a module, it will be easier to cut and paste these lines from existing source rather than copying them by hand from this article.

```
#define __KERNEL__          /* kernel code */

#define MODULE              /* always as a module */
#include <linux/module.h>   /* can't do without it */
#include <linux/version.h> /* and this too */

/*
 * Then include whatever header you need.
 * Most likely you need the following:
 */
#include <linux/types.h>    /* ulong and friends */
#include <linux/sched.h>    /* current, task_struct, other goodies */
#include <linux/fcntl.h>    /* O_NONBLOCK etc. */
#include <linux/errno.h>    /* return values */
#include <linux/ioport.h>   /* request_region() */
#include <linux/config.h>   /* system name and global items */
#include <linux/malloc.h>   /* kmalloc, kfree */

#include <asm/io.h>         /* inb() inw() outb() ... */
#include <asm/irq.h>        /* unreadable, but useful */

#include "modulename.h" /* your own material */
```

After including the headers, there comes actual code. Before talking about specific driver functionality—most of the code—it is worth noting that there exist two module-specific functions, which must be defined in order for the module to be loaded:

```
int init_module (void);
void cleanup_module (void);
```

The first is in charge of module initialization (looking for the related hardware and registering the driver in the appropriate kernel tables), while the second is in charge of releasing any resources the module has allocated and deregistering the driver from the kernel tables.

If these functions are not there, **insmod** will fail to load your module.

The **init_module()** function returns 0 on success and a negative value on failure. The **cleanup_module()** function returns **void**, because it only gets invoked when the module is known to be unloadable. A kernel module keeps a usage count, and **cleanup_module()** is only called when that counter's value is 0 (more on this later on).

Skeletal code for these two functions will be presented in the next installment. Their design is fundamental for proper loading and unloading of the module, and a few details must dealt with. So here, I'll introduce you to each of the details, so that next month I can present the structure without explaining all the details.

Getting a major number

Both character drivers and block drivers must register themselves in a kernel array; this step is fundamental for the driver to be used. After **init_module()** returns, the driver's code segment is part of the kernel, and won't ever be called again unless the driver registers its functionality. Linux, like most Unix flavors, keeps an array of device drivers, and each driver is identified by a number, called the major number, which is nothing more than the index in the array of available drivers.

The major number of a device is the first number appearing in the output of **ls -l** for the device file. The other one is the minor number (you guessed it). All the devices (file nodes) featuring the same major number are serviced by the same driver code.

It is clear that your modularized driver needs its own major number. The problem is that the kernel currently uses a static array to hold driver information, and the array is as small as 64 entries (it used to be 32, but it was increased during the 1.2 kernel development because of lack of major numbers).

Fortunately, the kernel allows dynamic assignment of major numbers. The invocation of the function

```
int register_chrdev(unsigned int major,
                    const char *name,
                    struct file_operations *fops);
```

will register your char-driver within the kernel. The first argument is either the number you are requesting or 0, in which case dynamic allocation is performed. The function returns a number less than 0 to signal an error, and 0 or greater to signal successful completion. If you asked for a dynamically assigned number, the positive return value is the major number your driver was assigned. The **name** argument is the name of your driver, and is what

appears within the /proc/devices file. finally, **fops** is the structure used for calling all the other functions in your driver, and will be described later on.

Using dynamic allocation of major numbers is a winning choice for custom device drivers: you're assured that your device number doesn't conflict with any other device within your system—you're assured that **register_chrdev()** will succeed, unless you have loaded so many devices that you have run out of free device numbers, which is unlikely.

Loading and unloading

Since the major number is recorded inside the filesystem node that applications use to access the device, dynamic allocation of the major number means that you can't create your nodes once, and keep them in /dev forever. You need to recreate them each time you load your module.

The scripts in this page are the ones I use to load and to unload my modules. A little editing will suit your own module: you only need to change the module name and the device name.

The **mknod** command creates a device node with a given major and minor number (I'll talk about minor numbers in the next installment), and **chmod** gives the desired permissions to the new devices.

Though some of you may dislike creating (and changing permissions) any time the system is booted, there is nothing strange in it. If you are concerned about becoming root to perform the task, remember that insmod itself must be issued with root privileges.

The loading script can be conveniently called *drvname*_load, where *drvname* is the prefix you use to identify your driver; the same one used in the **name** argument passed to **register_chrdrv()**. The script can be invoked by hand during driver development, and by rc.local after module installation. Remember that insmod looks both in the current directory and in the installation directory (somewhere in /lib/modules) for modules to install.

If your module depends on other modules or if your system setup is somehow peculiar, you can invoke modprobe instead of insmod. The modprobe utility is a refined version of insmod which manages module dependencies and conditional loading. The tool is quite powerful and well documented. If your driver needs exotic handling, you're better off reading the manpage.

At time of writing, however, none of the standard tools handles generation of device nodes for automatically allocated major numbers, and I can't even

conceive how they could know the names and minor numbers of your driver. This means that a custom script is needed in any case.

Here's *drvname*_load:

```
#!/bin/sh
# Install the drvname driver,
# including creating device nodes.

# FILE and DEV may be the same.
# The former is the object file to load,
# the latter is the official name within
#  the kernel.

FILE="drvname"
DEV="devicename"

/sbin/insmod -f $FILE $*  || \
 {echo "$DEV not inserted" ; exit 1}

# retrieve major just assigned
major=`grep $DEV /proc/devices | \
  awk "{print \\$1}"`

# make defice nodes
cd /dev
rm -f mynode0 mynode1

mknod mynode0 c $major 0
mknod mynode1 c $major 1

# edit this line to suit your needs
chmod go+rw mynode0 mynode1
```

And *drvname*_unload:

```
#!/bin/sh
# Unload the drvname driver

FILE="drvname"
DEV="devicename"

/sbin/rmmod $FILE $* || \
 {echo "$DEV not removed" ; exit 1}

# remove device nodes
cd /dev
rm -f mynode0 mynode1
```

Allocating Resources

The next important task of **init_module()** is allocating any resources needed by the driver for correct operation. We call any tiny piece of the computer a “resource”, where “piece” is a logical (or software) representation of a physical part of the computer. Usually a driver will request memory, I/O ports, and IRQ lines.

Programmers are familiar with requesting memory. The **kmalloc()** function will do it, and you can use it exactly like it was **malloc()**. Requesting I/O ports, on the contrary, is unusual. They're there, free of charge. There is no “I/O port fault” equivalent of a “segmentation fault”. However, writing to I/O ports belonging to other devices can still crash your system.

Linux implements essentially the same policy for I/O ports as is used for memory. The only real difference is in the CPU not generating exceptions when you write to a port address that you have not requested. Port registering, like memory registering, is also useful to help the kernel's housekeeping tidy.

If you ever scratched your head about the port address to assign to your newly acquired board, you'll soon forget the feeling: **cat /proc/ioports** and **cat /proc/interrupts** will quickly uncover the secrets of your own hardware.

Registering I/O ports you use is a little more complicated than requesting memory, because you often have to “probe” to find out where your device is. To avoid “probing” ports that other devices have already registered, you can call **check_region()** to ask if the region you are considering looking in is already claimed. Do this once for each region as you probe. Once you find the device, use the **request_region()** function to reserve the region. When your device is removed, it should call **release_region()** to free the ports. Here are the function declarations from **<linux/ioports.h>**:

```
int check_region(unsigned int from,
                 unsigned int extent);
void request_region(unsigned int from,
                    unsigned int extent,
                    const char *name);
void release_region(unsigned int from,
                    unsigned int extent);
```

The **from** argument is the beginning of a contiguous region, or range, of I/O ports, the **extent** is the number of ports in the region, and **name** is the name of the driver.

If you forget to register your I/O ports, nothing bad will happen, unless you have two such misbehaving drivers, or you need the information to fit a new board in your computer. If you forget to release ports when unloading, any subsequent program accessing the /proc/ioports file will “Oops”, because the driver name will refer to unmapped memory. Besides, you won't be able to load your driver again, because your own ports are no longer available. Thus, you should be careful to free your ports.

A similar allocation policy exists for IRQ lines (see **<linux/sched.h>**):

```
int request_irq(uint irq,
           void (*handler)(int, struct pt_regs *),
           ulong flags, const char *name);
void free_irq(uint irq);
```

Note again that **name** is what appears in the /proc/ files, and thus should be rather ***myhardware*** than ***mydrv***.

If you forget to register IRQ lines, your interrupt handler won't be called; if you forget to unregister, you won't be able to read /proc/interrupts. In addition, if the board continues generating irq's after your handler is unloaded, something weird may happen (I can't tell exactly, because it never happened to me, and I'm not likely to try it in order to document it here). [I *think* you get a kernel panic, but I've never managed (or tried) to make it happen, either—ED]

The last point I'd like to touch here is introduced by Linus's comment in **<linux/io.h>**: you have to *find* your hardware. If you want to make usable drivers, you have to autodetect your devices. Autodetection is vital if you want to distribute your driver to the general public, but don't call it “Plug and Play”, since that is now a trademark.

The hardware should detect both the ioports and the irq number. If the board doesn't tell which IRQ line it will use, you can go through a trial and error technique—it works great, if you do it carefully. The technique will be covered in a later installment.

When you know the irq number of your device, you should use **free_irq()** to release it before returning from **module_init()**. You can request it again when your device is actually opened. If you keep hold of the interrupt, you won't be able to multiplex hardware on it (and the i386 has too few IRQ lines to allow wasting them). Thus I run plip and my frame grabber on the same interrupt without unloading any module—I just open only one of them at a time.

Unfortunately, there exist some rare times where autodetection won't work, so you must provide a way to pass information to the driver about ports and irqs. A probe will usually fail only during system boot, when the first drivers have access to several unregistered devices, and can mistake another device for the one it looks for. Sometimes probing for a device can be “destructive” for another device, preventing its future initialization. Both these problems shouldn't happen to a module, which comes last, and thus can't request ports belonging to other devices. Nonetheless, a way to disable autodetection and force values in the driver is an important feature to implement. At least, it's easier than autodetection, and can help you in successfully loading the module before autodetection is there.

Load-time configuration will be the first topic of next issue, where the full source of **init_module()** and **cleanup_module** will be uncovered.

Additional information

The *Kernel Korner* columns of the following months will introduce further points of module-writing. Code samples can be found inside the kernel and on ftp sites near you.

In particular, what I describe is based on my personal experience with device drivers: both the **ceddrv-0.*xx*** and **cxdrv-0.*xx*** resemble the code I describe. Georg Zezschwitz and I wrote the ceddrv, which drives a lab interface (A/D, D/A, bells and whistles). The cxdrv driver is simpler, and drives a memory-mapped frame grabber. The latest versions of both drivers are available on ftp://iride.unipv.it/pub/linux for public ftp. **ceddrv** is also on tsx-11.mit.edu, while **cxdev** is on sunsite.unc.edu in apps/video.

There are quite a few books about device drivers out there, but they're often too system-specific and describe an awkward interface—Linux is easier. Generic books about Unix internals and the kernel source are the best teachers. I'd suggest to get one of the following:

- Maurice J. Bach, *The Design of the UNIX Operating System*, Prentice Hall, 1986
- Andrew S. Tanenbaum, *Operating Systems: Design and Implementation*, Prentice Hall, 1987
- Andrew S. Tanenbaum, *Modern Operating Systems*, Prentice Hall, 1992

Alessandro Rubini (rubini@foggy.systemy.it) is taking his PhD course in computer science and is breeding two small Linux boxes at home. Wild by his very nature, he loves trekking, canoeing, and riding his bike.

Archive Index Issue Table of Contents

Advanced search

Letters to the Editor

Various

Issue #23, March 1996

Readers sound off.

Keep it Separate

I'd like to make a short comment on the Product Review (December '95 pg. 34): *Caldera Network Desktop*.

I agree with Roger Scrafford “It's slick...” etc., but if Caldera could separate the commercial stuff from the free parts they would be playing more on the “team of FSF and Linux”.

I would be willing to pay for products like the full-blown CRiSP editor (Yes I love it, being a Brief user way back) or WordPerfect 6.1 with HTML.

But a student or hacker could download the Caldera Desktop for free.

As Linus Torvalds says (and I fully agree): “...the future is the desktop...”---meaning that what Caldera is doing for Linux is very important.

With this separated distribution method, maybe the Caldera Desktop could be a big success (like Netscape?).

Kindly,

O.J. -> Ole J. Utnes (Ole.J.Utnes@hive.no)

P.S. Keep up the good work with *LJ*.

They Do!

To answer your first point: Caldera does separate them, very carefully. It makes it very clear in the manual which parts are free, and which are commercial, and

makes it clear that you can install only the free parts on as many machines as you wish.

The desktop application is one of the commercial pieces they licensed from someone else, just like the font server, backup.unet, and the NetWare connectivity tools.

However, the Caldera Network Desktop is based on Red Hat Commercial Linux, and all the free pieces of CND are in Red Hat, which **is** available for FTP from many sites on the internet, including ftp.caldera.com, so your wish is granted.

That is, your wish is granted unless you specifically want them to give away their desktop application. Linus wasn't talking about a desktop metaphor like Looking Glass when he said that the future is in the desktop. He was talking about machines that sit on the desktop, as opposed to servers that live in a "glass house" and are administered by all-knowing gurus.

Index Wishes

I think the index should:

- Quietly appear in every December issue.
- Be printed on the last pages of the issue (like in science journals).

And, by the way, I think my father's TeX'ed index is better. But that's my own biased opinion. I think he e-mailed you about it just because you printed yours...

Peter Galbraith, research scientist galbraith@mixing.qc.dfo.ca Maurice-Lamontagne Institute, Department of Fisheries and Oceans Canada

Coming Right Up!

You will be glad to know that the index is scheduled to appear in every December issue, covering the previous year.

It will probably not be printed on the last pages of the issue. We've sold that advertising space, and we can't just take it back for a month.

Your father's index is different—it's more like the index to a book; ours is more like an extended table of contents. We may end up providing his index via the WWW and/or FTP.

You might also check out our index on our WWW site (http://www.ssc.com/). When we have all our articles on-line, that should be capable of a full-text search; it's currently capable of a keyword search.

More Questions

First I said: “Oh, a possibly boring issue...” from the titles on the cover. But no! You had a really good article on **find**, and I knew I'd learn something in that Linux System Administration article on adding a new disk.

One question to all of you: The **find** article uses **cpio** to copy a directory tree and the Administration article uses **tar**. I typically use:

tar cf -> ./ | (cd TARGET_DIRECTORY; tar xvf ->)

so I learned about the **->C DIRECTORY tar** option from Aileen's article. But should I be using **cpio**? What's best?

Also, a note on “Hints for splitting Linux across two disks”. I use her method of moving directories and pointing soft links to them as a last resort. Isn't it better to mount the second disk under /usr (or something) to begin with and then safely fully install Linux? Then at least everything is where is should be with a minimum of (sometimes confusing) links on the system.

Thanks for another good issue,

Peter Galbraith

It's a Matter of Taste

Most people prefer **tar** because it is a little easier to learn how to use, for most people. When **cpio** needs to create a directory that isn't explicitly included in the archive, it does it with 700 permissions, whereas **tar** uses 755. 755 is arguably more useful in most circumstances. Some people simply prefer **cpio**'s interface (see Eric's article on page 44). **find** and **cpio** are often thought of together; in fact, SVR4 find has a **->cpio** option that can be used as an action to add files to a **cpio** archive. I understand that some versions of **tar** have not properly preserved hard links, but I've just tested to make sure that GNU **tar** does preserve hard links correctly.

In general, most of the advantages of each archiver have been added to the other over time, so it's a matter of taste.

Eric Goebelbecker replied, in response to the second point:

“I would move the software over and soft link it myself for one reason: I did it, and know what I did. Come upgrade time, that knowledge becomes real valuable...”

S.u.S.E. Linux

Dear Editor,

I would like to commend the folks at S.u.S.E. in Germany for being generous enough to give something back to those from whom they have gained.

It is their practice to offer a free copy of their distribution to the authors of free software that they include. As the maintainer of GNU awk, I recently received my second distribution from them, and I felt that the Linux community ought to be aware of this company's fine practice.

Arnold Robbins arnold@gnu.ai.mit.edu

Archive Index Issue Table of Contents

Advanced search

The Desktop War

Michael K. Johnson

Issue #23, March 1996

Some people have decided that MS Windows has won the desktop war, and there will be no competition for the foreseeable future. I think that assumption is a little premature, for two reasons.

Linux users can run not only native Linux programs but also many DOS, SVR3 Unix, SVR4 Unix, and Macintosh programs by using emulation programs. It is very clear that this makes Linux a more universally useful system. There are emulators for lots of older computers as well, including Apple][, Commodore 64, and others, which are useful to some people, and certainly fun for people who like games they used to play on their old computers. What has been missing from this list so far has been real support for MS Windows programs. This is a pity, since there are more applications for MS Windows than any other computing platform on the planet. Not only are there many applications, but they are well-known, and people know how to use them.

Some people have decided that MS Windows has won the desktop war, and there will be no competition for the foreseeable future. I think that assumption is a little premature, for two reasons. The first is that despite the fact that MS Windows has **more** applications, many of them are not particularly high quality. The second, and far more important reason, is that even if you concede the **API war** to Microsoft, doesn't mean that Microsoft has won the war for the **desktop**. That is, even if all the applications out there were written only for MS Windows, MS Windows wouldn't be your only choice for an “operating system”—it's possible to emulate MS Windows. It's even possible to do a better job of it than Microsoft.

It's also important not to loose perspective; there are applications that Linux already runs, without any MS Windows emulation, and Linux is already a wild success in many important areas.

Wine

Most Linux users are aware that for about two and a half years, a team of programmers has been steadily working on a project called Wine, which is essentially a near-clone of MS Windows designed to run under the X Window System. For a long time, Wine could run only simple programs like Windows Solitaire, and when some onlookers waited for **a whole month** after Solitaire worked, and found that Microsoft Word **still** didn't work under Wine, they decided that Wine was dead and proclaimed their amazing deduction to all readers of the Linux Usenet newsgroups—though they neglected to include their skewed logic—which caused some other readers to believe them.

The Wine developers plodded carefully along, undeterred, and Wine has slowly matured, and is now approaching general usefulness. While each new release of Wine is still “for developers only”, some “real” MS Windows-based applications are starting to work. That doesn't mean that they are useful yet—there is no support yet for printing from Wine, for instance—but it does mean that the Wine developers are still making steady progress.

TWIN

Some Linux users have heard that a new company called Willows (www.willows.com) is developing a commercial programming library called TWIN, which was just released as BETA software a few weeks ago. It allows developers to write applications for MS Windows and then turn them into native applications (which maintain the MS Windows look and feel) on Unix and Linux systems, and in the future (according to Willows), Macintosh and OS/2 systems.

While TWIN is explicitly not being sold as an MS Windows emulator—“...we do not currently provide [or] support pure emulation capability for “off the shelf” products (such as MS WORD and EXCEL) as a stand-alone capability.''—they do provide the ability to run MS Windows-based binary programs that developers need, including both DLLs and EXEs. Their FAQ states: “I am currently running Microsoft Office Applications, Word, Excel and Project. These are the best ways for us to verify the library...”

Now that Willows' TWIN has been released, it is appropriate to compare it to Wine. The most important difference is that Willows says that TWIN is intended only as a developer's tool, and that the binary capabilities are intended to help developers. By contrast, Wine is intended primarily to become a tool for end-users to run their normal MS Windows applications, and the developer's library which comes with it is almost a by-product (which doesn't mean that it is low-quality or useless). Wine runs only on Intel platforms, whereas TWIN has “binary emulation” and can run on other platforms as well. TWIN can use

standard MS Windows printer drivers, whereas printing support is still missing from Wine. TWIN is commercial software, and Wine is freeware. TWIN is more complete.

Willows is also participating in an effort to make an ISO standard, non-proprietary version of the MS Windows programming API. The Application Programming Interface for Windows, or APIW, is a publicly available standard that will not only help Willows, Sun, and other companies that wish to clone the MS Windows programming interface, but will also help the Wine project in two ways:

- By providing more documentation on how the MS Windows programming interface works (Microsoft has not been helpful on this point, and has refused to take part in the APIW specification process), the Wine programmers will need less trial-and-error to determine doubtful points, making their job easier, and
- By helping standardize how MS Windows applications work, APIW has the potential to make it easier for Wine to support more applications.

No stone is being left un-turned in the effort to provide MS Windows-based applications to Linux users. The team of programmers that works on the Linux DOS box, DOSEMU, has been able to run MS Windows 3.1 in the DOSEMU DOS box under Linux. This is not particularly stable, and kernel patches are required, but for people who need to run MS Windows-based applications **right now**, don't want to re-boot to do so, and have sufficient technical understanding, it is an option.

The Desktop is Not Conceded

A few points (or rather, bullets):

- Linux is a success on the desktop, even without MS Windows application support.
- Support for MS Windows applications is important, though it isn't paramount.
- MS Windows application support is not being ignored, though it is not yet ready for “prime time” use.

Archive Index Issue Table of Contents

Advanced search

cpio

Eric Goebelbecker

Issue #23, March 1996

In this month's column, Eric moves beyond find to cover duplicating files and directory trees using the versatile cpio command. cpio uses space on tape more efficiently than tar and is an excellent alternative for creating archives on platforms that do not have the GNU utilities available. Read on for a thorough discussion of cpio and its three modes of operation: Pass-through, Create and Extract.

In an earlier article, I suggested using the following command for duplicating files and directory trees:

```
$ find . -depth | cpio -pdmv dest_dir
```

Since the focus of that tutorial was the **find** utility, I didn't discuss **cpio** in depth.

cpio can create and extract archives on diskette, tape or in files using eight different archive formats, including tar. It can also create an almost perfect duplicate of a directory tree, preserving file ownership, modes, and access times. Since cpio is designed to accept a list of files such as the output of ls or **find**, it is more suited for comprehensive backup systems than tools like conventional tar; the set of files processed can be easily controlled programmatically.

cpio also has some less obvious advantages. The default cpio format uses the space on tape much more efficiently than the conventional tar format, and it can also skip damaged sections of archives and continue during restore operations, instead of quitting completely.

GNU tar, which will be covered in an upcoming tutorial, addresses many of these issues. However, when creating archives for other platforms that do not have the GNU utilities available, cpio is an excellent alternative.

Two of the reasons why cpio is not used are readily apparent. The list of possible command line switches fills nearly half a typewritten page, and since it does not accept file names or wildcards as arguments, novices can find it pretty intimidating. But cpio can be worth the extra effort.

cpio has three modes of operation. Pass-through mode, which is what I used in the example above for duplication, create mode, which is used for creating archives, and extract mode, which is used for extracting files from archives.

Pass-Through Mode

As its name implies, in pass-through mode cpio acts as a conduit for copying lists of files from one destination to another. The ability to do this while creating subdirectories as needed and handling special files makes it a crucial tool for any system administrator to be familiar with.

For example, one common situation on a multi-user system is the need for more drive space for user directories. The administrator will need to perform the following steps: add an additional drive to the system, create one or more file systems, copy the user directories from the old file system to the new one, and then, depending upon the circumstances, change the file system mount points in order to make the transition unobtrusive.

There are three methods available for copying the users' files over to the new disk. One is to use tar to archive the files and extract them to the new area. This requires the time necessary to archive and extract the files.

Another is to use **cp**'s recursive mode to copy them directly. This mode copies only regular files and links. It also follows symbolic links, which can duplicate a lot of files when used carelessly.

Of course, few system administrators know exactly what is in their users' directories. A developer may have special files such as sockets or pipes. Any user may have files with special permissions in order to prevent unwanted access. Administrators do not have time to inspect home directories that carefully, and many users do not want them to anyway.

```
$ find . -depth | cpio --pass-through \
 --preserve-modification-times \
 --make-directories --verbose /mnt/export
```

This command causes find to output the name of every file under the present directory. (The -depth option insures that directory names are output before the names of the files in them.) cpio reads these file names in and copies them to /mnt/export.

The switches passed to cpio are:

--pass-throughOperate in pass-through mode.

--preserve-modification-timesSet the modification times of the new files to that of the old ones.

--make-directoriesCreate directories when necessary. (This option works when restoring archives, also.)

--verboseVerbose mode. This mode will produce output for all files. An alternative is the **-dot** option which only produces a **.** for each file processed. (These options work in all modes.)

The command above creates an exact duplicate of the original directory, regardless of the types of files or any special file modes that were set.

If the files are being copied to the same file system, the **--link** option can be used to hard link files when necessary.

Create Mode

Create mode creates archive files. (This is also referred to as “copy-out” mode.) cpio accepts a list of file names, just as it does in pass-through mode. But instead of creating duplicate files in another area, it creates an archive and sends it to standard output.

Since it is sent to standard output, the archive can be redirected to any device or file such as a tape, diskette, or standard file.

```
$ find -depth /export/home \
    | cpio --create > /dev/fd0
```

This creates an archive of the /export/home directory tree on the floppy drive at /dev/fd0. Of course, the /export/home area probably won't fit on one floppy, but cpio prompts for another device or file name when each floppy is filled, so it can be replaced, and the user can type the device name again. (Note that **find**'s -depth switch is still recommended to prevent possible problems when the archive is extracted.)

When it comes to creating archives, cpio has many options. One of the most important is the format of the archive.

bin(default) the binary format encodes files in a non-portable method. Therefore, it is not suited for exchanging files between Linux on a PC and Linux on other architectures such as Alpha or Power PC.

odcold (POSIX.1) portable format. This is portable across platforms, but is not suited for file systems with more than 65536 inodes, which means most of today's larger hard disks.

newcnew portable format. This is portable across platforms, and has no inherent limit on number of inodes.

crcnew portable format, with a checksum added.

tarcompatible with tar, but only supports file names up to 100 characters.

ustarnew tar format. Supports up to 255 character file names.

hpbinnon-portable format used by HP/UX.

hpodc"portable" format used by HP/UX. Stores device files differently.

The archive format is specified with the --format switch.

Out of all the formats, the crc format is probably the best, since it is portable and has an extra degree of error checking via the checksum.

A better method for creating an archive would be:

```
$ find /export/home -depth | cpio --create \
  --message="Insert next disk and type /dev/fd0 " \
  --format=crc > /dev/fd0
```

This uses the crc format for the archive and prompts the user with **Insert next disk and type /dev/fd0** as each floppy is filled. The **--message** option, which works in both create and extract mode, replaces the default message.

There are many other options available for the creation of archives, which I will cover later.

Even though GNU tar does have many of the advantages of cpio, the ability to use find to specify the files to be backed up provides much more flexibility than shell wildcards. [You can do this with tar, too, but you have to send the output of find into a file and use that file as an "include file" for tar—ED]

Extract Mode

Extract mode (also referred to as “copy-in” mode) extracts files from archives. This mode is inconsistent with the other two, since file names are specified on the command line, instead of via a list on standard input.

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

This command restores all of the files from the archive in /dev/fd0, since no file names were specified. If the archive spans more than one volume, cpio will prompt for each volume the same way it does when archives are created. The --message option can be used to override the default message, as in create mode.

cpio automatically recognizes archive formats during extraction, so it is not necessary to specify them on the command line.

The path passed to cpio by find is stored in the archive. Therefore it is important to pay attention to how find is used.

```
$ find . -depth | cpio --create > /tmp/archive
```

This creates an archive that extracts into the present working directory.

```
$ find /export/home -depth | cpio --create \
    > /tmp/archive
```

This creates an archive that will try to extract to /export/home, regardless of the circumstances. If the **-d** option is specified the directory is created if it does not already exist. (If /export/home does not exist and **-d** is omitted, the extraction will fail.)

Anything specified on the command line that is not an option is treated as a filename pattern.

```
$ cpio --extract "back" < /dev/fd0
```

This will extract files in the archive that have **back** in their name. No other files will be restored. Multiple patterns can also be specified.

```
$ cpio --extract "back" "save" < /dev/fd0
```

This will extract files with “back” or “save” in their names.

In addition to providing patterns on the command line, they can be provided as lines in a file. The file is specified with the **--pattern-file=*filename*** option. This provides a lot of flexibility in restoring files, since the actual path does not have

to be known and wildcards are not needed. Frequently restored patterns can be stored in a file.

The **--nonmatching** option is used to specify files not to extract.

It may help to see the contents of the archive before extracting anything from it.

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

The **--list** option lists the contents of the archive. The option **--numeric-uid-gid** forces the list to show user and group IDs numerically, instead of trying to resolve the names with the passwd and group files.

Instead of standard input and output the archive can be sent to (or extracted from) a file.

```
$ find /export/home -depth | cpio --create \
  --file=/vol/archive
```

This option works either for creating or extracting archives. To use a remote tape drive specify the hostname and user name before the filename. (The user must have access to the remote host without a password. This can be done by using the file .rhosts)

```
$ find /export/home -depth | cpio --create \
  --file=eric@bajor:/dev/rmt0
```

One of the key advantages of creating archives with this option is that disk files (archives not on tape or floppy) created with this option can be appended to with the **--append** option.

This command will work if eric has no password, (not recommended) or if the host that the command is run on is listed in the .rhosts file in eric's home directory.

When restoring an archive it is sometimes desirable to not alter the file modification times:

```
$ find /export/home -depth | cpio --extract \
  --preserve-modification-times --file
/vol/archive
```

The **--preserve-modification-times** option works in extract mode in addition to pass through mode.

In addition to preserving modification times, the access times for archived or copy files can be preserved so that the cpio operation does not affect the original files:

```
$ find . -depth | cpio --pass-through \
 --make-directories --preserve-modification-times \
 --reset-access-time /vol/copy
```

This will copy the current directory to /vol/copy while copying the modification times on the old files to the new and also leaving the access times on the original files untouched.

The default action for cpio, when operating in copy-in (extract) or pass-through mode, is to prompt a user for confirmation before writing over existing files, if the existing file is newer. By default, cpio will not replace the existing files. The **--unconditional** option overrides that behavior:

```
$ cpio --extract --unconditional "back" "save" \
  < /dev/fd0
```

The **--dereference** option copies the file pointed to by a symbolic link, instead of the link itself, in archive creation and pass-through mode.

The **--rename** command will prompt the user to interactively rename each file. This only works in extract mode.

When acting as a system administrator, it is sometimes useful to restore an archive or duplicate a directory and change the user or group id of the target in the process.

```
$ cpio --extract --owner=eric.staff < /dev/fd0
```

This will restore the archive on /dev/fd0 and set the owner of all the extracted files to eric and the group to staff. Only root may use this option. If the group is left out, it will not be changed unless the **.** is included, in which case the group will be set to the user's login group.

Another option related to file ownership is **--no-preserve-owner**. This is the default behavior for non-root users. Files will belong to the user copying or extracting them, instead of the original user. For root the default is to preserve ownership.

There are also advanced options related to transferring data between big-endian and little-endian architectures and for controlling I/O buffer sizes to optimize performance.

Summary

The cpio command may seem cryptic at first glance, but after you use it a few times, it will become an indispensable addition to your Linux toolkit. Especially if you are one of the many users with no tape drive and no commercial backup utility, learning cpio and swapping floppies sure beats the (non-existent) alternative after a disk crashes or you make a mistake with the rm command...

Eric Goebelbecker (eric@interramp.com) 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.

Archive Index Issue Table of Contents

Advanced search

Linux Configuration and Installation

Steve Wegener

Issue #23, March 1996

This book includes the Linux 1.2.8 kernel and the Slackware 2.3 distribution along with a host of other Linux goodies (the CD is packed).

Author: Patrick Volkerding, Kevin Reichard and Eric Johnson

Publisher: MIS: Press

ISBN: 1-55828-426-5

Price: $39.95

Reviewer: Scott Wegener

Times have changed in the Linux community since the original kernels were first made available by Linus Torvalds. My first Linux installation consisted of downloading parts of an SLS distribution at night, every night for a week, from a local BBS at 2400 baud, only to find that many of the files were corrupt and unusable. I then downloaded a Slackware distribution in a similar fashion, and two weeks after deciding that I wanted to install Linux, I finally had a usable install set. Now if only there was some documentation on installing Linux...

Thankfully, those days are gone forever. The Linux kernel and distributions have evolved at a tremendous rate, documentation now exists, more and more software has been ported to Linux, and potential Linux users now have their choice not merely between two or three distributions to install or retrieve from the net, but have many CD packages to choose from as well.

Linux: Configuration and Installation is a book/CD package which includes the Linux 1.2.8 kernel and the Slackware 2.3 distribution along with a host of other Linux goodies (the CD is packed). The book is authored by Patrick Volkerding, the maintainer of the Slackware Linux distribution, along with Kevin Reichard

and Eric Johnson, both veteran Unix gurus. Their experience shows throughout the book and the CD. Although the CD can't compare to having the full Sunsite archive, it has a **lot** of useful extras like an X-windows CD browser, a Windows bootdisk/rootdisk program, and some of the more popular Linux and X packages currently available.

The book is divided into four sections: “Linux Installation and Configuration”, “Using Linux”, “Linux Communications and Networking”, and “Linux Programming”.

The first section is a new Linux user's godsend. Although now a fairly experienced Linux user, I remember scouring the Internet for some of the information contained in these few chapters. Chapter 1 not only gives an overview of PC hardware, but more importantly, lists which hardware is supported by the Linux 1.2.*x* kernel—invaluable to a first time installer of Linux. I think most of us know at least one horror story about unsupported hardware; this chapter can help ensure a new system will run Linux or, at a minimum, resolve why it might not.

Chapter 2 gives a very thorough walk-through of a “typical” Linux installation, covering every aspect of installing as well as several typical problems encountered in the process. Unfortunately, each Linux installation can have its own unique character, so it is impossible to cover every difficulty that could possibly arise. For example, the kernel I made a bootdisk from had a problem with the caching on my CD-ROM drive; as a result, I had to do a partial installation from my DOS hard drive, then compile a 1.3 kernel in order to complete installation off the CD.

I have two minor complaints about the walk-through:

- It assumes every user will be using a swap partition rather than a swapfile. Swap-files are covered, but only much later in the book. Most readers will find this section only after doing an installation using the Chapter 1 walk-through as a “template.”
- Loadlin is mentioned **after** LILO configuration has been done in the walk-through. I have yet to get LILO to work properly with Windows 95, and mentioning the existence of Loadlin **before** someone installs LILO may save some headaches.

Chapters 3 and 4 cover X-Windows installation and configuration and, for the most part, does an excellent job. These chapters include an overview of the window managers, the different X servers needed for different video boards and, most importantly, give an in-depth explanation on configuration files and the xf86config utility. The main configuration file for X-Windows, XF86Config,

which is one of the most daunting tasks for a Linux Linux newbie to set up (a fact the book readily acknowledges) is explained virtually line for line. A minor gripe: I was surprised to find no reference to the X utility vgaset. Not everyone has a monitor whose specifications exactly match up with the given monitor list; vgaset has been invaluable in final monitor/X configuration.

Chapter 5 covers the Linux file system, Un*x/Linux commands, and other general Un*x and Linux topics useful for the Linux beginner. Shells, changing passwords, filename completion, and shell history are all covered to a degree that ensures a new Linux user coming from an MS-DOS or a Windows environment won't be lost. Printing is briefly covered as well, although no mention of Ghostscript or of typical printing problems is made. Ghostview and references to the Printing-HOWTO are made later on in the book, but a separate consolidated section on printing should have been written or skipped altogether. The best feature of the chapter is definitely the elvis/vi overview; until a new user can find better references on editors and/or gets used to Linux editors, even a few pages about vi can save a lot of frustration.

Chapter 6 covers day-to-day use of X-Windows, commonly used X programs and utilities, and is one of the best chapters in the book for novices and experienced users alike. **fvwm** and its configuration file, .fvwmrc, are covered very well, with most settings fully explained as well as X resources and several X utilities. I simply can't praise the chapters on X-Windows enough (Chapters 3, 4, and 6); X is one of the most “terrifying” things to learn under Linux and any information on it helps tremendously.

Chapter 7 is another “must read” for beginners and experienced Linux users looking to further their knowledge. Most of the traditional text processing tools are explained—Emacs, groff, TeX, texinfo/info, and sed. These sections aren't exhaustive tutorials but are more than enough to get someone started using the tools. The man page format is also discussed in a section I found to be personally useful. Tar and gzip are adequately covered, and a very good section on beginning system administration covers some typically misunderstood subjects, including scheduling commands(cron/at/batch), managing users and groups, the /etc directory and passwd file, and more. While the chapter won't make you an instant savvy sysadmin, it does a good job of explaining some sysadmin tasks and is a good place to start.

Chapters 8 and 9 are quite useful to those users new to the Internet. The chapters contain much information about Linux's communication programs (Seyon and Minicom), basics of TCP/IP and host names, and most of the standard slew of Internet programs and utilities. The basics of e-mail, telnet, FTP, and the WWW are explained in an easy to understand fashion; in short, these two chapters comprise a decent introduction to the Internet under Linux.

Chapter 10 is something that many programmers from DOS or Windows environments will appreciate—a programming overview for Linux, along with an introduction to most of the more common programming tools. There are simply too many different tools to have a single chapter cover even one of them thoroughly. Intermediate programmers may ignore the chapter, but beginners or programmers new to Linux will be thankful for it. Examples are given for Perl, gawk and Tcl, and make/imake are briefly explained, which is a nice but unexpected “bonus” to round out the book's many topics.

The CD itself contains the full Slackware 2.3 distribution, which includes the X window system (XFree3.1.1), Linux kernel 1.2.8, and the standard disk series for Slackware. There isn't too much to say about the latter; Slackware has been one of the best Linux distributions since its inception, and it is easy enough to install that most non-Linux users will have few problems, if any. The full set of Linux HOWTOs and FAQs are included, both in Windows Write format (a nice touch) and ASCII/Linux versions. A full set of precompiled kernels are also on the CD and descriptions in the book help you choose the correct kernel for your system.

There is a non-destructive partitioning program included on the CD, called FIPS, as well as full source for a lot of the distribution packages. Unfortunately, I wasn't brave enough to try FIPS—I have 1 gig of storage between two HDs and would not like to tempt Fate. Some notable programs included in either binary or source:

LessTif: An alpha Motif work-alikeSamba: Utility to connect to Windows based networksSlirp: SLIP emulation for shell accounts(source)httpd: NCSA WWW serverlemacs: Lucid Emacs(Emacs for X)AUIS: Andrew User Interface System, a group of integrated appsperl-5.001: Latest version of Perl programming language

Overall, the book is well written and contains a surprising amount of information, given the vast number of subjects that can be discussed about Linux. Most of my gripes about the book are minor, and considering the amount of information covered, it's quite understandable that not every item I wanted to see was there. While it isn't a replacement for all the Linux Documentation Project publications, it does an exceptional job of placing a huge amount of Linux information into one reference book and would be a welcome addition to any beginning to intermediate Linux user's library. The book itself is more than worth the price, and packaged with the CD, it's a combination that can't go wrong.

Scott Wegener (wegster@elwha.evergreen.edu) is 26 years old and started programming in BASIC on a TRS-80 CoCo in 1982. An ex Navy aviation

electronics technician, Scott is currently in the last year of studies toward his BS in Computer Science at the Evergreen State College in Olympia, WA.

Archive Index Issue Table of Contents

Advanced search

Building a Linux Internet Server

Phil Hughes

Issue #23, March 1996

This book offers the reader a chance to implement something which just happens to use Linux.

Authors: George Eckel and Chris Hare

Publisher: New Riders

ISBN: 1-56205-525-9

Price: $40.00

Reviewer: Phil Hughes

The first thing that caught my eye about this book is that it is about doing something with Linux. Up until this point the major focus of Linux books has been how to install it, how to network it, what commands are available and such. This book, instead, offers the reader a chance to implement something which just happens to use Linux.

Let me quickly cover what I found wrong with this book. It in no way detracts from the usability of the book, only from the number of copies that are likely to be sold. The problem is that the audience isn't identified. As I read the book I continued to wonder who would really want to buy it. Sections start off with introductory material but quickly (in a few pages) get to the nitty gritty of command options and configuration files.

Finally, when I had finished reading the book and thought about it, I realized that the book was intended for someone like me. That is, someone with lots of Unix experience but a minimum of experience setting up Internet servers. Or the Unix systems administrator who was just told by his boss that the company was about to have a Web presence and he should implement it. The problem

could easily be solved with a one-page Preface (are you listening New Riders?), and I would expect sales to jump.

That said, let's get to the content of the book. The first part covers an introduction to the Internet and talks about the ideas behind a business presence and advertising on the Internet. This part includes profiles of users, trends and a look at companies that are already on the Internet (primarily as WWW sites).

If part one convinced you, part two goes on to cover how to get connected. After a brief look at terminology and connection characteristics, the book goes on to cover the types of services you may want. Their list includes finger, telnet, e-mail, ftp, WAIS, Archie, Gopher and World Wide Web. Part two ends with information on types of connections with associated costs, legal considerations and security.

Assuming you want to go through with the mission of offering services, part three of the book goes on to describe each service and show you how to get it up and running. This amounts to about half the book (150 pages) and covers Linux installation basics and how to set up ftp, freeWAIS, Gopher and WWW services. There is also a section on ZDIST, beta software that offers enhanced functionality over freeWAIS.

While seeing a description of how to set up freeWAIS is very rare (and, if you need it, well worth the cost of the book), I expect the service most often offered is WWW. There are huge books written on just setting up a WWW site, but in less than 20 pages, this book manages to explain where to get client software, how to set up the NCSA server software and how to get started. The next chapter goes on to cover management of WWW services and includes some design information, sources of conversion utilities, log management and analysis, CGI and security.

Appendices cover available Gopher services and sources of free software on the net. Finally, a short glossary covers the common terms you would expect to be new to the Internet-neophyte.

The CD-ROM has Slackware Linux on it (version 2.0.1) plus what appears to be a browser for some operating system that uses programs with names that end in .exe and .dll. Not having that system I was not able to evaluate this. Also, as Slackware 1.0.1 is history, the CD is not the exciting part of the book.

In conclusion, if you know what you are doing with a Unix system and are considering setting up some sort of Internet server, the book is well worth the money. And if you think your company should set up such a server but you

need to convince management, there is enough in this book that they can read and understand to explain why it is a good (or not so good) idea.

Archive Index Issue Table of Contents

Advanced search

New Products

Phil Hughes

Issue #23, March 1996

WorkGroup Solutions Releases WGS Linux Pro 3.0, SSC Introduces WEBnut and more.

WorkGroup Solutions Releases WGS Linux Pro 3.0

WorkGroup Solutions has announced the release of its WGS Linux Pro 3.0, one of the most advanced 32-bit, multi-user, multitasking, multimedia network operating systems available. Version 3.0 is a selection of the “best of Linux” enhanced with WGS's confidence/stability testing and bug fixes. It also features the new “EZinstall” for the fastest, easiest installation possible, and the user's manual has been revised to be more comprehensive and easier to understand. Price: $99.00; $49.00 upgrade for current Linux users.

Contact: WorkGroup Solutions, P.O. Box 460190, Aurora, CO 80046-0190. Phone: 303-699-7470. Fax: 303-699-2793. E-mail: info@wgs.com. URL: www.wgs.com.

SSC Introduces WEBnut

Specialized Systems Consultants, Inc. (SSC) has announced the availability of its new CD-ROM, WEBnut. WEBnut includes all the software necessary to build a high-performance Web server—just add PC hardware (386 or better PC with 8MB of RAM) and an Internet connection. The CD include the Apache and NCSA servers; database and CGI facilities and examples; VRML browser, Web browser, sample Web site; HTML validation tools, graphic tools and conversion utilities; log analysis tools; and a complete copy of the Linux operating system. Also included with the WEBnut CD is one of SSC's HTML reference cards and a coupon for a free issue of *WEBsmith* magazine. Price: $13.95.

Contact: Specialized Systems Consultants, Inc., P.O. Box 55549, Seattle, WA 98155-0549. Phone: 206-782-7733. Fax: 206-782-7191. E-mail: info@linuxjournal.com. URL: http://www.ssc.com.

ISE Ships EiffelMath

Interactive Software Engineering (ISE) has announced shipment and general availability of EiffelMath, a library of reusable object-oriented components for numerical computations in finance, banking and scientific applications. EiffelMath is fully supported on a wide variety of Unix platforms, including Linux, SunOS, Solaris 2.4, HP 9000, and IBM RS6000. EiffelMath applies the power of object-oriented concepts and Eiffel to facilitate the programming of numerically-intensive computations in such areas as probability, statistics, numerical integration, linear and non-linear equations, ordinary differential equations, eigenproblems, fitting and interpolation, orthogonal factorizations, linear least squares, optimization, special functions, Fast Fourier Transforms and time series analysis.

Contact: Interactive Software Engineering, 270 Storke Road, Suite 7, Santa Barbara, CA 93117. Phone: 805-685-1006. Fax: 805-685-6869. E-mail: info@eiffel.com. URL: www.eiffel.com.

Linux Gets Common Desktop Environment

Executive Tools, Inc. has announced the upcoming release of ET Desktop, their port of the Common Desktop Environment (CDE) for Linux. CDE is a graphical user interface and application toolbox that is provided by major Unix system vendors, including DEC, IBM, HP, and Sun. It is an application framework that gives software developers the ability to create attractive, portable, machine-independent applications with ease.

Contact: Executive Tools, Inc., P.O. Box 215, Round Rock, TX 78680-0215. Phone: 800-864-5150. Fax: 512-255-9442. URL: www.etools.com.

CTAR Now Available for Linux

UniTrends Software Corporation, developer of CTAR and CTAR:NET, has announced the availability of CTAR for Linux. CTAR (an abbreviation for “compressed tar”) is the fast, fully featured, menu driven, comprehensive floppy/tape/etc. archive system used by top companies around the world. CTAR compresses files as it backs them up with average compression rates of 40-60% and up to 95% on large database files. CTAR is fully compatible with all versions of tar and offers over 100 powerful features, allowing easy transfer of entire file systems between computers and across operating system platforms. The new CTAR for Linux is fully menu driven with color support, making it ideal for Web servers and networked systems.

Contact: UniTrends Software Corporation, 1601 Oak St., Suite 201, Myrtle Beach, SC 29577. Phone: 800-648-2827. Fax: 803-626-5202. E-mail: sales@unitrends.com

Empress for Linux v6.8 Now Available

Empress Software Inc. has announced the Empress multimedia RDBMS version 6.8 for Linux. Empress for Linux is a full featured, multi-user version of Empress' multimedia RDBMS, while Personal Empress for Linux is a single user version which includes a full documentation suite and many extras all on one CD-ROM. The multi-user version of Empress for Linux include RDBMS, 4GL application generator, GUI Builder development tool, Empress Connectivity, Empress Hypermedia, and more. It also includes Empress-in-One, a point and click interface which enables even the novice user to begin development immediately. Personal Empress for Linux includes RDBMS, 4GL application generator, GUI development system, Dynamic SQL, Empress-in-One, and Empress Hypermedia. Both version of Empress include a.out and ELF binary formats. Price: Multi-user—according to number of concurrent users; Personal —$149.00 USD.

Contact: Empress Software, Inc., 3100 Steeles Ave. East, Markham, Ontario, Canada. Phone: 905-513-8888. Fax: 905-513-1668. E-mail: sales@empress.com. URL: www.empress.com.

CRiSP 4.2 Announced

Vital, Inc. has announced its new version of CRiSP, version 4.2. CRiSP is the only graphical file editor for both X and Windows platforms. The new release incorporates several major new enhancements as well as a major face-lift for its user interface. Some of the new features include complete encapsulation with HP Softbench 3.0/4.0, Centerline's CodeCenter, and Microsoft Visual C/C++ development environments; integrated support for IPC, named pipes, message passing with external 3rd party applications; dynamic screen colorization and colorized printing for 40 new languages; personalized toolbars/tool icons; new GUI mail interface; recallable named projects and keyboard macros; 64-bit compatibility across all platforms; and support for unlimited file/column sizes with dynamic memory models.

Contact: Vital Inc., 4109 Candlewyck Drive, Plano, TX 75024. Phone: 214-491-6907. Fax: 214-491-6909. E-mail: info@vital.com, orders@vital.com.

About Discussion Server v1.0b Released

AEX Software, Inc. has released a groupware product, About Discussion Server v1.0b, that works over the World Wide Web. This product provides functionality

similar to Usenet news and Lotus Notes, but at a fraction of the cost and with tighter connections to the Web. Among other features, About Discussion Server allows you to set up your own discussions over the Web, allows public and password locked secure discussions, allows a great deal of customization per discussion, allows posting of HTML including graphics and forms, has search facilities integrated into the discussions, and it runs as a CGI program. Demo versions are downloadable from AEX's Web site and include a Linux version. Price: $49.99 single-server, single discussion; $499.99 single-server, multiple discussions.

Contact: AEX Software, Inc., 1411 Lincolnshire Dr., Champaign, IL 61821. E-mail: aex@pennant.com. URL: http:www.pennant.com/aex.

Archive Index Issue Table of Contents

Advanced search

Upcoming Events

Uniforum '96

February 14-16, 1996
Moscone Center, San Francisco, CA

UniForum occurs twice a year concurrent with USELINUX, and is the largest UNIX-specific show in the world. Sponsored by Softbank Comdex, Inc., Uniforum '96 is an event for everyone from UNIX experts to professionals who are about to take their organization or department into an open environment for the first time. Features over 300 exhibitors with applications for training and implementation strategies for the Internet and WWW. Speaker include IBM's Lou Gerstner, Hewlett Packard's Lew Platt, and Jim Clark from Netscape. Dennis Ritchie and Dave Presotto from AT&T Bell Labs will give a presentation on Plan 9.

Belinda Frasier and Jonathon Gross, respectively Publisher and Editor of WEBsmith, will be at the SSC/*LJ*/WEBsmith booth to answer questions. Jon will give a presentation on “Linux and the Web- A Construction Project”, and Michael Johnson, Editor of *LJ* will give a talk. SSC products, catalogs, and copies of *LJ* and WEBsmith will be available.

Further, up-to-the-minute information and registration information is available at http://www.uniforum.org/. Registration can be completed on-line or over the phone by calling 617-449-5554. If you enter code 91 at the prompt and have a fax number ready, registration forms will be faxed to you.

DECUS (Digital Equipment Computer Users Society) Canada Symposium

February 26-March 1, 1996
Hyatt Regency Hotel
Vancouver, BC, Canada

One of the premier educational events in Canada, The 29th Annual DECUS Canada Symposium is sponsored by DECUS Partners Digital Corporation, Fonorola, Onyx Computers Inc., and Pioneer. Wes Melling, DEC Vice President-OpenVMS, is the keynote speaker. Phil Hughes will present a session on Monday, February 26 entitled “Linux: The Open System for Everyone”. The conference will include 15 full-day seminars in all; topics include “Making RAID Work”, Internet Security, TCP/IP Fundamentals and Internetworking Concepts, Shell Programming in Unix, and an Introduction to OSF DCE. “Topic Areas” for DECUS Canada are The Internet, Operating Systems, Client Server Technologies, Application Development, On the Horizon, Networking, and Hardware Technologies.

The Information Technology Exposition (I.T.EX '96) will take place on Tuesday and Wednesday, February 27th and 28th. This exhibition will feature Digital and its Business Partners showcasing the latest, most advanced business solutions for information technology professions. I.T.EX '96 will be open to the general business public from 10am to 4:30pm on the above dates.

A special feature of the Vancouver DECUS Symposium is *Cafe Internet*. The cafe will be reminiscent of classic Parisian venues, with red and white checked tablecloths, French waiters, candle light, espresso-and full Internet access. Coffee and access to terminals and workstations will be complimentary. Local dial-out access will be available. Evening events include the traditional “Magic” session, a banquet evening and LUG night.

Information and Registration materials are available by calling 416-218-2181, Monday through Friday, 9am-5pm Eastern Time. The fax number is 416-218-2111. DECUS Canada's headquarters are located at 4110 Yonge Street, Willowdale, Ontario, Canada, M2P 2C7. Materials may also be requested by sending e-mail to rulag@decus.ca or information@decus.ca. DECUS Canada's URL is http://www.decus.ca.

University of Washington Twenty-Second Annual Computer Fair

March 13-14, 1996
University of Washington, Seattle WA
Husky Union Building (HUB)

Sponsored by: Computing and Communications Departments, UW

The UW Computer Fair is the oldest and largest computer show in the Pacific Northwest. It brings together professionals from the University and the community for presentations and demonstrations of state-of-the-art computer equipment, software, and support materials. There will be 60 presentations, including one by Phil Hughes on “Linux: The Unix for PCs”, and 160 displays of computer and communications technology for business, administrative, scientific, personal and educational applications. All Fair activities are free of charge to attendees.

Exhibitors at last year's fair included Digital Equipment Corporation, DTP Micro Systems, Groupware, Greendisk, Netmanage, NW Nexus, Novell, Sun Microsystems, Washington Software Association, Sun Microsystems, and Zipperware/Maland Communications, as well as SSC, IBM, and Microsoft Corporation.

For more information contact: University of Washington, Computer Fair, Box 354842, Seattle, WA 98195-4842. Phone: 206-543-3630. Fax: 206-685-4045. E-mail: compfair@u.washington.edu.

Second Symposium on Operating Systems Design and Implementation (OSDI '96)

October 28-31, 1996
Seattle, Washington

Sponsored by the USENIX Association
Co-sponsored by ACM SIGOPS and IEEE TCOS

Cynthia Deno, the Exhibition and Publicity Coordinator for USENIX at The UNIX and Advanced Computing Systems Technical and Professional Association, writes us that there has been a change in the dates of the symposium from those given in the September 1995 issue of *LJ*.

The Program Committee is currently seeking papers describing original work concerning the design, implementation and use of modern operating systems. Besides mature work, we encourage submissions describing exceptionally promising, well-grounded speculative work, or enlightening negative results. For submission guidelines, please contact the program chairs at osdi@cs.rice.edu.

For more information about the above USENIX events contact USENIX Conference Office, 22672 Lambert Street, Suite 613, Lake Forest, CA USA 92630; phone 714-588-8649; fax 714-588-9706; e-mail conference@usenix.org. WWW http://www.usenix.org. E-mail mailserver info@usenix.org. In your message include the line “send conferences catalog”.

Archive Index Issue Table of Contents

Advanced search

Consultants Directory

> This is a collection of all the consultant listings printed in *LJ* 1996. For listings which changed during that period, we used the version most recently printed. The contact information is left as it was printed, and may be out of date.

ACAY Network Computing Pty Ltd
Australian-based consulting firm specializing in: Turnkey Internet solutions, firewall configuration and administration, Internet connectivity, installation and support for CISCO routers and Linux.

Address:
Suite 4/77 Albert Avenue, Chatswood, NSW, 2067, Australia
+61-2-411-7340, FAX: +61-2-411-7325
sales@acay.com.au
http://www.acay.com.au

Aegis Information Systems, Inc.
Specializing in: System Integration, Installation, Administration, Programming, and Networking on multiple Operating System platforms.

Address:
PO Box 730, Hicksville, New York 11802-0730
800-AEGIS-00, FAX: 800-AIS-1216
info@aegisinfosys.com
http://www.aegisinfosys.com/

American Group Workflow Automation
Certified Microsoft Professional, LanServer, Netware and UnixWare Engineer on staff. Caldera Business Partner, firewalls, pre-configured systems, world-wide travel and/or consulting. MS-Windows with Linux.

Address:
West Coast: PO Box 77551, Seattle, WA 98177-0551
206-363-0459
East Coast: 3422 Old Capitol Trail, Suite 1068, Wilmington, DE 19808-6192
302-996-3204
amergrp@amer-grp.com
http://www.amer-grp.com

Bitbybit Information Systems
Development, consulting, installation, scheduling systems, database interoperability.

Address:
Radex Complex, Kluyverweg 2A, 2629 HT Delft, The Netherlands
+31-(0)-15-2682569, FAX: +31-(0)-15-2682530
info@bitbybit-is.nl

Celestial Systems Design
General Unix consulting, Internet connectivity, Linux, and Caldera Network Desktop sales, installation and support.

Address:
60 Pine Ave W #407, Montréal, Quebec, Canada H2W 1R2
514-282-1218, FAX 514-282-1218
cdsi@consultan.com

CIBER*NET
General Unix/Linux consulting, network connectivity, support, porting and web development.

Address:
Derqui 47, 5501 Godoy Cruz, Mendoza, Argentina
22-2492
afernand@planet.losandes.com.ar

Cosmos Engineering
Linux consulting, installation and system administration. Internet connectivity and WWW programming. Netware and Windows NT integration.

Address:
213-930-2540, FAX: 213-930-1393
76244.2406@compuserv.com

Ian T. Zimmerman
Linux consulting.

Address:
PO Box 13445, Berkeley, CA 94712
510-528-0800-x19
itz@rahul.net

InfoMagic, Inc.
Technical Support; Installation & Setup; Network Configuration; Remote System Administration; Internet Connectivity.

Address:
PO Box 30370, Flagstaff, AZ 86003-0370

602-526-9852, FAX: 602-526-9573
support@infomagic.com

Insync Design
Software engineering in C/C++, project management, scientific programming, virtual teamwork.

Address:
10131 S East Torch Lake Dr, Alden MI 49612
616-331-6688, FAX: 616-331-6608
insync@ix.netcom.com

Internet Systems and Services, Inc.
Linux/Unix large system integration & design, TCP/IP network management, global routing & Internet information services.

Address:
Washington, DC-NY area,
703-222-4243
bass@silkroad.com
http://www.silkroad.com/

Kimbrell Consulting
Product/Project Manager specializing in Unix/Linux/SunOS/Solaris/AIX/ HPUX installation, management, porting/software development including: graphics adaptor device drivers, web server configuration, web page development.

Address:
321 Regatta Ct, Austin, TX 78734
kimbrell@bga.com

Linux Consulting / Lu & Lu
Linux installation, administration, programming, and networking with IBM RS/6000, HP-UX, SunOS, and Linux.

Address:
Houston, TX and Baltimore, MD
713-466-3696, FAX: 713-466-3654
fanlu@informix.com
plu@condor.cs.jhu.edu

Linux Consulting / Scott Barker
Linux installation, system administration, network administration, internet connectivity and technical support.

Address:
Calgary, AB, Canada
403-285-0696, 403-285-1399
sbarker@galileo.cuug.ab.ca

LOD Communications, Inc
Linux, SunOS, Solaris technical support/troubleshooting. System installation, configuration. Internet consulting: installation, configuration for networking hardware/software. WWW server, virtual domain configuration. Unix Security consulting.

Address:
1095 Ocala Road, Tallahassee, FL 32304
800-446-7420
support@lod.com
http://www.lod.com/

Media Consultores
Linux Intranet and Internet solutions, including Web page design and database integration.

Address:
Rua Jose Regio 176-Mindelo, 4480 Cila do Conde, Portugal
351-52-671-591, FAX: 351-52-672-431
http://www.clubenet.com/media/index.html/

Perlin & Associates
General Unix consulting, Internet connectivity, Linux installation, support, porting.

Address:
1902 N 44th St, Seattle, WA 98103
206-634-0186
davep@nanosoft.com

R.J. Matter & Associates
Barcode printing solutions for Linux/UNIX. Royalty-free C source code and binaries for Epson and HP Series II compatible printers.

Address:
PO Box 9042, Highland, IN 46322-9042
219-845-5247
71021.2654@compuserve.com

RTX Services/William Wallace
Tcl/Tk GUI development, real-time, C/C++ software development.

Address:
101 Longmeadow Dr, Coppell, TX 75109
214-462-7237
rtxserv@metronet.com
http://www.metronet.com/~rtserv/

Spano Net Solutions
Network solutions including configuration, WWW, security, remote

system administration, upkeep, planning and general Unix consulting. Reasonable rates, high quality customer service. Free estimates.

Address:
846 E Walnut #268, Grapevine, TX 76051
817-421-4649
jeff@dfw.net

Systems Enhancements Consulting
Free technical support on most Operating Systems; Linux installation; system administration, network administration, remote system administration, internet connectivity, web server configuration and integration solutions.

Address:
PO Box 298, 3128 Walton Blvd, Rochester Hills, MI 48309
810-373-7518, FAX: 818-617-9818
mlhendri@oakland.edu

tummy.com, ltd.
Linux consulting and software development.

Address:
Suite 807, 300 South 16th Street, Omaha NE 68102
402-344-4426, FAX: 402-341-7119
xvscan@tummy.com
http://www.tummy.com/

VirtuMall, Inc.
Full-service interactive and WWW Programming, Consulting, and Development firm. Develops high-end CGI Scripting, Graphic Design, and Interactive features for WWW sites of all needs.

Address:
930 Massachusetts Ave, Cambridge, MA 02139
800-862-5596, 617-497-8006, FAX: 617-492-0486
comments@virtumall.com

William F. Rousseau
Unix/Linux and TCP/IP network consulting, C/C++ programming, web pages, and CGI scripts.

Address:
San Francisco Bay Area
510-455-8008, FAX: 510-455-8008
rousseau@aimnet.com

Zei Software
Experienced senior project managers. Linux/Unix/Critical business software development; C, C++, Motif, Sybase, Internet connectivity.

Address:
2713 Route 23, Newfoundland, NJ 07435
201-208-8800, FAX: 201-208-1888
art@zei.com

Archive Index Issue Table of Contents

Advanced search