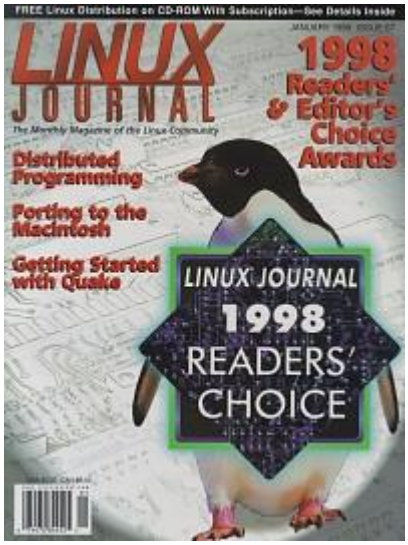
Linux Journal Issue #57/January 1999



Features

DIPC: The Linux Way of Distributed Programming *by Mohsen Sharifi and Kamran Karimi*

This article discusses the main characteristics of Distributed Inter-Process Communication (DIPC), a relatively simple system software that provides uses of the Linux operating system with both the distributed shared memory and the message passing paradigms of distributed programming.

Transform Methods and Image Compression *by Darrel Hankerson and Greg A. Harris*

An introduction to JPEG and wavelet transform techniques using Octave and Matlab.

LJ Interviews Kent McNall of Apropos *by Marjorie Richardson*

A talk with the head of a company using Informix SE for Linux in a point-of-sale application almost before it was announced.

1998 Readers' Choice Awards *by Amy Kukuk*

You voted, we counted, here are the results.

1998 Editor's Choice Awards *by Marjorie Richardson*

A look at the Editor's choices for best products of 1998 and why she chose them.

News & Articles

Introduction to LyX *by Ulrich Quill*

Make working with LaTex easier by using the WYSIWYG editor LyX.

x-automate: Control Your Home with Linux *by Stewart Benedict*
Mr. Benedict show us the way to live in the home of the future by using our computer to control lights and appliances.
A Short History of Women in Technology *by Thomas Connelly*
If you think all computer professionals are men think again. Mr. Connelly tells us about some well-known women in computer annals.
The Proper Image for Linux *by Randolph Bentson*
Dr. Bentson did a survey of Linux kernel developers to find out about their backgrounds. Here are the results.
Understanding a Context Switching Benchmark *by Randy Appleton*
A look at the Linux kernel scheduler.
An Introduction to VRML *by Tuomas Lukka*
Getting Started with Quake *by Bob Zimbinski*
First Canadian National Linux Installfest *by Dean Staff*

Reviews

VariCAD Version 6.2-0.3 *by Bradley Willson*
SciTech Display Doctor 1.0 *by James Youngman*
PartitionMagic 4.0: A Linux User's Perspective *by Roderick Smith*

Columns

Take Command Calendar Programs *by Michael Stutz*
Mr. Stutz introduces us to a digital method for keeping track of appointments and those important dates in our lives.
Linux Means Business Linux as a PACS Server for Nuclear Medicine *by Cheng-Ta Wu*
Linux is being used in a Taiwan hospital as a server for medical images and as a firewall.
System Administration Caching the Web, Part 1 *by David Guerrero*
Improve your users' browsing and save your bandwidth by using proxy servers to cache web pages.
Kernel Korner Linux for Macintosh 68K Port *by Alan Cox*
“I don't care if space aliens ate my mouse” or a case study in both the technical and human issues in porting the Linux OS to a new M68K target platform.
At the Forge Creating a Web-based BBS, Part 1 *by Reuven M. Lerner*
Ready to create your own virtual community? Here's how to begin.

Departments

Letters to the Editor
Stop the Presses *by Norman M. Jacobowitz*
1998 Atlanta Linux Showcase
New Products
Best of Technical Support

Strictly On-line

Installation and Configuration of FreeBSD *by Sean Eric Fagan*

Here's how to set up a web server using another freely available operating system, FreeBSD, a high performance, mature, UNIX-like system.

Supplement

Enterprise Solutions Supplement

Archive Index

Advanced search

LINUX JOURNAL

DIPC: The Linux Way of Distributed Programming

Mohsen Sharifi

Kamran Karimi

Issue #57, January 1999

This article discusses the main characteristics of Distributed Inter-Process Communication (DIPC), a relatively simple system software that provides users of the Linux operating system with both the distributed shared memory and the message passing paradigms of distributed programming.

Before Linux, powerful UNIX operating systems were considered a luxury. Linux made it possible for ordinary people to have access to an affordable and reliable computing platform. The only problem is that Linux was originally based on decades-old designs (see Resources 7), making it less attractive for more technically minded users. Linux's answer to this problem is either port and adaptation or introduction of newer concepts.

Building *multi-computers* (see Resources 1) and programming them are among the more popular research subjects and demand for them is rapidly rising. Any solution to distributed programming under Linux should keep up with one of Linux's more important features: *availability* to ordinary users.

Motivation

Linux already had symmetric multi-processing capabilities. However, it did not provide enough standard facilities for distributed software development. Programmers and users had to resort to different add-on packages and various programming models to write or use distributed software. The mechanisms provided by these packages usually differed greatly from one another, each requiring users to learn some new material which was not of any use to them when migrating to other methods. Many also required detailed involvement of the programmer in the process of transferring data over the network; an example is the PVM (Parallel Virtual Machine) software (see Resources 8). The

need for a simpler distributed programming model, usable by more programmers, was obvious.

What is Distributed Programming?

The DIPC Software

DIPC (Distributed Inter-Process Communication) is a software-only solution for enabling people to build and program multiple computers easily. Each node can be an ordinary personal computer. These nodes must be connected to each other by a TCP/IP (see Resources 3) network. It does not use network broadcasting, which helps it work in networks without such capabilities. Also, no assumption of a synchronized clock is made. These features mean that DIPC could be used in a wide area network (WAN).

Right from the start, it was decided that *ease of application programming* and the *simplicity of the DIPC itself* should be among the most important factors in the system design, even if it were to mean some loss in performance. This decision was backed by the observation that computing and telecommunications equipment speeds are improving rapidly, while training and programming times for distributed applications are not.

In DIPC, UNIX System V IPC mechanisms (see Resources 4), consisting of semaphores, messages and shared memories, are modified to function in a network environment. This means that installing DIPC requires changing and recompiling the kernel. Here, the same system calls used to provide communication between processes running on the same computer can be used to allow the communication of processes running on different machines. There is no new system call for the application programmer's use. There is also no library to be linked to the application code, and no need for any modifications in compilers. DIPC could be used with any language that allows access to the operating system's system calls. It is completely *camouflaged* in the kernel.

The result is that DIPC supports both the message passing and the distributed shared memory paradigms of distributed programming, providing more options for the application programmer (see Resources 5). It also allows the processes to share only selected parts of their address space in order to reduce the problems of false sharing.

It was decided to implement DIPC in the user space as much as possible, with minimal changes to the kernel. This leads to a cleaner and simpler design, but in a monolithic operating system such as Linux it has the drawback of requiring frequent copy operations between kernel and user address spaces (see

Resources 2). As UNIX does not allow user space processes to access and change kernel data structures at will, DIPC must have two parts. The more important part is a program named **dipcd**, which runs with superuser privileges; dipcd forks several processes to do its work. The other part is inside the kernel giving dipcd work and letting it see and manipulate kernel data. The two parts use a private system call to exchange data. This system call must not be used by other processes in the system.

DIPC provides easy data transfer over the network and assumes that the code to use these data already resides at the suitable places. This is justifiable when one considers that in most cases, the program's code changes much less frequently than the data.

About the Example Distributed Program

Listing 1. Example Distributed Program

DIPC is only concerned with providing *mechanisms* for distributed programming. The policies, e.g., how a program is parallelized, or where an application program's processes should run, are determined by the programmer or the end user.

DIPC Clusters

DIPC enables the creation of *clusters* of PCs. Computers in the same cluster could work together to solve a problem. DIPC's clusters are logical entities, meaning they are independent of any physical network characteristics. Computers could be added or deleted from a cluster without the need to change any of the network parameters. Several clusters may exist in the same physical network, but each computer can belong to at most one of them. Computers on the same cluster can even be connected to each other by a WAN. As far as DIPC is concerned, computers in one cluster never interact with computers in other clusters.

In normal System V IPC, processes specify numerical *keys* to gain access to the same IPC structure (see Resources 4). They can then use these structures to communicate with each other. A key normally has a unique meaning in only one computer. DIPC makes the IPC keys globally known. Here, if the application programmer wishes, a key can have the same meaning in more than one machine. Processes on different computers can communicate with each other the same way they did in a single machine.

Information about all the IPC keys in use is kept by one of dipcd's processes called the *referee*. Each cluster has only one referee. In fact, it is having the

same referee that places computers in the same cluster. All other processes in the cluster refer to this one to find out if a key is in use. The referee is DIPC's name server. Besides many other duties, the referee also makes sure that only one computer at a time will attempt to create an IPC structure with a given key value, hence the name. Using a central entity simplifies the design and implementation but can become a bottleneck in large configurations. Finding a remedy to this problem is left to the time when DIPC is actually running in such configurations.

Users may need to run some programs (e.g., utilities) in all the computers in the system at the same time, and these programs may need to use the same IPC keys. This could create interference. To prevent any unwanted interactions, distributed IPC structures are declared by programmers. The programmer must specify a flag to do this. The structures are local by default. The mentioned flag is the *only* thing the programmer should do to create a distributed program. The rest is like ordinary System V IPC programming. Other than this flag to keep DIPC compatible with older programs, the system is totally transparent to programmers.

DIPC Programs

DIPC's programming model is simple and quite similar to using ordinary System V IPC. First, a process creates and initializes the needed IPC structures. After that, other processes are started to collaborate on the job. All of them can access the same IPC structures and exchange data. These processes are usually executing in remote machines, but they could also be running on the same computer, meaning distributed programs can be written on a single machine and later run on real multi-computers.

An important point about DIPC is that no other UNIX facility is changed to work in a distributed environment. Thus, programmers cannot use system calls, such as **fork**, which create a process in the local computer.

The fact that DIPC programs use numerical keys to transfer data means they do not need to know where the corresponding IPC structures are located. DIPC makes sure that processes find the needed resources just by using the specified keys. The resources could be located in different computers during different runs of a distributed program. This logical addressing of resources makes the programs independent of any physical network characteristics.

Simple techniques allow the mapping from logical computing resources needed by a program to physical resources to be done with no need to remake the program. As DIPC programs do not need to use any physical network addresses, they do not need recompiling to run in new environments. Of course, this does not prevent the programmer from choosing to make his

program dependent on some physical system characteristics. For example, he could hard code a computer address in his code. DIPC programmers are discouraged from doing this type of coding.

When dipcd is not running, the kernel parts of DIPC are short circuited, causing the system to behave like a normal Linux operating system. As a result, users can easily disable the distributed system. Also, normal Linux kernels are not affected by DIPC programs, meaning there is no need to change and recompile these programs when they are to be executed in single computers with no DIPC support.

DIPC's Distributed Shared Memory

Distributed Shared Memory (DSM) (see Resources 6) in DIPC uses a multiple-readers/single-writer protocol. DIPC replicates the contents of the shared memory in each computer with reader processes so they can work in parallel, but there can be only one computer with processes that write to a shared memory. The *strict consistency* model is used here, meaning that a read will return the most recently written value. It also means there is no need for the programmer to do any special synchronization activity when accessing a distributed shared memory segment. The major disadvantage with this scheme is a possible loss of performance in comparison to other DSM consistency models.

DIPC can be configured to provide a segment-based or page-based DSM. In the first case, DIPC transfers the entire contents of the shared memory from computer to computer, with no regard as to whether all data will be used. This could reduce the data transfer administration time. In the page-based mode, 4KB pages are transferred as needed, making possible multiple parallel writes to different pages.

In DIPC, each computer is allowed to access the shared memory for at least a configurable time quantum. This lessens the chance of the shared memory being transferred frequently over the network, which could result in bad performance.

Error Detection in DIPC

DIPC assumes a fail-stop (see Resources 9) distributed environment, so it employs time-outs to find out about any problem. The at-most-once semantics (see Resources 1) is used here, meaning DIPC tries everything just once. In case of error, it simply informs the relevant processes, either by a system call return value or, for shared memory read/writes, by a signal. DIPC itself does not do anything to overcome the problem. The user processes should decide how to deal with the error. This is normal behavior in many other cases in UNIX .

Security in DIPC

It is important to provide some means to make sure that the data are accessed only by people with proper permissions. DIPC uses login names, not user IDs, to identify users. Remote operations are performed after assuming the identity of the person who executed the system call originally. For this to work, one login name on all computers in a DIPC cluster should denote the same person.

In order to prevent intrusion to DIPC clusters, addresses of the computers allowed to take part in a cluster should be put in a file for DIPC to consult.

Current Status of DIPC

DIPC is under development mainly in the Iran University of Science and Technology's (IUST) Department of Computer Engineering, but people from different parts of the world are currently working on it. A port to Linux for Motorola 680x0 processors has been completed. This made DIPC a heterogeneous system, as the two versions can communicate with each other. DIPC's sources and related documents can be found on the Internet via anonymous FTP at sunsite.unc.edu, in /pub/Linux/system/network/distrib/, or can be downloaded from DIPC's web page at http://wallybox.cei.net/dipc/.

Call for Cooperation

DIPC belongs to the Linux users community, and the ultimate goal is to make it an integral part of the Linux operating system. Considering the inadequacy of computing and informational facilities in IUST, the only way to make sure this software will survive is for interested people to join in its development.

To subscribe to DIPC's mailing list, send e-mail to majordomo@wallybox.cei.net with the body containing “subscribe linux-dipc”. Postings should go to linux-dipc@wallybox.cei.net.

Conclusion

DIPC is a simple distributed system that works by bringing new functionality to an IPC system designed decades ago. Many of the DIPC's nicer features are the result of its being hidden inside the kernel. Considering its main characteristics, DIPC has the potential to introduce ordinary programmers to distributed programming, thus making Linux one of the first operating systems with *usable* and *really used* distributed programming facilities.

Several experimental distributed systems are available for use. Many of them have been implemented in universities running UNIX variants on workstations produced by different manufacturers. The fact that, in most cases, researchers

did not have free access to the underlying operating system's source code has had a big influence on the design decisions. The availability of source code in Linux has provided new ways to deal with the problems of distributed programming. DIPC is an example of what can be done when one has access to the operating system sources. Some could mention the problems in porting DIPC to proprietary operating systems with no publicly available source code as a drawback. However, in our opinion, proprietary operating system vendors and their users are the ones at a loss here, as they cannot take advantage of more easy-to-use distributed systems developed by third parties. This statement does not mean DIPC could not be implemented in other UNIX variants supporting System V IPC, but implies that the port can only be attempted by people with access to kernel source code.

Resources

Acknowledgements

Mohsen Sharifi (mshar@vax.ipm.ac.ir) is a lecturer in the Computer Engineering Department of the Iran University of Science and Technology. He heads a Linux-based software engineering laboratory. His main interest is the application of object-oriented methodology to the development of distributed operating systems. He received his BS, MS and Ph.D. in Computer Science from the University of Manchester in the United Kingdom. He is a member of the British Computer Society.

Kamran Karimi (karimik@un.iust.ac.ir) lives in Tehran. He has a BS in Electrical Engineering and an MS in Computer Science, both from Iranian universities. He

is a former Amiga programmer and though he had to sell his Amiga 1200 a long time ago to finance the DIPC project, some of his Amiga programs are still used. His main research interests are artificial intelligence, operating systems and programming languages. DIPC was the subject of his master's thesis and among the first such projects in Iran.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Transform Methods and Image Compression

Greg A. Harris

Darrel Hankerson

Issue #57, January 1999

An introduction to JPEG and wavelet transform techniques using Octave and Matlab.

This article has its origins in a data compression course we've been developing over the past few years at Auburn University. The course is elementary and begins with the basic (text) compression methods of Shannon and Huffman. Some of these methods can be appreciated with pencil-and-paper examples; others, such as images to be modified by compression, need some machine experimentation.

Students may choose to present a project as part of their course evaluation. We've seen various projects, including an amusing example of Huffman-on-a-hand-calculator, an overview presentation of PNG (Portable Network Graphics) and a project concerning smoothing in JPEG.

We will introduce the transform techniques of JPEG and wavelets, discuss some mathematical themes shared by these methods, and illustrate the use of a high-level linear algebra package in understanding such schemes. The images were generated using Octave and Matlab, primarily on GNU/Linux (x86) and Solaris (SPARC), but also on a Macintosh.

Image Compression and Transforms

Data compression methods with zero information loss have been used on image data for some time. In fact, the popular GIF format uses an LZW scheme (the basic method used in UNIX *compress*) to compress 256-color images. PNG is more sophisticated and capable, using a predictor (or filter) to prepare the data for a *gzip*-style compressor. (Greg Roelofs has an introduction to PNG and some notes on patent questions concerning GIF [see Resources 8].) However,

applications using high-resolution images with thousands of colors may require more compression than can be achieved with these *lossless* methods.

Lossy schemes discard some of the data in order to obtain better compression. The problem, of course, is deciding just which information is to be compromised. Loss of information in compressing text is typically unacceptable, although simple schemes such as elimination of every vowel from English text may find application somewhere. The situation is different with images and sound; in those cases, some loss of data may be quite acceptable, even imperceptible.

In the 1980s, the Joint Photographic Experts Group (JPEG) was formed to develop standards for still-image compression. The specification includes both lossless and lossy modes, although the latter is perhaps of the most interest (and is usually what is meant by "JPEG compression"). G. K. Wallace has a paper (see Resources 10) discussing the standard in some detail.

The method in lossy JPEG depends on an important mathematical and physical theme for its compression: local approximation. The JPEG group took this idea and fine-tuned it with results gained from studies on the human visual system. The resulting scheme enjoys wide use, in part because it is an open standard but mostly because it does well on a large class of images, with fairly modest resource requirements.

JPEG and wavelet schemes fall under the general category of transform methods. The development of wavelet techniques has taken place more recently than the classical method in JPEG, and is a consequence of the never-ending search for "better" basic images.

Roughly speaking, the first step in lossy compression schemes like JPEG and wavelets is to break down an image into a weighted sequence of simpler, more basic images. At this stage, the image may be reconstructed *exactly* from knowledge of the basic images and their corresponding weights. The effectiveness of the method depends to a great extent on the choice of the basic images. Once a set of basic images, or *basis*, has been chosen, arbitrary images can be replaced by equivalent collections of weights. A basic image having a correspondingly large weight is an indication of its characteristic importance in the overall image. (The assumption here is that the basis images have been *normalized*, so that they have the same mathematical size.)

The mathematics behind this process is expressed in the language of linear algebra. There is considerable mathematical freedom in the choice of basis images; however, in practice they are usually chosen to exhibit features

intrinsic to the class of images of interest. For example, JPEG chooses basic images designed to reflect certain classical spatial frequencies.

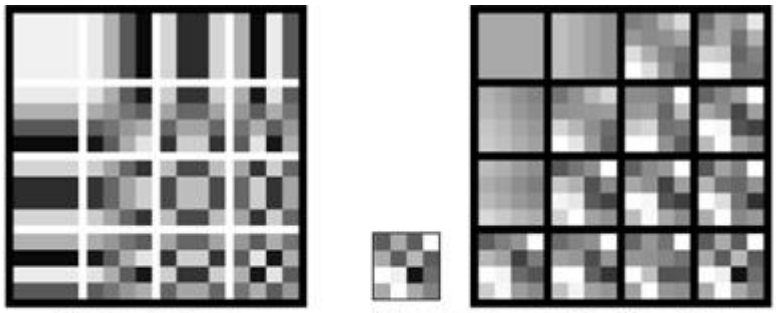
The process of using a basis to resolve an image into a collection of weights is called a *transform*. To simplify things, we'll consider gray-scale images (color is discussed briefly in the conclusion), which can be represented as **mxn** arrays of integers. The range of values isn't important in understanding the mathematical ideas, although it is common to restrict values to the interval **[0,255]**, giving a total of 256 levels of gray. As an example, Figure 3(a) shows an image containing **256x256** pixels with 145 shades of gray represented.

Mathematically, any basis for the space of **mxn** gray-scale images must contain exactly **mn** images—the number of pixels in an **mxn** image. Consequently, the transform of an **mxn** image will have **mn** weights. The weights can be conveniently arranged into an **mxn** array called the *transformed image* even though it isn't a true image at all.

The transformation process, in itself, is certainly not a compression technique (since the transformed image is the same size as the original), but it can lead to one. Suppose the basis images can be chosen so that, for a wide class of images, many of the weights turn out to be small: for a given image, set these small weights to zero and use the resulting array of modified weights to represent it. Since the transform of the image has been modified, it can be used only to approximate the original. How good is the approximation? That depends on how good the scheme is for throwing out nonzero weights, that is, on the appropriateness of the basis elements and the number of weights which can be discarded. JPEG and wavelet methods both employ this type of process and offer significant compression benefits, often with minimal impact on the quality of the reproduction. They differ in the choice of basis images, i.e., in the transform used, and subsequently in the method used to discard small weights. However, both share the idea of picking a basis that can efficiently represent an image, often using only a small number of its basic images.

The Cosine Transform and JPEG

In this section, several examples using the *cosine transform* are presented. This transform is used by JPEG, applied to **8x8** portions of an image. An **NxN** cosine transform exists for every **N**, which exchanges spatial information for frequency information. For the case **N=4**, a given **4x4** portion of an image can be written as a linear combination of the 16 basis images which appear in Figure 1(a).



(a) 4•4 basic images (b) sample image and the 16 partial sums

Figure 1: *Image elements for the 2D cosine transform (N = 4), sample image, and the 16 partial sums.*

The transform provides the coefficients in the linear combination, allowing approximations or adjustments to the original image based on frequency content. One possibility is simply to eliminate certain frequencies, obtaining a kind of partial sum approximation. The implicit assumption in JPEG, for example, is that the higher-frequency information in an image tends to be of less importance to the eye.

The images in Figure 1 can be obtained from the scripts supplied on our web site (see Resources 4) as follows. We'll use ">" to denote the prompt printed by Matlab or Octave, but this will vary by platform.

Define the test image:

```
> x = round(rand(4)*50) % 4x4 random matrix,
                        % integer entries in
[0,50]
```

This will display some (random) matrix, perhaps

```
     | 10 20 10 41 |
     | 40 30 2  12 |
     | 20 35 20 15 |
```

and we can view this "image" with the instructions:

```
> imagesc(x);      % Matlab users
> imagesc(x, 8);   % Octave users
```

Something similar to the smaller image at the lower left in Figure 1(b) will be displayed. (We chose the **4x4** example for clarity; however, the viewer in Octave may fail to display it properly. In this case, either the image can be padded before display or a larger image can be chosen.) Now ask for the matrix of partial sums (the larger image in Figure 1(b)):

```
> imagesc(psumgrid(x)); % Display the 16 partial
                        % sums
```

The partial sums are built up from the basis elements in the order shown in the zigzag sequence above. This path through Figure 1(a) is based on increasing frequency of the basis elements. Roughly speaking, the artificial image in Figure 1(b) is the worst kind as far as JPEG compression is concerned. Since it is random, it will likely have significant high-frequency terms. We can see these by performing the discrete cosine transform:

```
> Tx = dct(x, 4) % 4
                 % of x
```

For the example above, this gives the matrix

```
      | 79.25   9.47  4.75 -11.77 |
      |  6.25 -19.69 -4.25 -11.60 |
      |  5.97   8.02 12.73 -15.64 |
```

of coefficients used to build the partial sums in Figure 1 from the basis elements. The top left entry gets special recognition as the *DC coefficient*, representing the average gray level; the others are the *AC coefficients*, AC0,1 through AC3,3.

The terms in the lower right of **Tx** correspond to the high-frequency portion of the image. Notice that even in this "worst case", Figure 1 suggests that a fairly good image can be obtained without using all 16 terms.

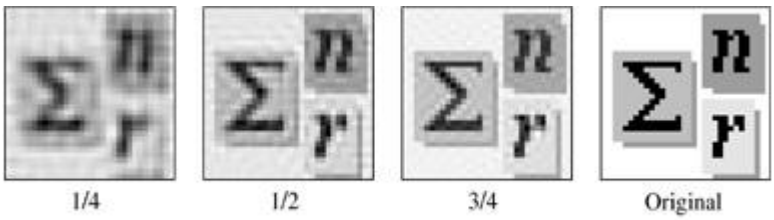


Figure 2: *Partial sums build up to the original image.*

The process of approximation by partial sums is applied to a "real" image in Figure 2, where **1/4**, **1/2** and **3/4** of the 1024 terms for a **32x32** image are displayed. These can be generated with calls of the form:

```
> x = getpgm('math4.pgm'); % Get a graymap image
> n = length(x);         % n is the number of rows
                         % in the square image
> y = psum(x, n*n / 2); % y is the partial sum
                         % using 1/2 of the terms
> imagesc(y);            % Display the result
```

Our approximations retain all of the frequency information corresponding to terms from the zigzag sequence below some selected threshold value; the remaining higher-frequency information is discarded. Although this can be considered a special case of a JPEG-like scheme, JPEG allows more sophisticated use of the frequency information.

JPEG exploits the idea of local approximation for its compression: **8x8** portions of the complete image are transformed using the cosine transform, then each block is *quantized* by a method which tends to suppress higher-frequency elements and reduce the number of bits required for each term. To “recover” the image, a dequantizing step is used, followed by an inverse transform. (We've ignored the portion of JPEG which does lossless compression on the output of the quantizer, but this doesn't affect the image quality.) The matrix operations can be diagrammed as:

```
    transform    quantize     dequantize     invert
x  ------>  Tx  ----->  QTx  -------> Ty  ---> y
```

In Octave or Matlab, the individual steps can be written:

```
> x = getpgm('bird.pgm'); % Get a graymap image
> Tx = dct(x);       % Do the 8
> QTx = quant(Tx);   % Quantize, using standard
                     % 8
> Ty = dequant(QTx); % Dequantize
> y = invdct(Ty);    % Recover the image
> imagesc(y);        % Display the image
```

To be precise, a rounding procedure should be done on the matrix **y**. In addition, we have ignored the zero-shift specified in the standard, which affects the quantized DC coefficients.

It should be emphasized that we cannot recover the image completely—there has been loss of information at the quantizing stage. It is illustrative to compare the matrices **x** and **y**, and the difference image **x-y** for this kind of experiment appears in Figure 3(f). There is considerable interest in measuring the “loss of image quality” using some function of these matrices. This is a difficult problem given the complexity of the human visual system.

The images in Figure 3 were generated at several “quality” levels, using software from the Independent JPEG Group (see Resources 5). The sizes are given in bits per pixel (bpp); i.e., the number of bits, on average, required to store each of the numbers in the matrix representation of the image. The sizes for the GIF and PNG versions are included for reference. (“Bird” is part of a proposed collection of standard images at the Waterloo BragZone [see Resources 11] and has been modified for the purposes of this article.)

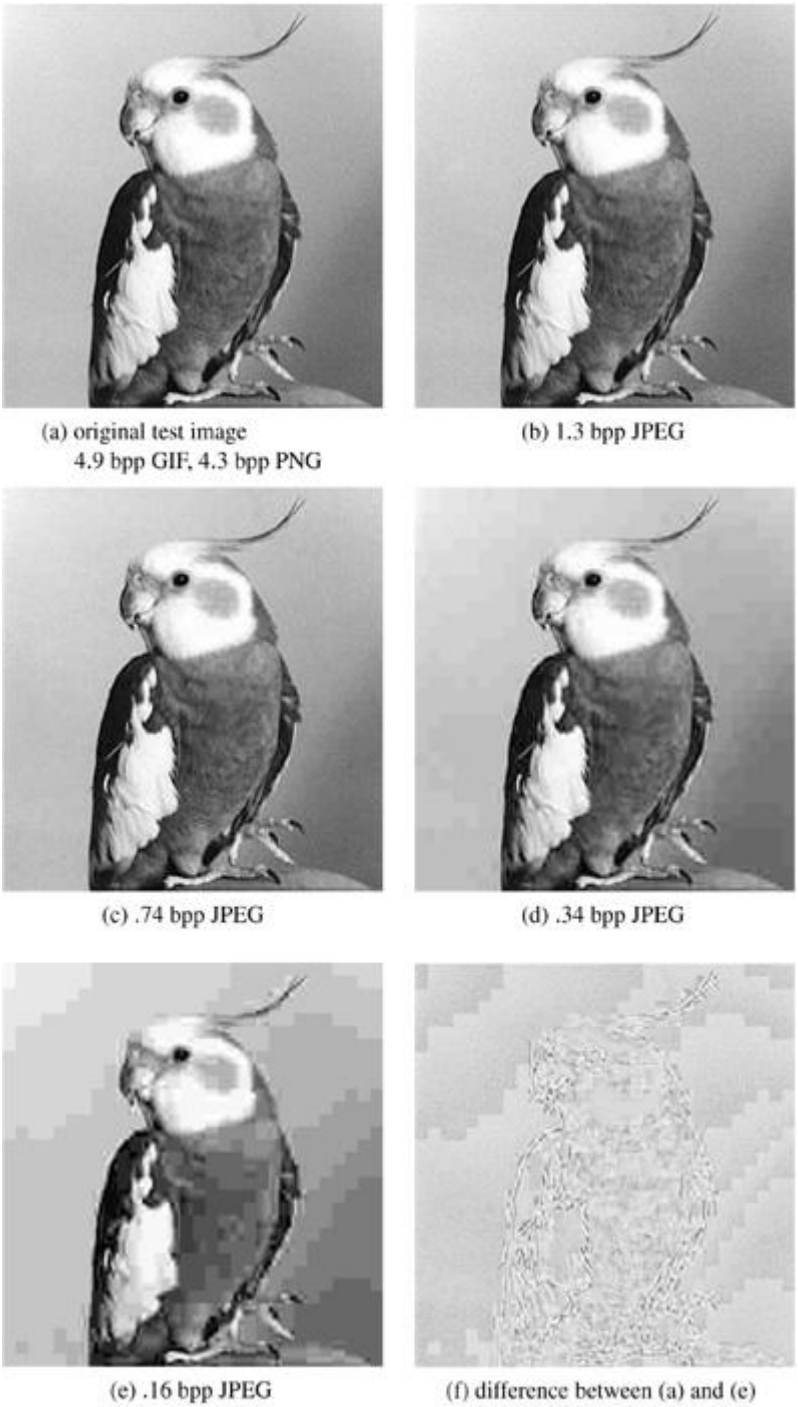


Figure 3: *GIF, PNG, and JPEG compression on 'bird'.*

A JPEG Enhancement

One troublesome aspect of JPEG-like schemes is the appearance of "blocking artifacts," the telltale discontinuities between blocks which often follow aggressive quantizing. The image on the left in Figure 6 was produced using a scalar multiple of the suggested luminance quantizer. Clearly visible blocks can be seen, especially in the "smoother" areas of the image.

JPEG operates on individual **8x8** blocks in the image and processes them independently. There can be significant loss of detail information within the

individual blocks if the quantizing is aggressive. The cosine transform used in JPEG has properties which may (indirectly) help smooth the transition between neighboring blocks; however, the tracks of the block-by-block processing can be apparent when the blocks are reassembled and the image restored. In this case, it may be desirable to implement a smoothing scheme as part of the restoration process. This section considers the back-end smoothing procedure discussed in the book *JPEG Still Image Data Compression Standard* (see Resources 7).

The JPEG decompresser may have only rough estimates about much of the original frequency information, but it typically has fairly good estimates of the average level of gray in each original **8x8** block (because of the way quantizers are chosen). The idea is to use the average gray (DC-coefficient) information of its nearest neighbors to adjust a given block's (AC-coefficient) frequency information. Figure 4 illustrates the process with a single “superblock” consisting of a center **8x8** image and its nearest neighbors. The center block in the image on the right has been “smoothed” by the influence of its nearest neighbors (the surrounding eight **8x8** blocks).

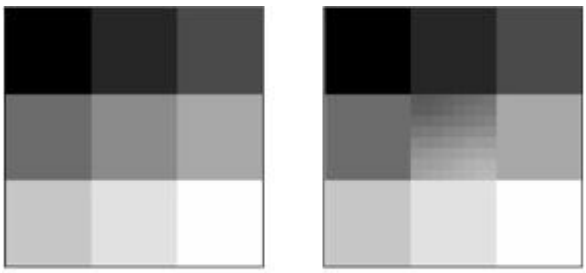


Figure 4: *Original and the “smoothed” superblock.*

The process on a more complicated image is illustrated in Figure 5. Here, the image is plotted as a surface where, at each pixel **(y,x)**, the height of the surface represents the gray value. For a given **8x8** block, the **3x3** superblock consisting of its nearest neighbors contains **3282** total entries. The polynomial

```
+ a6y2 + a7x + a8y + a9
```

is fitted by requiring that the average value over each subblock matches the average gray estimate (this gives nine equations for the unknowns **a1,...,a9**). The polynomial defines a surface over the center block, which approximates the corresponding portion of the original surface. Figure 5 shows a surface in (a) and its polynomial approximation in (b).

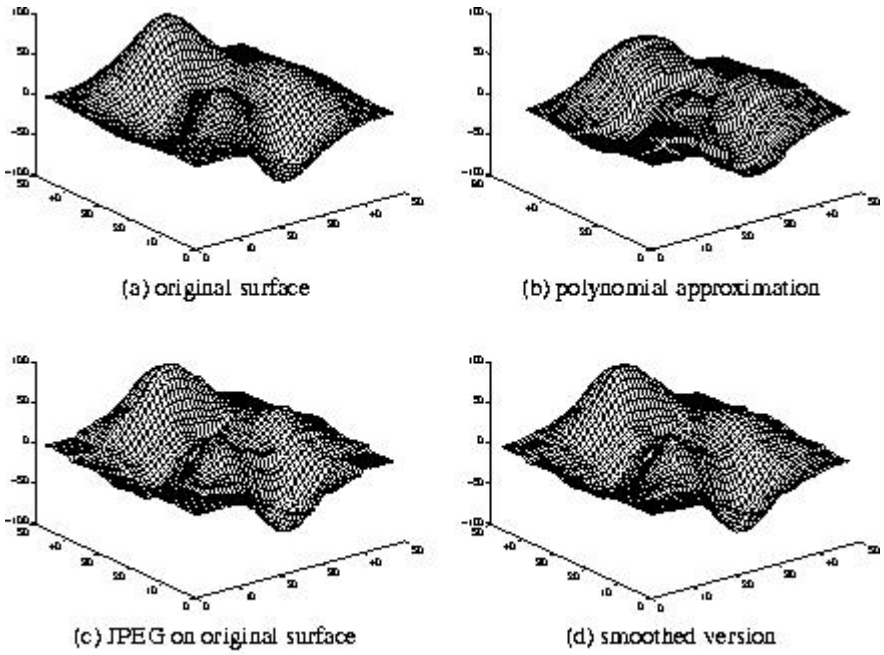


Figure 5: *The smoothing process.*

The JPEG decompresser can perform the transform procedure on a polynomial approximation, obtaining a set of predictors for the frequency information of the original image. The original estimates passed by the compressor can be adjusted using these predictors in the hope of reducing the blocking problem.

In Figure 5, the lowest five frequencies were considered for adjustment by the predictors: zero values passed by the compressor were replaced by the predicted values (subject to a certain clamping). The procedure applied to an aggressively-quantized bird image appears in Figure 6. The deblock.m script (see Resources 4) performs the smoothing. The following code was used to generate the right-hand image:

```
> x = getpgm('bird.pgm'); % Get a graymap image
> Tx = dct(x);       % Do the 8
> QTx = quant(Tx, 4*stdQ); % Quantize, using
                           % 4*luminance
> Ty = dequant(QTx);       % Dequantize
> Tz = deblock(Ty);        % Smooth
> z = invdct(Tz);          % Recover the image
> imagesc(z);              % Display the image
```

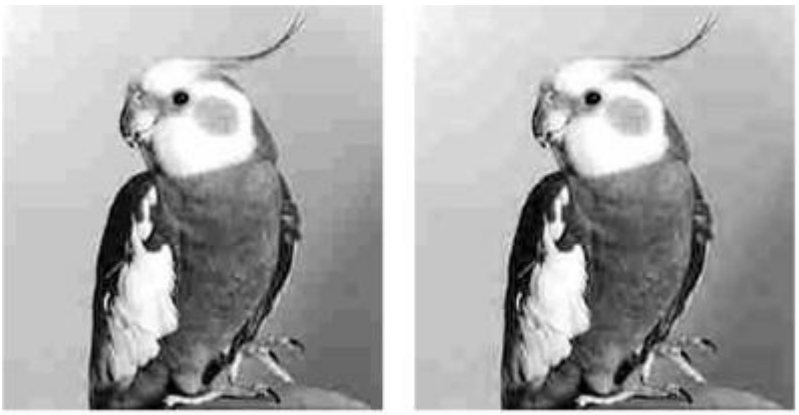


Figure 6: *'bird' with aggressive quantizing, then smoothed.*

This kind of smoothing scheme is attractive, in part because of its simplicity and the fact that it can be used as a back-end procedure to JPEG (regardless of whether the original file was compressed with this in mind). However, JPEG achieves its rather impressive compression by discarding information. The smoothing procedure sometimes makes good guesses about the missing data, but it cannot recover the original information.

A Wavelet Example

Features of a signal we wish to examine can guide us in our quest for the “right” basis vectors. For example, the cosine transform is an offspring of the Fourier transform, the development of which was, in a sense, a consequence of the search for basic frequencies with which periodic signals could be resolved.

The Fourier transform is an indispensable tool in the realm of signal analysis. When used as a compression device, we might wish it had the additional capacity of being able to highlight local frequency information—generally, it doesn't. The weights given by the Fourier expansion of a signal may yield information about the overall strength of the frequencies, but the information is global. Even if a weight is substantial, it doesn't normally give us any clue as to the location of the “time interval” over which the corresponding frequency is significant.

The interest in and use of wavelet transforms has grown appreciably in recent years since Ingrid Daubechies (see Resources 1) demonstrated the existence of continuous (and smoother) wavelets with compact support. They have found homes as theoretical devices in mathematics and physics and as practical tools applied to a myriad of areas, including the analysis of surfaces, image editing and querying and, of course, image compression.

In this section, we present an example using the *Haar* wavelet, which in one sense is the simplest of wavelets. The 16 basis elements in Figure 7 form a basis for the set of **4x4** images. Compare these with the cosine transform elements in Figure 1. One can begin to see the formation of elements with localized supports even at this “coarse” resolution level.

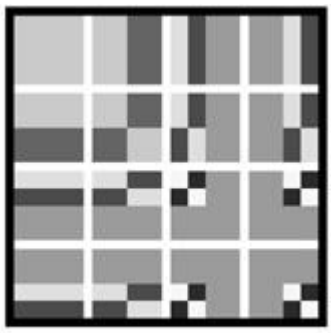


Figure 7: *The 4•4 Haar basis elements.*

The simple (lossy) compression scheme used in the example is not as elaborate as the quantizing scheme used in JPEG. Basically, we throw away any weight which is smaller than some selected threshold value. In Figure 8, we have used this simple scheme on “bird” at several tolerance settings.

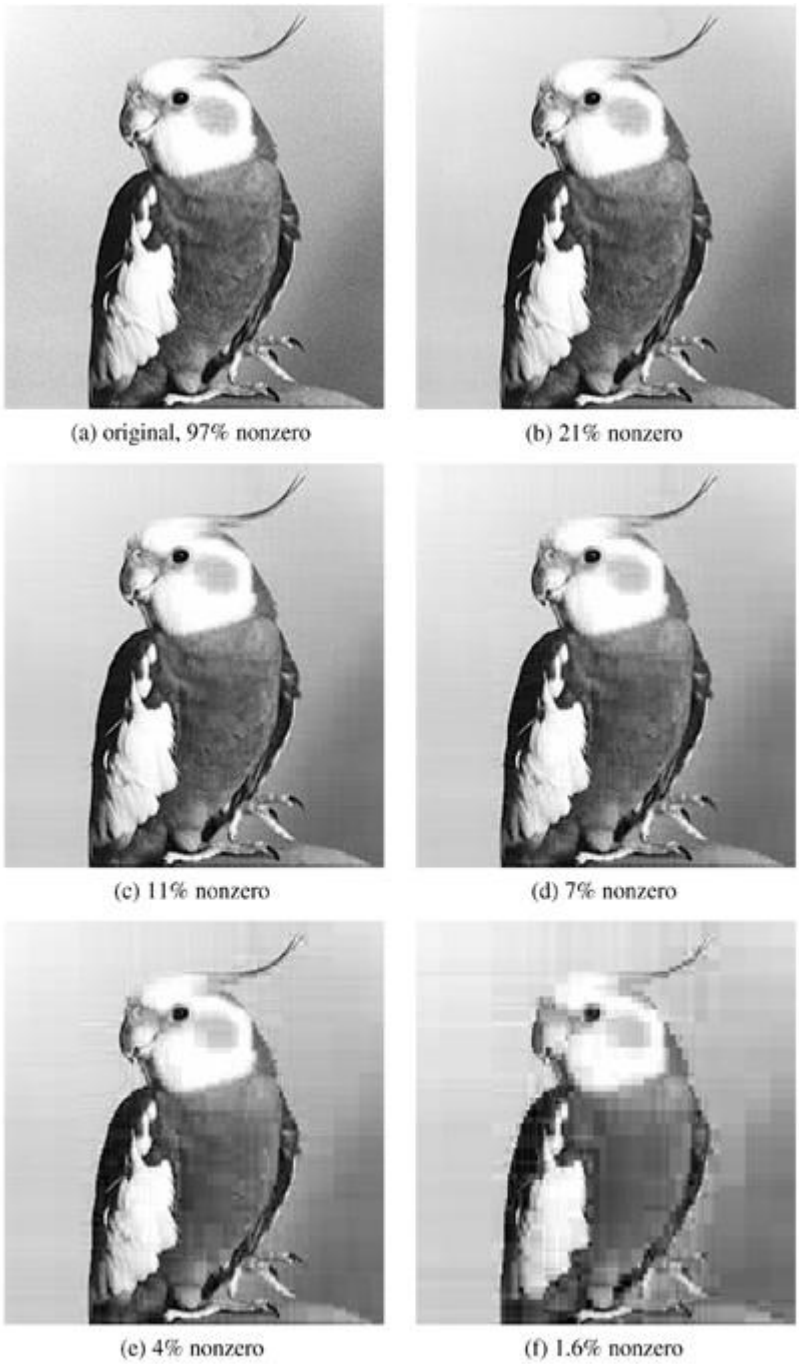


Figure 8: *'bird' (256•256) using Haar wavelet transform with simple thresh-olding. In (b)–(f), the percentage indicates the number of nonzero coef•cients in the transformed array after a threshold condition has been applied.*

Setting a weight to zero in the transformed image is equivalent to eliminating the corresponding basis array in the expansion of the image. This illustrates a certain kind of simple-minded partial sum (projection) approach to compression, similar to the example in Figure 2. Examples of more sophisticated wavelet schemes can be done with Geoff Davis' Wavelet Image

Compression Construction Kit (see Resources 2). Strang's article (see Resources 9) provides a short, elementary introduction to wavelets.

Conclusion

The discussion of JPEG and wavelets has centered on gray-scale images. Color images may assign a red, green and blue triple (rgb) to each pixel, although other choices are possible. Color specified in terms of brightness, hue and saturation, known as luminance-chrominance representations, may be desirable from a compression viewpoint, since the human visual system is more sensitive to errors in the luminance component than in chrominance (see Resources 7). Given a color representation, JPEG and wavelet schemes can be applied to each of the three planes.

This article was adapted from a recent book (see Resources 3). More information, such as details of the smoothing procedure, along with the scripts and complete documentation may be obtained from our web site (see Resources 4).

Information on Matlab (for GNU/Linux and other platforms) is available through http://www.mathworks.com/. Octave is developed by John W. Eaton with contributions from many folks, and is distributed under the GNU General Public License. Complete sources and ready-to-run executables for several platforms are available via anonymous ftp from ftp.che.wisc.edu in the /octave directory. An introduction to Octave appeared in a previous *Linux Journal* article (see Resources 6) and on-line information can be found via http://www.che.wisc.edu/octave/.

Resources

Greg A. Harris joined the faculty at Auburn University after completing a degree in mathematics at the University of Nebraska-Lincoln. Along with Darrel Hankerson and Peter D. Johnson, Jr., he is the author of Introduction to Information Theory and Data Compression, CRC Press, 1997. The photograph was taken in Zion National Park during winter 1997.

Darrel Hankerson joined the faculty at Auburn University after completing a degree in mathematics at the University of Utah. Along with Greg A. Harris and Peter D. Johnson, Jr., he is the author of Introduction to Information Theory and Data Compression, CRC Press, 1997.

Archive Index Issue Table of Contents

Advanced search

LJ Interviews Kent McNall of Apropos

Marjorie Richardson

Issue #57, January 1999

A talk with the head of a company using Informix SE for Linux in a point-of-sale application almost before it was announced.

Phil Hughes, my boss and publisher of *Linux Journal*, talked to Kent McNall at the Informix Conference held in Seattle in August. Being impressed with the fact that Apropos was already using Informix for Linux in their point-of-sale products on the day of the Informix announcement, Phil suggested I interview Kent and find out how Apropos came to be using Linux. I "talked" to Kent by e-mail on September 3.

Marjorie: Tell us a bit about yourself.

Kent: I am the President of Apropos Retail Management Systems of Lynnwood, WA. I'm 36 years old, and I've been in the computer industry since age 20, when (like Bill Gates) I dropped out of college to join the computer revolution. Since my earliest days in this industry I have gravitated towards multi-user business solutions on new-generation platforms, primarily UNIX-based. I have also gravitated towards the retail industry; the first software program I wrote was a point-of-sale system.

During my career, I've worked with almost every type of PC and PC operating system: CP/M, MP/M, many flavors of UNIX, Apple II/III/Lisa/Mac, and every generation of IBM PC (including Junior). I've worked with Novell and Windows 3/3.1/3.11/95/98/NT3.51/NT4.0.

After selling, installing and supporting systems for several companies throughout the 1980s, I formed Apropos Retail Management Systems in March of 1989 with my partners Sten Karlsen and Gary Gill. Originally chartered as a general business systems provider, we transformed the company into a software development company in 1992, developing enterprise-wide retail management solutions for small- to mid-sized chain retailers. Our systems revolve around all the most difficult tasks computers do for us today: wide area networks, data synchronization and very large databases—all with a requirement of 100% uptime from our clients.

Marjorie: What event first brought Linux to your attention?

Kent: I have many associates and acquaintances the world over who have met through the Internet. Linux is extremely popular on the Internet. A friend of mine from Poland brought it to my attention in 1996 when I was trying to bring up a web server. He told me about Linux and Apache; I was off and running.

Marjorie: What sort of evaluation procedures helped you decide to use Linux as the operating system of choice in your business?

Kent: The most important evaluation criteria, to this day, is word of mouth. A lot of people I know are using Linux in mission-critical, high-uptime environments. A high percentage of the Internet is running on Linux right now. Our own evaluation including testing, of course. We were surprised at the degree of “off the shelf” compatibility that Linux had. We expected to be very limited in the types of hardware we could use, i.e., controllers, graphics cards, etc. Because we've always been SCO users, we also needed to have compatibility with SCO binaries, which the iBCS module provides. Compatibility is the true acid test.

Marjorie: What advantages do you see in using Linux? Disadvantages?

Kent: I admit that my initial focus was on the cost advantages of Linux. The more I've worked with the product, however, the more I'm impressed by all the other advantages. Linux is truly compliant with the standards of the industry; the advantage of being so new is that there is no legacy “baggage” in the operating system. It was designed to POSIX standards from the outset. It is the fastest Intel-based UNIX I've ever seen. It is quite reliable. The support is incredible—despite the fact that the paradigm of support is very different and takes some getting used to. Most people wouldn't expect 24x7x365 support for

a free software product, but with Linux, you have it. The Linux community is incredible. The biggest advantage I see to Linux is that it will be a true alternative to Windows NT on the Intel platform in the future. I don't see any other operating system currently available that can make that claim—not OS/2, and SCO is simply off track.

The disadvantage of Linux is in the polish. Some companies are packaging Linux and doing a good job. The challenge is to keep one of those companies from becoming dominant and setting us on an SCO-type path of price inflation. One of the larger players has already tried to put a $400 price tag on a packaged Linux—BOO! The free, downloadable Linux needs more polish. It also goes without saying that the software development community must develop software compatible with Linux and the standards Linux represents.

Marjorie: What do you find most attractive about Linux?

Kent: I guess I'm a rebel—but I like the feeling that I'll have an option other than Windows NT in five years. After that, it is price—I'm a software developer and reseller. A free operating system is just the ticket for my customers.

Frankly, I think operating systems should have been free a long time ago. Linux is blazing the trail in this area, and I hope it brings price pressure on the rest of the industry. I also very much like the fact that Linux is so rock-solid and reliable—a true enterprise-class operating system.

Marjorie: How do you think Linux compares with other operating systems?

Kent: As a UNIX compared to other UNIX operating systems on Intel platforms, Linux is less expensive, faster, more open and more compliant to standards than any UNIX I know of. It is also less “polished”, as I've said. Linux has a brighter future than any other UNIX I could name. Linux has a much broader user-support base than any other UNIX, and it also has more industry support, particularly on the Internet side. Although hard numbers are not easy to come by, I think there is little doubt that far more Linux has been deployed in the past two or three years than any other version of UNIX. Remember, my main perspective is that of a “greedy” businessman—I can't give you the bit-by-bit lowdown on technical differences.

As compared to NT, don't get me started. I have found NT to be unreliable with even relatively small (50MB and less) databases. Linux is totally reliable with these databases. Linux is fast—I could easily put 30 to 40 users on a Pentium-based Linux system with a database application. That same box would die an ugly death after running NT. We rarely reboot a Linux system; we are constantly rebooting NT systems. Certainly any time a configuration change is made, NT

has to be rebooted. Almost any type of hardware can be dynamically linked to a Linux system on the fly—it is amazing how often we actually do this. Our Linux web server has been up for eight months. We rebooted our corporate NT server 12 times last week (I just looked).

I mentioned before that I'm a rebel—but in reality, every IS professional I've talked to tells this same story about NT versus UNIX or Linux servers, database servers and mission-critical applications. It is amazing how powerful Microsoft's marketing arm is.

I do have to mention the GUI interfaces on Linux; like the operating system and installation portions of Linux, they are not as polished as the MS Windows interfaces. I've seen some really nice X implementations for Linux, but it is still too hard for the average user to get there. This is a part of the polish that needs to happen with Linux.

Marjorie: What do you think needs to be added to Linux to make it more attractive to business users?

Kent: Again, we get back to the polish. Installation needs to be as easy (or easier) than NT, with more automatic sensing of installed components. Business users need to able to order their servers preconfigured with Linux. I predict that companies such as Dell and Gateway will be offering this within a year. Business users need a high level of comfort with support and service behind their operating system. Linux needs to be invisible as a server operating system—and this is certainly the case now. **Marjorie**: Did you or Informix initiate the idea of a Linux port? If it was you, how did you go about convincing Informix that they needed to port to Linux?

Kent: There had been pressure on Informix for a long time to port to Linux, and by no means do I take more than one voice's credit for getting Informix to do an actual port. Much of the pressure came from the International Informix Users Group and the Informix Users Group; it has been a major topic of conversation on the Informix newsgroup threads for a very long time. I do believe that a lot of the pressure on Informix was perceived by Informix executives as coming from the “hacker” community, which was incorrect—but let's face it, Linux has been perceived in that light for some time. However, that is changing. When I started putting pressure on Informix last year, I did turn up the heat a bit—I bent every ear I could, including Bob Finocchio's in December. I talked with them about the business case for Linux—I have the competitive edge when my operating system is free and my competitor is selling a $1200 copy of NT. When I'm trying to sell an Informix/Apropos system to a 200-store retailer, that's a lot of money. I also tried to convince them that the other major database players were not asleep and would eventually port to Linux, which

has certainly turned out to be true—but Informix has beaten the competition by months with their Linux port.

I give Informix all the credit in the world for listening to their customers. Informix is a great business partner, period. I also applaud the IIUG for their patience in working with Informix to get this port done—this is their victory. We are showing support for Informix in the only way it really counts—we've ordered our first Informix SE licenses for Linux!

Marjorie: Tell us all about your Apropos product. How does it use Informix and Linux?

Kent: Apropos is an enterprise-wide software system for chain retailers. If you were a retailer with 150 stores, you would call Apropos for a total solution, from point-of-sale to your corporate office. Our offering is very unique in that the entire system is based on a database (Informix), and written in a 4GL language (Informix Dynamic 4GL). Our product line also includes complete Data Warehousing, for which we utilize the Informix Metacube product. You can see that our partnership with Informix is truly a foundation of our business. A new product we're offering is the Apropos Retail Intranet—retailers absolutely love this part of the product. Some of our customers are Esprit de Corp of San Francisco, bebe of San Francisco, Pro Golf Discount and Intrawest, which runs such resorts as Whistler/Blackcomb, Mammoth Mountain and Mount Tremblant in Quebec.

The in-store server for a retail store has traditionally been SCO UNIX, running an Informix SE database engine. We can now install a Linux server at a fraction of the cost. A typical store will have our POS application running on Informix and will also use the Netscape Communicator browser for the Apropos Intranet. With Linux, even a single-station store can have the total reliability of a Linux-based application with the Netscape graphical interface on the same station.

Marjorie: What type of business is most likely to need Apropos?

Kent: Chain retail, strictly chain retail. Many of our clients also have e-commerce sites or mail order, but usually in conjunction with their retail operations.

Marjorie: Do you support other operating systems with Apropos?

Kent: Yes. We support SCO UNIX, HP/UX, IBM AIX, Windows NT and MS Windows desktop operating systems. Windows NT will run Informix, and we currently use it as a file/print server and for a small data warehouse. I would

point out that we have made a significant investment in Windows NT over the past four years and have two Microsoft-certified professionals and MCSE candidates on staff, including myself.

Marjorie: Have you considered making Apropos Open Source?

Kent: Our database is already open and always has been, and we freely distribute the documentation for our data model. There is not a lot of demand from our client base for our software to be open source.

Marjorie: Have all of your products been ported to Linux? If not, why not?

Kent: All of our products with the exception of the data warehouse have been ported to Linux. The reason we haven't ported the data warehouse is that Metacube requires the Informix Online Dynamic Server, which hasn't been ported yet (hint hint).

Marjorie: What advice would you give others who would like to convince companies to port their products to Linux?

Kent: Make a case for Linux in a way that business people can understand—dollars and cents. As the case for Linux builds in the systems community, it will be easier to gather information that shows the demand for Linux and the growing user base out there. I believe it is self-evident that the world does not want to be stuck with one operating system—and it is equally self-evident that Linux will be a player in the future. Many software companies and their executives will recognize this fact and respond to it by porting their products and developing new products that adhere to international open standards. Linux makes business sense. Sharp business people will see this.

Marjorie: What do you see in the future for Apropos/Linux/Informix?

Kent: At Apropos, we feel we are perfectly positioned for the future. Our software applications are totally Y2K-compliant; we are built on open systems standards from start to finish. We believe in our technology and our technology partners, particularly Informix and the Linux community. We are very close to our customers, uniquely so—and our customers are very successful companies. We will continue to build the best retail software in the industry on the best platforms. We'll continue our commitment to customer service. If you have good products and are committed to customer service, you are not going to go wrong. That's why Informix has come through a tough year very well, and it is why Apropos continues to be successful.

I see a very bright future for Linux. I think the train is just starting to pick up steam. As Microsoft continues to have problems because of their unfair

business practices, people will start to notice something strange about the old emperor, at least in the area of enterprise-class operating systems. They'll want alternatives. System OEM's will want to offer alternatives to their customers. Software developers will continue to catch the Linux wave. I personally find it very exciting, and I'm having a lot of fun watching it unfold. The fact that I can save my customers money and make more money for my business at the same time is great.

Marjorie: Thank you for your time.

Hot off the Presses

Archive Index Issue Table of Contents

Advanced search

1998 Readers' Choice Awards

Amy Kukuk

Issue #57, January 1999

You voted, we counted, here are the results.

This year, the number of Readers' Choice awards expands once more to include 28 categories of Linux favorites. The voting took place on the *Linux Journal* web site for two months. Over 3000 votes were cast, with over 50% of them from Germany. This is a small number of votes considering the number of subscribers we have. Next year, exercise your right to vote and help determine the winners.

Each year it is quite a surprise to scan through the results. Go ahead, see for yourself!

Favorite Audio Application

Winner: **RealAudio**

Runner Up: Soundstudio

Favorite Backup Utility

Winner: **BRU**

Runner Up: CTar

Most Used Linux Book

Winner: *Linux Network Administrator's Guide* by Olaf Kirch and Andy Oram

Runner Up: *Running Linux* by Matt Welsh

Best Browser of 1998

Winner: **Netscape**

Runner Up: Lynx

Most Used Business Application

Winner: **StarOffice**

Runner Up: Applixware

Favorite *LJ* Column

Winner: **Kernel Korner**

Runner Up: Best of Technical Support

Primary Communications Board

Winner: **Cyclades**

Runner Up: Digi

Most Used Database

Winner: **MySQL**

Runner Up: PostgreSQL

Best Development Tool

Winner: **GCC**

Runner Up: XEmacs

Favorite Linux Distribution

Winner: **S.u.S.E.**

Runner Up: Red Hat

Favorite Editor

Winner: **vi**

Runner Up: Emacs

Favorite File Manager

Winner: **Midnight Commander**

Runner Up: xfm

Most Played Linux Game:

Winner: **Quake**

Runner Up: XTetris

Best Graphics Application

Winner: **GIMP**

Runner Up: xv

Favorite Programming Language

Winner: **Perl**

Runner Up: Tcl/Tk

Most Loved Mailer

Winner: **Netscape**

Runner Up: Pine

Favorite Peripheral

Winner: **Ethernet**

Runner Up: ISDN

Best System Vendor

Winner: **VA Research**

Runner Up: Linux Hardware Solutions

Favorite Platform:

Winner: **Intel**

Runner Up: AMD

Most Used Portable

Winner: **Toshiba**

Runner Up: IBM

Favorite Security System

Winner: **PGP**

Runner Up: sshd

Favorite Shell

Winner: **Bash**

Runner Up: tcsh

Most Loved Special Purpose Tool

Winner: **PalmPilot**

Runner Up: plan

Best UPS

Winner: **APC**

Runner Up: Best Power

Favorite Video Tool

Winner: **Xanim**

Runner Up: RealPlayer

Best Linux Web Page

Winner: **slashdot.org**

Runner Up: linux.org

Favorite Window Manager

Winner: **KDE**

Runner Up: Fvwm

Most Used X Server

Winner: **XFree86**

Runner Up: Accelerated X

More information about the winning products and programs and other Linux hardware and software is available on our Linux Resources web site at http://www.linuxresources.com/.

Planning for the 1999 Readers' Choice Awards is already underway. If you have ideas for new categories, questions or comments, please e-mail info@linuxjournal.com. Voting for next year's awards will be held in August and September on the *Linux Journal* web site, http://www.linuxjournal.com/.

Archive Index Issue Table of Contents

Advanced search

Editor's Choice Awards

Marjorie Richardson

Issue #57, January 1999

A look at the Editor's choices for best products of 1998 and why she chose them.

When the *LJ* staff decided to have Editor's Choice Awards this year in addition to the Readers' Choice, I agreed without truly realizing how difficult it would be to make decisions. So many fine products that support Linux are available today, and the number grows daily. This has indeed been a good year for Linux users, beginning with the announcement that Netscape would become open source and proceeding through the announcements of support for Linux by all the major database companies.

Product of the Year—Netscape Communicator

I must admit this one wasn't a hard decision. It is my belief that Netscape's announcement that Communicator would be open source started it all. This announcement galvanized the world to find out about the Open Source movement and the Linux operating system that was responsible for its creation. Linux needed a big company in its corner in order for the word to spread, and Netscape provided just the initiative that was needed.

Most Promising Software Newcomers—GNOME and KDE

This was probably the most difficult decision, so it ended in a tie. So many new products are available for Linux this year; finally, the flood of software applications we have all been waiting for is happening. However, the one thing everyone has always said Linux needs to become competitive with the commercial operating systems is a user-friendly desktop—both GNOME and KDE are filling this need.

Best New Gadget—Schlumberger Smart Card

While I was given some interesting suggestions for this one, I never had any doubt that the Smart Card was the proper choice. A credit card with a Linux CPU on it is just too extraordinary. The computer chip embedded in the card stores not only mundane information about the card holder, but also biometric information that can be used for identification—talk about great security! The suggestion most people gave me was the PalmPilot, which is indeed a cool product, but even though Linux runs on it, the port was done by programmers outside 3Com. According to Mr. Bob Ingols, a 3Com staff member, 3Com does not support Linux and does not plan to.

Best New Hardware—Corel NetWinder

Corel Computer was the first company to declare Linux as its operating system of choice and sell computers with Linux pre-installed. With the continuing growth of Internet popularity, the network computer's day has come and the NetWinder is one of the best. It is small, powerful and easily configured. Best of all, it comes with Linux. Debian's recent port to the ARM architecture means that it too will run on the NetWinder. A close second was the Cobalt Qube Microserver—not only is it a great little server, it's cute too.

INFORMIX

Best New Application—Informix

Another tough one. My initial choice was the GIMP, but it's been around for some time (my first thoughts always seem to be free software). At any rate, a port of a major database to Linux has long been anticipated, and Informix made the breakthrough with other database companies following suit. With support from Informix, Linux can now enter the business “big leagues”. A close second, in my mind, is Corel's WordPerfect 8 for Linux for the same reason—to be accepted in the workplace, Linux needs this product.

Best New Book—*Samba: Integrating UNIX and Windows*

Some might call “foul” on this one, because it is published by SSC. However, this award is for the book and the author, John Blair, not for the publisher. *Samba: Integrating UNIX and Windows* was needed and its popularity has proved it. John has written a comprehensive book of interest to all who are running multi-OS shops. The book has been endorsed by the Samba Team, who has gone so far as to make John a member. If the award had been for “best all-around book on Linux”, I would have given it to the ever-popular (with good reason) *Running Linux* by Matt Welsh, published by O'Reilly & Associates.

Best Business Solution—Linux Print System at Cisco

In our October issue, we had a great article called “Linux Print System at Cisco Systems, Inc.” by Damian Ivereigh. In it, Damian described how Cisco was using Linux, Samba and Netatalk to manage approximately 1,600 printers worldwide in mission-critical environments. He also described how he did it and supplied the source code he used, so that others could also benefit from this solution—a wonderful way to contribute to the Linux community.

Most Desired Port—QuarkXPress

Linux Journal uses Linux as its operating system of choice on all but one lone machine. For layout, we must have an MS Windows 95 machine in order to run QuarkXPress. Each month we hold our breath during the layout period hoping that when Windows crashes (it always does), it won't be at a critical juncture. Crashing for no apparent reason creates extra work for Lydia Kinata, our layout artist, and much stress for all of us each month. We are more than ready to be rid of this albatross and have a total Linux shop. Next, like everyone else, we'd like Adobe to port *all* its products to Linux.

Marjorie Richardson is Managing Editor of *Linux Journal* and of the e-zine *Linux Gazette*. She had been a scientific applications programmer in the oil industry for 20 years before coming to SSC. She likes to quilt, read science fiction, watch action movies and musicals, go to the opera and camp with her husband, Riley. She can be reached via e-mail at info@linuxjournal.com.com.

Archive Index Issue Table of Contents

Advanced search

Introduction to LyX

Ulrich Quill

Issue #57, January 1999

Make working with LaTex easier by using the WYSIWYG editor LyX.

Although (or perhaps because) it is one of the most flexible typesetting tools, many people are a bit afraid of using LaTeX. They'd rather use a standard word processor. On the other hand, even for those of us quite accustomed to LaTeX, some kind of WYSIWYG editor would often come in handy, especially for shorter texts like letters.

Out on the Internet is a nice tool which may satisfy the needs of both. It is called LyX and its home page is http://la1ad.uio.no/lyx/. The primary download site is ftp://ftp.via.ecp.fr/pub/lyx/, but you can also simply follow the links from the home page. Initially it was written by Matthias Ettrich (ettrich@kde.org), one of the initiators of the KDE project, but it is now maintained by Lars Gullik Bjoennes (larsbj@ifi.uio.no). The most recent package is usually called lyx-current.tar.gz. For those of you running KDE, it may be interesting to know that Matthias Ettrich has just released a KDE version of LyX: KLyX (http://www-pu.informatik.uni-tuebingen.de/users/ettrich/klyx/klyx.html). From the screenshots it looks very promising, so it may be the version of choice if you have at least the basic KDE libraries installed.

For the installation of the source package, you will also need two libraries, which probably won't be included in your system. One is the Xpm library; the other is XForms, a library for simple XView GUI programming. You will need at least version 4.7 of libXpm and 0.81 of libforms. Both of them can already be obtained as Linux (ELF, a.out) binaries. You will find XForms at ftp://laue.phys.uwm.edu/pub/XFORMS/, ftp://ftp.via.ecp.fr/pub/xforms/ and ftp://ftp.cs.ruu.nl/pub/xforms/ and Xpm at Sunsite and its mirrors in libs/X/.

Installation

I will focus only on the installation of the binary distribution. It is available for several flavours of UNIX (even for OS/2) and is the easiest to install. Should you have any problems with the binaries, you can still switch back to compiling the sources (remembering to check for the other two libraries).

After you have downloaded lyx-current.tar.gz, log in as root and change to /usr/local. Now do:

```
tar xzf /
cd share/lyx
./configure
```

The configure program checks your local LaTeX installation and sets up the LyX configuration accordingly. Later on, you can view a file (from the Help/Documentation menu) that lists both the packages installed and not installed in your system.

Of course, you do not need to put the LyX package in /usr/local. Nevertheless, LyX can display its messages in languages other than English. For this feature, it expects to find the internationalization files in /usr/local/share/locale. If you install LyX in a different place, you will have to set the environment variable LYX_LOCALEDIR pointing to the appropriate share/locale directory. If you chose to install LyX in /opt, then for bash you would need:

```
LYX_LOCALEDIR=/opt/share/locale
```

and for tcsh:

```
setenv LYX_LOCALEDIR /opt/share/locale
```

If you don't need the internationalization, i.e., English-only messages are accessible, you can skip this step.

Running LyX

Each user can have his/her own resource file in which customizations can be made. Start by copying the file /usr/local/lyx/system.lyxrc to $HOME/.lyxrc. Now start LyX by typing **lyx**.

Figure 1. Welcome Screen

You will be welcomed by a window like the one shown in Figure 1. Looking at the LyX window, you will be reminded of any standard word processor. That's the main reason I think users who have been reluctant to use LaTeX will be encouraged by LyX to make the transition.

At the top is a menu bar with the usual features. Beneath is a button bar with a list box for choosing the desired style and several command shortcuts. (By the way, this button bar can be customized.) The window is dominated by a large text input area with a scrollbar. Finally, at the bottom is a status line.

If you do not wish to experiment on your own but would rather read a little introduction, choose the menu Help/Documentation. A file selector will open, letting you choose one of the package's documentation files. The file extension (*.lyx) shows that LyX uses its own format. However, each document can also be saved in LaTeX format.

Now back to the documentation. As the program is still in version 0.12, most of the files are not too detailed, but for a quick start I'd recommend either Main.lyx or Tutorial.lyx. In this selector you will also find the already mentioned results of the LaTeX configuration check (LaTeXConfig.lyx).

Let's suppose you selected the file Main.lyx. After confirming the message that this file is read-only, it takes a little while to load all the fonts into the X-server. At this point, error messages may appear in your console/shell window. Don't worry; it just means you have not installed all the X-server fonts LyX wants to use. Then, you are presented with one of the really nice features: a table of contents (TOC). A TOC is well-known in LaTeX, but LyX has the ability to display it in a completely interactive window, i.e., you can click with your mouse on any entry and the cursor jumps to the appropriate section (see Figure 2). If you prefer to read instructions on paper, print out the document. Select File/Print, check that everything in the opening dialog box is set to your needs and click on **OK** (or press **enter**).

Figure 2. LyX Table of Contents

Now let's start to write our first LyX document. Our goal will be a short article with a title, three sections, some mathematics stuff, perhaps a picture, some footnotes and a TOC.

First, if you opened Main.lyx before, close the TOC and select File/Close. If LyX asks you about saving any changes, say no. Now create a new document by selecting File/New. If you are not in your home directory, change to it by pressing the Home button. Now, type in a file name, e.g., MyFirstTest.lyx. After clicking on **OK**, you are presented with a blank page.

Very well. You (and LyX) are now ready to start typing. First, we create the title. Click on the down arrow beside “Standard”. A list box will open in which you can choose one of several styles. (If you know LaTeX, you will recognize most of them.) Move down and click on “Title”. The layout on the screen will change a

bit and you can type in your title, “My First LyX Document”, then press **enter**. Note that the style changes back to “Standard”. This is a characteristic of LyX. One style is valid within only one paragraph (ended by **enter**). If you want to type in more lines, press Newline (**ctrl-enter**).

To begin the first section, select the “Section” style in the list box to type in the section name (e.g., “The First Section”) and press **enter**. To add an itemize style, type in some text (“A simple itemize style:” **enter**), and select “Itemize” style—an asterisk will appear. Now you can write some items which are separated by **enter**. This is an exception to the mentioned rule: to get back to “Standard” style, you must select this explicitly. You can play around a little with the two font styles, emphasize and noun, by clicking on the button with the question mark and the person. The next section will contain some math. After adding a new section headline, click on the formula button and open the math panel (Math/Math Panel, MP). First, enter a three-dimensional unity matrix: type **E =**, click on the parentheses button in MP and click on **OK**. Now add the matrix by clicking on the matrix button in MP (the one with nine squares), set the size (3 by 3) and click on **OK**. This fills in the nine entries. For a unity matrix, the three rows should read 1 0 0, 0 1 0 and 0 0 1. If you wish, you can play around a little with the numerous buttons within the MP.

We will add a third section and insert a PostScript picture into our document. First, select Insert/Floats/Figure Float and type in any figure caption. After that (without clicking on **enter**), click on the “Insert Figure” button (just right of the formula button). In the dialog, select the first item (encapsulated PostScript). If you left-click on the appearing frame, a settings box opens. Click on “Browse” and select the file /usr/local/lyx/clipart/platypus.eps (or change the path according to your installation). Set the width to 3 inches and click on **OK**. The platypus should now appear on the screen. You will notice a red frame around the complete figure, with a small grey rectangle to the left reading “fig”. If you press the left button in this field, the whole figure will collapse into a small, red “fig”. Left-clicking on this brings back the whole frame. Pressing the left button anywhere within the figure opens the “Floats” context menu, giving you a choice of actions to perform.

Figure 3. Completed Text Document

By this time, your document should look like Figure 3. Congratulations—you've just finished your first LyX document. Now save it to disk (File/Save), if you have not already done so. If you like, print it out (File/Print). This may take a little time. As you know, LaTeX is still working behind the scenes, which means that the standard LaTeX procedure of compiling and printing has to be gone through.

If you followed the opening of the documentation file, you have already seen LyX's feature of displaying the table of contents (TOC) of a document in a separate window, letting you jump to each entry by using the mouse. How is this feature set up and invoked? In fact, you do not have to do anything special. LyX keeps track of the “Section” style as you assign it to parts of the text. Choose Edit/Table of Contents, and there it is. You will find all the sections you created; if not, just press the update button. Clicking on each entry takes your text cursor directly to the specified point.

More LaTeX Commands and Previewing

With LyX, you can do most any formatting that is possible in LaTeX. Experienced LaTeX users will notice some LaTeX commands (e.g., **\pagebreak**) are missing. This is not really a drawback. You can always assign the TeX style to any portion of text, either by choosing the style from the list box or by pressing the TeX button. This tells LyX that the marked portion of text is to be taken as native LaTeX code, thereby allowing you to use even those LaTeX commands which cannot be reached by menu entry. Even hard-core LaTeX hackers can still be satisfied by saving their document in the usual LaTeX format by selecting File/ Make LaTeX file. Although I admit that this file will not necessarily look the way it would if you had written the text directly in LaTeX, it opens up the possibility of hacking in anything you like and then running LaTeX directly on the file.

Although LyX is close to WYSIWYG, you might often still like a preview of your document. For this purpose, LyX offers both **xdvi** and **ghostview**. If both these programs are installed on your system, you can get a preview by selecting one of these two options from the file menu. The necessary background LaTeX commands (running LaTeX and, for the PostScript output, **dvips**) are done by LyX automatically, so you don't have to worry about whether all your files are up to date.

If you save text, the file is stored in the working directory you specify. But all the intermediate LaTeX-related output files (like .log, .dvi, .ps) are stored in a /tmp directory (the actual path can be specified in ~/.lyxrc), unless you explicitly instruct LyX to make a LaTeX file from your text.

Now that you have finished your first LyX (and perhaps your first LaTeX) document, it is time to drop a few notes on the customizations possible within LyX.

Templates

The first issue (which, in fact, is not really a customization of the application) is the template, which you may have noticed in the File menu (New from template...). You may already know the template notion from standard word

processors, and in LyX it works just the same: a standard template document defining the basic settings like fonts, layout, etc. A prototype of such a template may be a letter (some letter templates are included in the LyX distribution), where you would set up basic items like your personal address, a standard opening and closing phrase and the layout. To create such a template, you don't have to do anything special—just start editing a new document with the desired settings. For those parts of the text which are not standard and are to be changed in each document using this template (an example in the letter case may be the recipient's address), you can type in any text, e.g., “recipient's name”. Look at the distribution's templates for other examples. After you finish, save this template to disk like any other LyX document.

Unlike other word processors, LyX does not use a special format for templates. Any LyX document can be taken to be a template and vice versa. As a template is also a “normal” document, saving a newly created template to disk also saves all the layout options currently selected. So, if you have created a letter template using a letter-sized sheet of paper, this page size is also saved to disk, as is all the font information, etc. If you do not want each of your later letters to use all these settings, you have two options:

1. Leave the template file untouched. In this case, each time you create a new document using this template, you have to reset the settings to your specific needs after selecting the template. Remember, selecting a template is more or less like loading in a file; thus, all the settings saved in this file overwrite the current parameters.
2. Change the template file by hand using any plain text editor. In this case, you can remove all the layout commands which you do not wish to be set by the template.

The more general you wish your template to be, the more likely you will choose option 2. On the other hand, this requires some knowledge and understanding of the LyX command language. It won't be too hard if you already know LaTeX, but for the first experiments you should perhaps leave the file untouched.

Customization

The last issue to be mentioned here is the ~/.lyx directory which is the place where real customization can take place. Again, leave the files as they are until you have become a bit more acquainted with LyX. In this directory, all your personal configuration files are stored. As long as your configuration and the global system's are the same, nothing is stored here.

You can, for example, store your preferred key-bindings. There are two standard bindings (PC-modern and Emacs), but you can also define your own key-binding scheme and tell LyX the file where it is stored. You can also define

your own tool bar, various (LaTeX-related) commands, printing defaults, file defaults and the like. It would take quite a while to discuss all these options in detail, but I recommend taking a look at the file. Fortunately, all options are explained well, so it is easy to figure out where to do what.

Summary

LyX, a WYSIWYG editor for LaTeX, is by no means a standard word processor. It heavily depends on LaTeX, i.e., you need to have LaTeX installed properly, and you can only do formatting that is also possible with LaTeX. It has the distinct advantage of directly displaying all the different style elements on the screen without fiddling around with “strange” formatting commands. Those who are not familiar with LaTeX will find it much easier to take the first step toward using this powerful package. Those who already know LaTeX will still find LyX useful for writing shorter pieces of text such as letters. And, after all, as the LaTeX file format is supported in output, you even have the freedom to hack the most complex LaTeX commands by hand if necessary.

Ulrich Quill received his diploma in physics from Ruhr-University Bochum, Germany. His thesis topic was image analysis with neural networks. He is now in the Ph.D program in the Department of Neurophysiology, working on biophysically realistic simulations of neural networks. He helps with the system administration of a SUN/Solaris, PC/Linux cluster. In his spare time, he enjoys reading, photography and spending time with his girlfriend. He can be reached at quill@neurop2.ruhr-uni-bochum.de.

Archive Index Issue Table of Contents

Advanced search

x-automate: Control Your Home with Linux

Stewart Benedict

Issue #57, January 1999

Mr. Benedict show us the way to live in the home of the future by using our computer to control lights and appliances.

Ever since I was a kid watching the Jetsons, I've envisioned living in the home of the future, with intelligent appliances and a central computer controlling the house, in sync with the occupant's activities.

Well, we're not quite there yet, but it has been looming on the horizon for some time. One of the early pioneers in the home automation industry was the X10 protocol. Pico Electronics, a small design house in Glenrothes, Fife, Scotland, developed the X10 project based on a previous project called X9, which was a random-access scheme for accessing tracks on an audio tape (much like CD technology of today). Pico was also responsible for designing some of the first calculator chips. While the X10 system does have its limitations (some of which are addressed in recent additions), it is a fairly cheap way to implement home automation without rewiring your home from scratch.

The X10 system consists of controllers and modules. The controllers send “power on”/“power off” messages through the existing house wiring to the controllers, which either throw a relay or use a triac circuit (an electronic circuit used in light dimming/motor speed control applications) to perform dimming operations. Each module is set to an address (1-16) and a house code (A-P), and the controllers are set to a house code. If multiple units are set to the same address, they can all be controlled by one message. If you need additional coverage, you divide the facility into zones by house code. X10 modules and controllers, as well as the CP-290 interface for computer control, can be purchased from a number of mail order vendors, or from Radio Shack.

One limitation of the X10 systems is one-way communication—no acknowledgement of the signal being received is returned. Also, the system is limited in range, sometimes having difficulty getting from one side of the 220V

line (split to provide 110V to the home) to the other, in the U.S. anyway. Bridge modules help with the second problem, and new units are now available that implement two-way communications (TW-523).

What does all this have to do with Linux? After having played around with Linux for a while and learning about **cron**, I started thinking about leaving my machine on all the time and controlling functions around the house using my existing X10 gear and the CP-290 interface which I used to plug into my old DOS box to control the house lights. The CP-290 can download programs and run the house on its own, but by letting Linux do the job, I get the extra bonus of letting the system do other tasks such as upload/download my mail, announce the time of day and give me verbal reminders.

Fortunately, someone else had the same idea. Aaron Hightower (aaronh@acm.org) wrote **x10-amh**, a command-line program which communicates with the CP-290, controlling those items a person might wish to control around the house.

x10-amh alone is enough to work with cron and do tasks automatically; you just make the appropriate x10-amh commands in your crontab file, and off you go. Here's a sample from my crontab file:

```
55 4 * * mon-fri exec /usr/local/bin/x10 -n 3,4,11
15 5 * * mon-fri exec /usr/local/bin/x10 -f 3,4,11
0 7 * * mon-fri exec /usr/local/bin/x10 -f 10,11,12
0 23 * * sun-thu exec /usr/local/bin/x10 -f 1,4,3,9,6,7
0 9 * * mon-fri exec /usr/local/bin/x10 -f 1,2,3,4,5,6,7,8,9,10,11,12
```

I'm a frequent participant in Usenet, and one of the groups I follow is comp.home.automation. If you read this group for any length of time, you see a plethora of announcements of the latest, greatest home automation software, with fancy GUIs written for MS Windows. Being an acknowledged Linux advocate, I started thinking that Linux deserved a fancy home automation GUI of its own.

Figure 1. x-automate Interface

x-automate is a Tcl/Tk frontend that works in conjunction with Aaron's x10-amh. It's fully customizable, with a remote-control-type interface (Figure 1) and an assortment of icons representing a few typical home-automation candidates around my home. In addition to the remote control interface, you can set up floor plans of your home (Figure 2), with the devices placed in a reasonable representation of their actual locations. Carrying the Jetsons' theme one step further, I also wrote a C program, **speak**, that sends ASCII text through another serial port to a bgmicro Digitalker speech synthesis board which then “says” what the system is doing. This part is optional. You can configure the system

without sound or implement another approach such as **cat xxx.au > /dev/audio** with the appropriate sound file.

Figure 2. Home Floor Plan

The program and its associated files should be stored together in a directory. (In my case, it's ~/x10.) You'll have to run the program from its directory to pick up the bitmap files. The configuration for your X10 modules is stored in ~/.x10rc and should look something like the one shown in Listing 1.

Figure 3. Configuration Interface

The program includes a tabular input section for modifying your configuration (Figure 3). The screen is much like a spreadsheet, with cells to edit the various parameters for each unit. You need to indicate the floor the unit is on (for multistory facilities), the x,y coordinates of the icon for the floor plan (you can fine-tune this once you load the floor plan), the house code and the unit number. In addition, you need to assign a descriptive phrase for the icon prefix and the device, whether it is dimmable, and the phrase to speak if you're configured for sound. You can tab from field to field, as well as insert and delete rows. Once you're finished, store the setup in your .x10rc file.

There is also a tabular screen for configuring scheduled events which can be uploaded to the CP290. **x10-amh** stores event data in the following format:

```
event {
        devmap 2
        daymap 1,2,3,4,5,6,7
        housecode a
        mode today
        minute 39
        hour 23
        function dim
        dimlevel 5
        }
```

Figure 4. Events Table

The table in x-automate simplifies editing this data (Figure 4). In short, you need to define the device, the days of the week for the event, the housecode, the mode, the minute, hour, function and dimlevel, if applicable. To upload this data from the command line, you run **x10-amh *filename***, where *filename* contains the schedule data in the above format.

x-automate also has the means for accessing other command-line options of x10-amh, mostly through the menu. You can set the CP-290's date/time, query the status of the CP290 and perform a self-test.

To create your floor plans, I recommend using **xfig**, saving or converting the file to an “xbm” file for displaying within x-automate. Keep in mind your screen size when designing your floor plan. A separate drawing can be made for each floor or area, with the file names entered in the .x10rc file. Your control icons will be displayed at the x,y coordinates as defined in your .x10rc file, and can be used to control devices in the same way as with “remote control”. Clicking on the edge of the icon and dragging will move the icon on the floor plan and update the x,y coordinates in the configuration, which can be resaved to your .x10rc file. A group of icons in both large and small sizes (for the floor plan) are included in the distribution. Feel free to make your own—if you make some nice ones, send them to me. Now, Linux users can have a fancy home-control GUI, just like that “other” operating system.

Resources

Credits

Stew Benedict has been hacking with computers since 1983. Discovering Linux about two years ago, he saw the light and is now working as a system administrator on a mixed UNIX/Microsoft network in a manufacturing facility. When he's not staring blankly at a CRT, he loves to spend time at home with his wife, daughter, dog and four birds. He can be reached at stewb@earthlink.net.

Archive Index Issue Table of Contents

Advanced search

A Short History of Women in Technology

Thomas Connelly

Issue #57, January 1999

If you think all computer professionals are men think again. Mr. Connelly tells us about some well-known women in computer annals.

One of the many public debates in Australia at the moment is on the question of women in the computing industry. For many people, the computer industry and computers in general are seen to be a domain where big boys play with toys. Of course, in a society and economy based on the division of labor this may very well be true, but that is another article. The heads of all the large companies are men: Bill Gates and Steve Jobs to name but two. However, the same must be said of almost all companies and institutions in modern society. The computer industry does not exist in a vacuum; as much as anything else in our world, it is a plaything of larger forces.

What of the role of women in computing? From the earliest days of computing to the writing of the Standard Template Library, women have played an active and leading role in computer science. The following examples should quickly prove this statement to be true.

Ada Lovelace

We can start with a question: who was the U.S. Army's programming language named after? Ada Lovelace, daughter of the English poet Lord Byron. (Rather ungallantly, Byron left Ada and her mother, Anne Isabella Milburke, when Ada was one year of age, to seek glory in Greece, where he succumbed to a fever instead of leading a stirring charge—history can be quite unforgiving.) A brilliant mathematician, she worked on the analytic engine with Charles Babbage, devising a method of programming based on the cards used on a Jaquard loom—a type of input some of us older people can remember from standardized testing in our school days, or from the Simpsons cartoon, where Apu wrote a tic-tac-toe game in his university days (before becoming the fifth Beatle).

With their combined algebraic skills, the pair set off to the racetrack to apply logic to horse racing in an attempt to win enough money to build their machine. This effort resulted in Lady Lovelace having to pawn her jewelry to keep out of debt—a lesson learned, I am sure. Financial problems aside, the machine, which was never built in their lifetime, was completed not that many years ago and did work, just as Ada said it would in her paper “Observations on Mr. Babbage's Analytical Engine”. Before the project collapsed in a fury of bad debt, Lady Lovelace wrote a working program to calculate Bernoulli numbers.

In this early moment of computing, a woman was actively involved. Indeed, if it is true that women have the keener language skills of the two sexes, it would follow that they would be more than able to contribute to computer science.

Grace Murray Hopper

Skipping a few decades, we come to the attack on Pearl Harbor and the American entry into World War Two. The epic navy battles of the Pacific Theater of Operations showed the need to find a way to quickly calculate the flight of a shell fired from the great eight-inch guns of the USN. The math was simple enough (maybe not for me, but for others), but in the stress of battle, errors were not uncommon. A calculator was devised to make the work simpler and easier. In the pressure of war, expediency won out over ingrained sexist ideas, and many women were recruited for the projects, which in a few years led to the birth of the electronic computer.

One of the most significant of these young women was Grace Murray Hopper. A slight woman, who taught at Vassar before the war and was obsessed with nanoseconds, she talked the USN into allowing her to volunteer even though the Navy preferred to have its scientific researchers as civilians. In the Bureau of Ordnance Computation, she worked on the early computers—vast machines weighing many tons and needing crews of programmers to work them. Tasks were performed by plugging wires into the back of the machine. Many of the wire-plugging programmers were women.

Grace Hopper, later promoted to Rear Admiral, is credited with many innovations in her field. Among the most important was her first use of the word *bug*. A moth once flew into the machine, and was “battered to death” by a relay. Grace, upon extracting the poor dead insect, taped it into one of her notebooks and wrote, “The first actual case of a bug being found.” A new phrase for the source of a hair-tearing error was coined. On a more serious note, her laziness (one of the virtues of a programmer) led her to develop the first compiler for the UNIVAC in the mid-fifties. Until then, all coding had been done in machine code, a time-consuming and often frustrating activity. The ability to write English words to get the job done was a great advance in

computer science, although it met with strong resistance from engineers at the time. Grace Hopper learned to loathe the phrase “but this is how we have always done it.”

The invention of the compiler led directly to her work on the development of the FORTRAN and COBOL programming languages, which she helped write and later refined and standardized as a member of the Standards Committee. COBOL, notwithstanding the success of C, is still the most common language in use today; more lines of code are produced in COBOL than in any other language. It is a fitting testimony to her achievement. The invention of the compiler is one of those things that is easy to take for granted, but for ease of use and the ability to port code, it is a very powerful tool.

Adele Goldstine

Another woman working during WWII was Adele Goldstine, who in 1946 revamped the ENIAC as a stored program computer, and is responsible for the quote “It was a son-of-a-bitch to program.” This development of the stored program allowed the computer to perform a new task without reconfiguring the entire system. She wrote the manual for the ENIAC as well.

Betty Holberton

At the same time, Betty Holberton was working on the UNIVAC and concerning herself with human engineering (i.e., user-friendliness). She developed a language called C-10, which allowed commands to be typed in rather than having to reset all the wires. Her system used mnemonic characters to input, for example, “a” for add and “b” for bring. In her work, she initiated the standard we still use today—the numeric pad next to the keyboard. In spite of all these efforts, she must be taken to task for her insistence that black was too intimidating a color for a computer, which resulted in the use of that horrible beige color for modern computers.

Conclusion

This is a discussion of only a few of the women involved in the development of the computer; however, many features we take for granted were developed by these women—the keyboard layout, the compiler, the stored program, the ugly colors and more.

Finally, two other women should be mentioned: first, the Editor of *Linux Journal*, Marjorie Richardson (I've never been one to miss a chance to court favor), and the author of *Essential System Administration*, Æleen Frisch. This text may have been of the most use to me in my vain attempts to conquer Linux.

More information about women in computer science can readily be found on the Internet at The Ada Project web site, built by Yale University, http://www.cs.yale.edu/HTML/YALE/CS/HyPlans/tap/tap.html.

Thomas Connelly lives in Sydney, Australia, with his partner Lyssa Wallace and his daughter Sofia, where he does market research (boring) to relieve the tedium fights with his Slackware Linux. He hopefully learns new tricks almost daily. He can be reached via e-mail at piglet@intercoast.com.au.

Archive Index Issue Table of Contents

Advanced search

The Proper Image for Linux

Randolph Bentson

Issue #57, January 1999

Dr. Bentson did a survey of Linux kernel developers to find out about their backgrounds. Here are the results.

I get mail from folks about my book, the device driver I wrote for Linux, and about articles I've written for *Linux Journal*. A few months ago I got one which said, in part:

> My boss is a great guy to work for ...[but he] is of the opinion that Linux is the work of “college punks” and will not consider it for serious work.

> He had a nightmare with the MINIX file system and is permanently convinced that UNIX simply cannot be trusted and that Linux is the work of pimply-faced sophomores with time on their hands. I got a good laugh out of that while looking at your picture and reading your bio.

I can only hope his laughter was kindly. The opinions expressed by his boss weren't the first I've heard of that sort. Nor, I fear, will this be the end of it. Nonetheless, I decided to take a shot at confronting these claims.

I had suspicions that Linux contributors are a bright, experienced and well-educated bunch of folks. The discussions in the various Linux newsgroups and mailing lists aren't lightweight, nor is the resulting operating system. My “feel” of the operating system is that it's based on a lot of mature judgments and there is some theoretical grounding in what's being done.

Credits

I gathered up a list of contributors (from /usr/src/linux/CREDITS) and sent off 241 notes. Partial text of this note is shown in the sidebar, “Letter to Contributors”. I sent my notes with some trepidation—I didn't want to bother

folks while they were working on important projects, and I feared a lack of response.

Letter to Contributors

I needn't have worried. So far I've received 103 replies, many of which have included a few words of encouragement. It seems that I wasn't the only one who wanted to respond to unjustified complaints about Linux. (Another 29 notes were returned with address errors. I hope to see corrections to the CREDITS file.)

Education

The level of response was the first piece of good news. The second was that I've been stunned by how strong the development team is with regards to both credentials and experience.

From these replies I found:

- 1 had completed just basic public education (high school)
- 15 had attended college or technical school
- 23 had an undergraduate degree (B.S., B.A., etc.)
- 19 had attended graduate school
- 15 had a graduate degree (M.S., M.A., etc.)
- 9 had done further graduate work
- 19 had a terminal degree (Ph.D., M.D., etc.)

That's got to totally demolish the image of college hackers—at least the sophomore part of it. I figured I was an exception when I started working on the Cyclades driver while avoiding rewriting my dissertation. I thought, once folks were awarded a Ph.D., they would be busy with research, teaching or some other interest. I guess Linux development may be the doctor's favorite hobby.

When I offered an earlier summary of these results, my correspondent reported that his boss wisely intoned, “those folks are all academia and none of them have ever tried to run a business.”

Experience

I had sort of expected a comment along those lines and fortunately asked a few more questions in my survey. One hundred of the replies also reported the number of years spent programming or doing system design.

- 4 had 1 year
- 10 had 2-4 years
- 31 had 5-9 years
- 40 had 10-20 years
- 16 had 20+ years

More than a few of us were programming before the integrated circuit came into general use. (Perhaps a mixed blessing—some of us may still suffer from post-FORTRAN syndrome.)

As I noted earlier, I have also felt that Linux has benefitted from a broad experience in its developer base. Linux may be a first operating system for a lucky few, but almost everyone (all but three) claimed to be at least a skilled user of another operating system. Eighty-three were skilled users of several other operating systems.

Nor was their contribution to the Linux kernel the first of that sort. Twenty have contributed to another operating system and another twenty-two have contributed to several other operating systems. One reported:

> Speaking for myself, I had the same idea Linus did, but he beat me to it. (I've heard others say this as well.) I knew how to build a UNIX-like system from the ground up, and there was a need for it for PCs. (Vendors were charging exorbitant amounts for poor products in those days, and there was no good 32-bit development system for 386s.) I just didn't have the time. I had been playing with MINIX when Linus showed up on the MINIX newsgroups, and it took off from there. I can tell you that though I was a student at the time, I'd been a professional systems programmer for many years before. So, I and many others knew what professional quality software was, as well as how to produce it. I think it turned out pretty well.

Current Use

Finally, I wanted to know if the contributors were “doing Linux” in their careers. Eighty-two said their current employment was based on their computer skills. It was interesting to note that over a third reported their current employment supported or relied on their Linux development efforts. Sadly, two reported

they were currently unemployed, but one of those also noted that he was "voluntarily unemployed to have time to put my life in better order."

Perhaps one significant difference between Linux development and academic or commercial development is the duration of personal interest. In an academic setting, a student typically has one term, or at most one year, to work on any given program. When programmers leave a company, support is picked up by someone who has no sense of what has gone before. There is greater continuity in the Linux community because of the nature of submission and distribution. No matter what is happening at school or where one works day to day, contributors can keep in touch with progress on their piece of the puzzle. One person noted, "Personally, I did start my code in school, but that does not stop me from maintaining it now."

Motivations

There are some other issues which weren't addressed by my survey. Although it might not seem relevant to quality and performance, a person's interest has a great deal to do with the outcome—it leads to a distinction between "craftsmanship" and "work product". Another person noted:

> "Intent" is what I think all of these debates are about. In the commercial world there is only one true answer to "Why are you helping develop Linux?"—"To make a living." In the Linux community I'm quite certain the answer would be more closely aligned to "For me to use." The Linux community tends to be self-driven and self-motivated, and that is what leads to the successes and the apparent failures in our development environment.

> We are not a company; we don't have any one person, or group of people, setting the direction Linux will take. That direction is set by those with the energy to actually *do* something.

Another motive, akin to what pushed me to first join the effort, was shared by another respondent who said, "When I wrote [my code] for the Linux kernel I was working at [my former employer]. Linux use there was extensive, and I wanted to give something back."

Motivation leads to the final and most significant issue—one which cannot be examined by a developer survey.

Quality

In a world driven by marketing, image is the basis for purchasing decisions. Even if a good image could be established for Linux by listing credentials or

tabulating years of experience, I'd be reluctant to shift to that level. I'd much rather see acceptance and popularity for Linux based on quality and performance.

Even though I hadn't asked specific questions on this topic, a few people offered comments. One note seemed to identify, however obliquely, what may be the key to Linux's success.

> In general, my experience is that most software I have seen which was developed by students is not of the professional quality I would like to see. On the other hand, much of the commercial software I have seen, which was developed by professional software development companies, is also not of the professional quality I would like to see. The difference is most people do not get to see the internals of commercial software.

Developing on this theme, another wrote:

> The reason Linux is stable and usable is not because of its student programmers [or lack thereof]. It is because of the overwhelming feedback that alpha and beta testers provide. When you read the Linux kernel, you will find many parts are poorly structured, poorly written and poorly documented. However, people dared to test it and report their problems; Linus and friends respected the error reports and went ahead to fix them. That is why it works so well.

> In addition, psychology sometimes causes weird effects. If a user discovers a bug in his system, reports the bug and sees it fixed eventually, that user is happy because he was treated with respect. Most likely, he is even happier than he would be in the bug-free case.

We not only need to bring the CREDITS file into an accurate state, but we also need to acknowledge the thousands who have contributed to Linux by using it and sharing their discoveries—good or bad—with others.

Peter H. Salus reports the UNIX philosophy in *A Quarter Century of UNIX* as:

- Write programs to work together.
- Write programs that handle text streams, because that is a universal interface.

I'd like to close by adding another entry, suggested by UNIX and dominant in Linux:

- Write programs you enjoy.

Postscript

I just received a note from the person who sparked the original survey. He reports:

> I took my “hand-me-down” Linux box, an unimpressive 75MHz Pentium with 64MB RAM and a tiny 600MB HD to work. My boss was amazed that office applications such as StarOffice were available and was quite impressed when I read a Word document with StarOffice and then converted it to HTML. Samba was another revelation. Overall performance impressed him. In a few crude tests, it outperformed a “commercial” system running with 128MB RAM, dual 200MHz processors and all ultra-fast/ultra-wide SCSI drives.

> After a couple of callers indicated an interest in UNIX versions, we checked the price of current systems. My boss decided Linux was indeed priced right, and asked me to start on a port.

It looks like we've won one more away from the dark side.

Randolph Bentson 's first UNIX experience was booting a BSD VAX system on July 3, 1981—the whole town had a celebration the next day. Dr. Bentson started contributing to the Linux kernel in May 1994, and his book Inside Linux: A Look at Operating System Development describes how many modern operating system features have evolved and become essential parts of Linux. He can be reached at bentson@grieg.seaslug.org.

Archive Index Issue Table of Contents

Advanced search

Understanding a Context Switching Benchmark

Randy Appleton

Issue #57, January 1999

A look at the Linux kernel scheduler.

One of the most important tasks of an operating system kernel is to manage processes and threads. A process is a program in execution, and a thread is just a CPU state stored within a process. A CPU context is either a process or a thread. Most processes have only one thread, but some processes, particularly servers, have more. Sometimes programs other than servers use multiple threads; Netscape Navigator is an example of such a program. These processes and threads represent the programs the user has selected to run. If managing processes and threads is slow, overall computer performance will also be slow.

Linux advocates naturally assume that Linux offers better performance than MS Windows. Gregory Travis (greg@littlebear.com) set out to test this assumption by testing the times needed to manage processes and threads. His benchmark was a simple program that timed simple operations like **fork** or **sched_yield** by generating a large number of processes or threads that did nothing more, individually, than looping in place. His results generated quite a controversy in the Linux newsgroups and the kernel e-mail list. All tests were done on a 200 MHz Pentium. (See Table 1.)

Table 1

Linux can create a process twice as fast and a thread three times as fast as NT. Process creation takes longer than thread creation (12x for NT, 3x for Linux), because processes have much more overhead. Memory maps and file descriptor tables are just two of the things that must be created for a process (but not for a thread). Note, however, that Linux creates an entire process (the clone function) in about the same amount of time NT takes just to create a thread (1.0 ms vs. 0.9 ms).

Using this benchmark and an unmodified Linux 2.0.30 kernel, NT is much faster than Linux at context switching, either between processes (1.9x) or threads (3.2x). Since context switching occurs much more frequently than context creation, it would seem that Linux has a problem. But does it?

Understanding the Problem

Why does this simple benchmark make the Linux context switching code so much slower than Windows NT? To answer that, one must understand the Linux scheduler.

The scheduler has a list called the run queue containing all contexts ready to run. These contexts are not sorted in any way. Each context also has a goodness, which is a measure of the current priority of the context. Two loops are within the scheduler. The first loop (the searching loop) finds the context with the highest goodness from the ready queue and selects that context to run. The second loop (the recalc loop) recalculates the goodnesses if all contexts have used their entire time slice. This second loop runs only occasionally. The code in Listing 1 shows the structure of the schedule function.

The search loop needs a small bit of CPU time for every runnable process and needs it on every context switch. The above benchmark shows the context switch times with 40 contexts present and ready to run. That corresponds to a load average of 40 for a single CPU system or 20 for a dual CPU system. These are very large load averages, much larger than what is normally considered healthy. Generally, there will be only a few (perhaps one to three) contexts to choose from. Most processes and threads, even very active ones, will be waiting for some I/O event to occur and are, therefore, not on the ready queue and not considered for selection.

Even heavily loaded machines will generally have most contexts waiting for I/O to complete; those contexts will not be on the run queue. Consider the site http://www.winsite.com/, a very heavily loaded Internet site, with about 200 copies of **httpd** running as web servers. The total number of processes for all purposes is about 420. The machine is a 333MHz Pentium II connected to three T1 lines.

Measurements of this machine indicate that 24,221,164 context switches occurred over a 17-hour period (400 switches per second). For these switches, during 4% of the time there were over 10 contexts ready to choose from at one moment, and during only 0.1% of the time were there twenty or more. The longest run queue during the 17-hour stretch was only 36 contexts, out of the 420 possible. The mean run-queue length averaged only 2.5 contexts. In a sense, this benchmark represents a worst case for Linux, and the average case, even for heavily loaded machines, is much better.

The second (recalc) loop is even more important in understanding these benchmark results. This recalc loop runs only when every context on the ready queue has used up its entire time slice (generally 20 ticks or 0.2 seconds). The recalc loop then recalculates the priority of each context. Normally, this recalc loop runs only occasionally. Most rescheduling is done in response to an interrupt or because of an I/O request; therefore, most contexts do not use their whole time slice. However, when a process or thread wants to yield the CPU, it calls sched_yield, which treats it as if the process had used its entire time slice. In this way, any process that calls sched_yield has its priority lowered and will not be scheduled again for a substantial period of time. The code for sched_yield is shown in Listing 2.

When all contexts have used their entire time slice, the scheduler recalculates the priority of all contexts using the recalc loop. Since during this benchmark all contexts do one cheap operation (sched_yield) and then have **counter** set to zero, this recalc loop runs much more often than normal. Again, this benchmark seems to be a worst case for Linux.

For the web site described above, measurements show only one of 200 context switches required a recalculation of its priority—the other 199 context switches did not.

Half the problem can be solved easily; the other half would be more difficult.

The Search Loop

There is probably no way to make the search loop run faster; it is well-written code. However, it could be eliminated by simply keeping contexts in the ready queue in sorted order; then the next context to run is the context at the head of the ready queue. Keeping the ready queue in sorted order would require code to take each context being added to the ready queue and place it in the appropriate position. The time needed to find this position would add complexity to the scheduler. For large load averages (implying many contexts on the ready queue), there might be a considerable time savings, but in the more normal case of small ready queues, there is no significant savings.

The Recalc Loop

Minimizing the time needed by the recalc loop would be easy. Again, it is well-written code not likely to be improved upon, but it need not run as often as it does. By changing sched_yield, the recalc loop can run much less often.

For Linux 2.0.30, sched_yield acts as if the yielding context has used the entire time slice. Instead, what if sched_yield acted as if the yielding context had used only one tick of its time slice? Several effects would be noticed:

- The yielding process would have its priority reduced by one, rather than temporarily set to zero.
- The recalc loop would run much less often. Generally, it would run 1/20 as often (depending on process priority).

The new sched_yield is shown in Listing 3. Compare it to the one in Listing 2. Only one more line is included, yet a large increase in performance is shown for this benchmark.

Table 2 summarizes the performance after this change was made. Note that as the run-queue length increases, both the NT and the Linux scheduler take longer to context switch. Linux starts out being the faster context switcher, but NT does relatively better as run-queue length increases. For run-queue lengths of 20 or less (almost always the case in real life), Linux is better.

Table 2

Summary

By making a two-line change to the source code, this benchmark can be greatly improved. However, the benchmark arguably does not reflect real-life usage. Nevertheless, only a two-line change to the kernel is required for a significant benefit to a small number of users. After this change, Linux outperforms Windows NT in all aspects of process and thread creation and in context switching.

All listings referred to in this article are available by anonymous download in the file ftp.linuxjournal.com/pub/lj/listings/issue57/2941.tgz.

Randy Appleton (randy@euclid.nmu.edu) is a Professor of Computer Science at Northern Michigan University. He's been involved in Linux since the 0.99 days, and manages the largest collection of Linux computers in Michigan's Upper Peninsula. His other interests include foreign travel and flying small airplanes.

An Introduction to VRML

Tuomas Lukka

Issue #57, January 1999

Mr. Lukka takes a look at VRML basics including scripting, animation and applications.

VRML is intended to be for virtual reality what HTML is for text—a structured, standard, cross-platform format for static or interactive hyperlinked content. Just like HTML, it can be written by hand or generated by a program (the latter is usually preferable).

About a year ago VRML was catching on quickly, but much of that enthusiasm seems to have faded. It has not yet taken its place alongside HTML, JPEG and CGI as one of the basic, widely deployed web technologies.

One reason why this might be true is that no workable VRML browser for Linux or other UNIX-like operating systems is available. The ones that do exist are incomplete and generally not able to display web content in VRML well enough for serious use. For example, the cross-platform offerings are not able to handle Script nodes, which are one of the most important additions to the new VRML97.

My theory as to why this might influence the general acceptance of VRML is that a large fraction of people who are interested in cool technologies are using Linux and do not want to support an application that does not work with Linux and other properly working operating systems.

My scientific work needs the interactive visualization that VRML provides. While many different 3-D packages are available, VRML has the advantage in that it supports many sorts of interactions with the scene. Also, I would be able to send the diagrams I made with it to colleagues by e-mail.

After finding the available browsers to be unsatisfactory, I decided to write my own VRML browser. This browser, called FreeWRL, is rapidly approaching

VRML97 specification compliance, and supports most of the capabilities lacking in other Linux browsers. I wrote it in Perl, as it was the only language that would enable me to get it working quickly.

With the availability of FreeWRL, I think many free OS users might wish to evaluate or re-evaluate VRML. In this article, I'll attempt to lay out the basic concepts of VRML and explain how it might be useful.

To run the code listed here, you need a VRML browser. If you don't have one, and are running Linux or some other UNIX-like OS, go to http://www.iki.edu/lukka/freewrl/ and follow the installation instructions.

Notice in particular the words “if you have any trouble, e-mail me.” The very worst thing you can do with free software is to download it, see that it doesn't work for some reason, leave it, and tell your friends that it doesn't work. The author will never know what is wrong with his package and will not be able to improve it, while rumors will be spreading everywhere that the package is not worth trying. Good bug reports are the least you can do for free software developers in return for their effort. “Thank-you” e-mail is nice, but not nearly as vital.

The Basics

Instead of a lengthy description, let's start with the code for drawing a simple world (see Listing 1). We'll start with the inner section: a sphere is described with a radius of 0.5. (In VRML, the units are usually called meters but that is just a convention.) This **sphere** is the **geometry** of a **Shape** node, whose “appearance” has a **diffuseColor** of 0.8 0 0, i.e., very bright red. Therefore, the Shape actually describes a red sphere. This shape is one of the **children** of a **Transform**, whose **translation** is 0 1 0, i.e., one meter upwards on the Y axis, which, from the default viewpoint on the positive Z axis, is up. The first line is a comment telling the browser this file is written in VRML version 2.0.

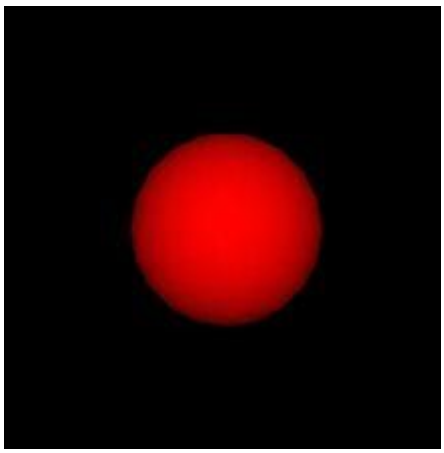


Figure 1

In short, we have just described a red sphere with a radius of half a meter and lying one meter above the origin. (See Figure 1.) Now the code in Listing 1 may seem a somewhat onerous way of describing such a simple scene. For example, the following might be easier:

```
# THIS IS NOT LEGAL VRML—SEE TEXT
Sphere {
        center 0 1 0
        radius 0.5
        color 0.8 0 0
}
```

In fact, you can do this: it is possible to create your own nodes suitable for your own application using *prototypes*, which are similar in spirit to preprocessor macros in C, as shown in Listing 2.

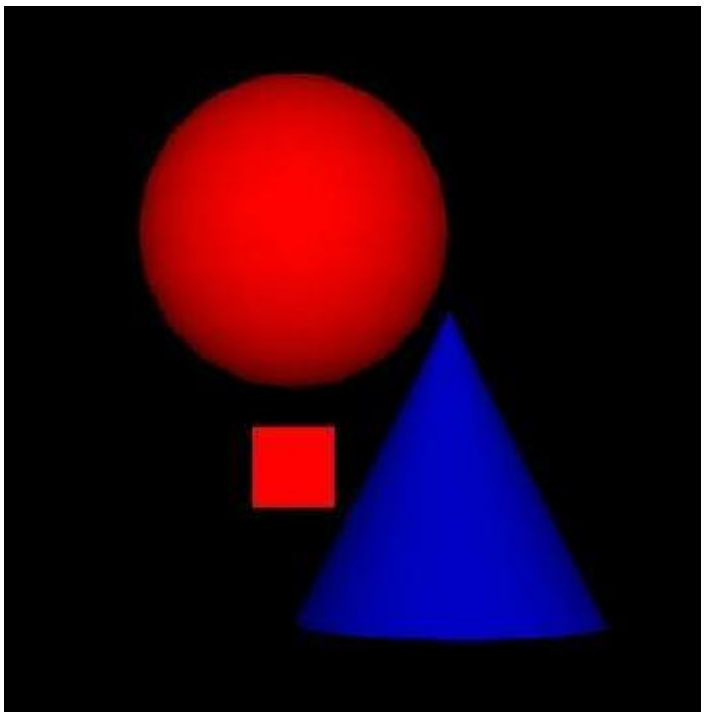


Figure 2

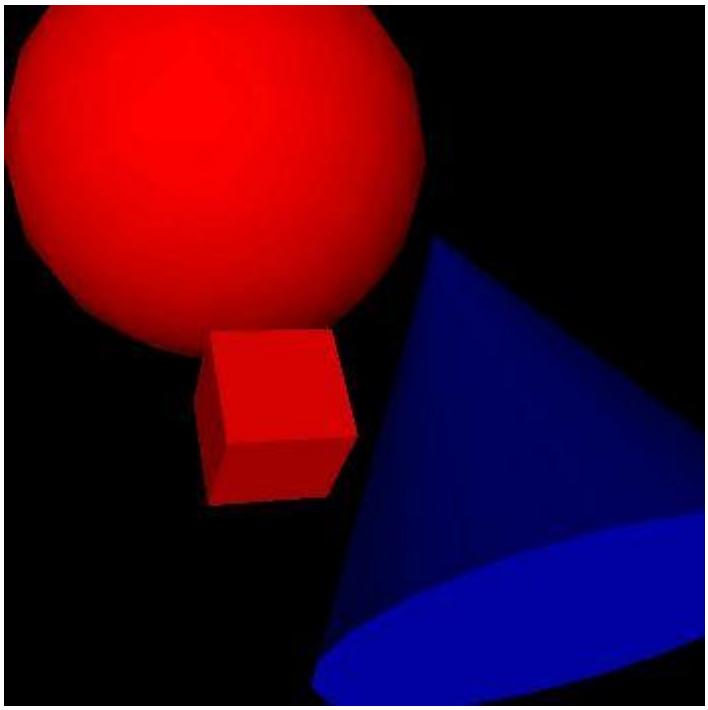


Figure 3

The scene described by the code in Listing 2 is shown from two different angles in Figures 2 and 3. Now you can define any number of scenes like this, using the **PROTO** you just defined. Of course, you do have to deal with including the original **PROTO** in the beginning. It is also possible to refer to it in another file using so-called **EXTERNPROTO**s, which include the interface part (in brackets) but omit the definition (in braces). Even this might sometimes feel like too much typing. In that case, you can easily write a Perl script to include the correct **PROTO**s in your VRML files, or to do whatever other repetitive tasks you need done.

In scene graphs, you can give a node a name using the **DEF** statement and use it again in another place with a **USE** statement—see Listing 3.

Note that unlike with **PROTO**s, where a complete copy of the contents of the prototype is made, only one Appearance node is present, which is referred to in two places. For example, if we later animate the appearance to change the color from red to blue, both the box and the sphere change color; whereas if we animate the color of a **Thing** in the **PROTO** example, only that one **Thing** changes color. Once a node has been named with **DEF**, it can be used in **USE** statements any number of times.

Now that you have a sense of what is going on, let's have a more formal description of what we just did. A VRML *world* is described as a hierarchy of *nodes* called the *scene graph*. Each node is of a particular *node type*, and for each node type the VRML97 specification defines various fields. Each field has a type and a default value. The syntax for a node is

```
NodeType {
        field value
        field value
        ...
}
```

where the syntax for a value depends on the type of the field. For example, the value of an **SFVec3f** is three floating-point numbers and the value of an **SFNode** is another node just like above.

The single-valued (SF) field types are:

- **SFBool**: a Boolean value, written **TRUE** or **FALSE**
- **SFFloat**: a floating-point number
- **SFImage**: an image, described by several integer values, first width, height and number of components, then width*height values for the pixels
- **SFInt32**: a 32-bit integer
- **SFNode**: a node
- **SFRotation**: a rotation: 3 floating-point values for an axis and one for the angle in radians
- **SFString**: a string in double quotes. Inside, double quotes and backslashes are quoted by a backslash
- **SFTime**: a floating-point value, time in seconds since a specific origin
- **SFVec2f**: a two-dimensional vector containing two floating-point values
- **SFVec3f**: a three-dimensional vector containing three floating-point values

For most SF field types there exists a corresponding MF field type, which simply means zero-or-more values of the corresponding SF type. For example, the following would be legal values for a **MFVec3f** field:

```
[]
0.1 2 3
[0.1 4 2]
[0.5 2 6 1 6 4 7 4 6]
[0.5 2 6, 1 6 4, 7 4 6]
```

That is, the values may or may not be separated by commas and must be surrounded by brackets if there are more or less than one of them.

These names for the field types are also used inside the **PROTO** declarations: the syntax of a **PROTO** is basically

```
PROTO Name [
        field Type fieldName defaultvalue
        exposedField Type fieldName defaultvalue
        eventOut Type fieldName
        eventIn Type fieldName
        ...
] {
        Node { ... }
```

```
        ...
}
```

where inside the **PROTO** body, field values can also be specified using **IS** statements, which equate that field with one of the published fields of the **PROTO** (e.g., the **center** field of the **Thing PROTO** above).

For descriptions of all the node types and their fields as well as the exact grammar and semantics of VRML, see the VRML97 specification (found at http://www.vrml.org/) or a book. Figure 3 shows Netscape displaying the definition for the **IndexedFaceSet** node (which makes it possible to describe arbitrary polygonal geometry) from the web site.

This is basically all you need to know to create static VRML scenes, except for the nitty-gritty details which you can find in the above-mentioned source. Once you do create worlds using FreeWRL, send me a note by e-mail and I'll put a link to your worlds on the FreeWRL web page.

Animation and Interaction

Of course, creating static scenes is not such a big deal. The truly interesting part of VRML is its ability to interact with the user. Suppose you want to demonstrate the concept of a cross product of two vectors to a friend. You can draw all sorts of 2-D diagrams, but what would be a better description than a 3-D graph in which your friend could adjust two vectors and see the cross product change in real time (especially since you can embed it into a web page using the HTML **embed** tag)? One of the FreeWRL demos (Figure 4) does exactly this, plus the vector sum and difference.

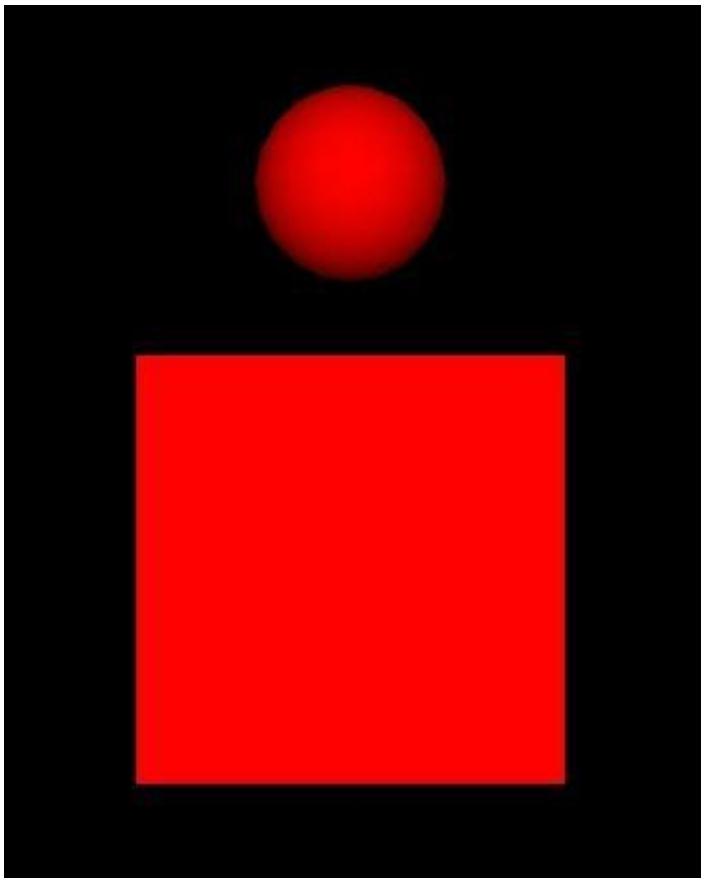


Figure 4

In the previous section, you saw how the Nodes describe the scene to be rendered. What I didn't mention is that nodes can send *events* to each other along specified *routes*. The event routes are independent of the scene hierarchy.

Routes are specified by ROUTE statements. For example, Listing 4 creates a white box (Figure 5), which cycles smoothly through red and blue when clicked, returning to white 2.5 seconds after the click. What happens when you click on the box? First, the BUTTON node senses it (most sensors in VRML sense events from their siblings, in this case the Shape node defining the box) and sends a touchTime event. The ROUTE statement causes that event to be routed into the startTime of the TimeSensor TS, which causes it to start generating time events for one cycle (the length of the cycle is 2.5 seconds). At each clock tick, the TimeSensor sends an event called **fraction_changed** with an SFFloat value between 0 and 1 (giving the fraction of the time in the cycle that has gone by). This event is then routed to CI which is a ColorInterpolator, i.e., it does piecewise linear interpolations of color values. CI then sends a value_changed event that MAT receives as diffuseColor, then the color of the box changes. For example, if CI receives a set_fraction event with the value 0.3, it checks its key field and notices that 0.3 is between the first and second value. It is six tenths of the way to the second value; therefore, the output is the color (0.6 0.6 0.6).

Having events go through an interpolator to reach their destination is a fairly common idiom in VRML, as this allows you to specify arbitrary mappings relatively easily and cheaply. Several different kinds of interpolators are in the VRML97 specification, for different data types.

Scripting

However, interpolators will take you only so far: for one thing, they can't toggle on a light at a particular point in the sequence. It would have been easy for the VRML97 specification authors to add a node type to accomplish this, but they chose not to. Instead, they chose to create a very general interface to external scripting languages, Java and JavaScript. You can define arbitrary behaviours for your world using these programming languages.

If we want to have the Box in the previous example disappear and be replaced by a small blue sphere for the first 1.5 seconds of each cycle, we need to use a Script node. This particular example is shown in Listing 5. The TS and CI nodes and the routes between them and MAT are the same as in the previous example.

As is obvious from the example, the syntax of specifying fields inside the Script node is similar to the **PROTO** interface section: instead of just saying **fieldName value** we are saying **kind Type fieldName** followed by *value* for fields. This is because the specification defines only three fields for the Script node: **url**, **directOutput** and **mustEvaluate** and leaves it up to the programmer to define the **eventIns**, **fields** and **eventOuts** of his script.

The actual script inside the Script node is specified in the **url** field. In this case, it is written in JavaScript and is embedded into the URL. It would also have been possible to write the script into a file (with the .js suffix) and to refer to that file in the URL. Alternatively, the script could have been written in another scripting language (e.g., Java or Perl).

Applications

So far, we've discussed how to write VRML. However, the most interesting results come from creating VRML automatically. Static worlds that are programmed once are interesting mostly for games, art or advertisements. Great possibilities remain to be exploited in the interface between the rest of the information world and the 3-D browser.

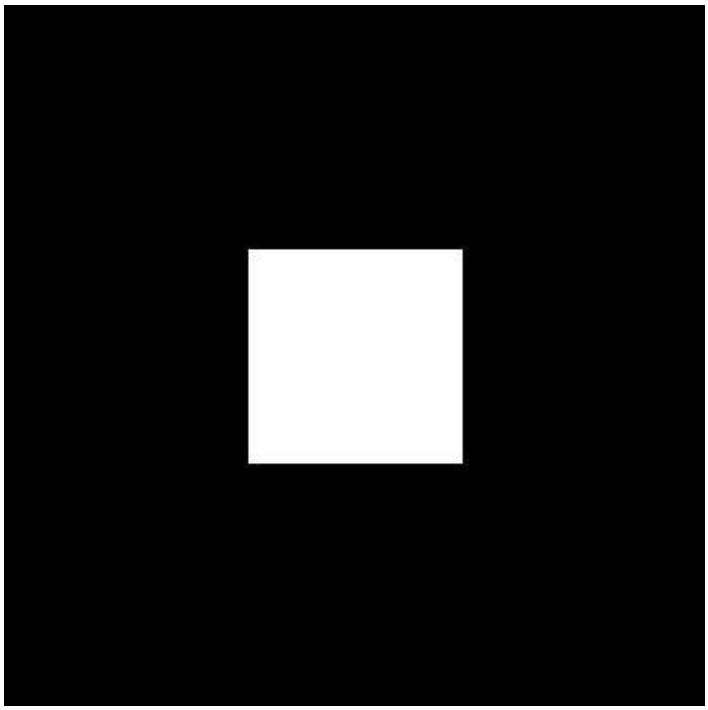


Figure 5

Also, we can use the API provided by the browser to create an application that uses VRML to provide a part of the GUI. A fairly simple interface can be written as a Perl program that uses the FreeWRL browser and GTK together to provide a user interface, which would be difficult to do in VRML or GTK alone. The main window consists of an entry into which you can enter a function you want the program to plot, a button to plot it, and three labels showing the X,Y coordinates and the function value at the point where your mouse last touched the function surface. The program also places a blue box at this point in the 3-D window. It would be trivial to add code to take 2-D or 3-D snapshots of this scene, as either the usual image files (GIF/JPEG) or VRML, once you have the scene displayed in the browser.

Basically, in addition to creating the GTK GUI, the code simply creates a browser window, loads a VRML world from a string and then calls a browser method to obtain a reference to an **ElevationGrid** node in the scene. It then sends events to this ElevationGrid to set the shape of the surface via the **height** field. The program also registers a listener for a **TouchSensor** node in the scene, so it is able to obtain the mouse position on the surface. The really interesting thing is that all the code for this application is under 200 lines with comments. (The application is included in the FreeWRL distribution, so I will not include the full source code here.)

It is also possible to access the VRML browser through a Java API called EAI (external authoring interface). This enables one to write web applets that access a VRML scene. At the time of this writing, FreeWRL partially implements

the Java EAI API but isn't yet able to provide this API while running inside Netscape. By the time you read this article, this situation may have changed.

Conclusions

Rather than writing a complete tutorial, I've tried to give an overview of what VRML is and what it might be useful for. I hope I've provided some new ideas that you can put to use at the right time. If you need more complete references, links to many sources can be found at http://www.vrml.org/, the web site of the VRML Consortium. The FreeWRL home page is at http://www.iki.fi/lukka/freewrl/ (iki.fi is a redirector to wherever the home page happens to be at the time).

All listings referred to in this article are available by anonymous download in the file ftp.linuxjournal.com/pub/lj/listings/issue57/3085.tgz.

Tuomas J. Lukka (lukka@fas.harvard.edu) got his Ph.D. at the University of Helsinki in 1995. He is currently on a three-year Junior Fellowship at Harvard University, spending his time doing research on artificial intelligence and molecular quantum mechanics, as well as playing music and writing free software.

Archive Index Issue Table of Contents

Advanced search

Getting Started with Quake

Bob Zimbinski

Issue #57, January 1999

How to get this most popular game to run on your Linux system.

Quake is one of the coolest games available for any platform. Thanks to Dave Taylor, who began id Software's tradition of porting their games to Linux back in 1995 with Linux Doom, today we have Quake for Linux. This article is meant to be a quick start to getting Quake running on your Intel Linux system. If you encounter problems not addressed here, look at the Linux Quake HOWTO at http://www.linuxquake.com/howto for more detailed troubleshooting information.

Figure 1. Player about to capture the blue flag in the “Capture the Flag” game

Necessary Files

The minimum system requirements for Quake are shown in the “System Requirements” sidebar. To install Quake on your Linux system, you will need some flavor of the official Quake distribution from id—either the retail DOS/Windows CD-ROM from a software store, or the shareware version downloaded from the Net. Alternatively, if you already have Quake installed on a DOS/Windows machine, you can use the relevant files from that installation.

In addition to the official Quake files, you will need Linux-specific binaries. All the necessary files for Linux Quake are available at ftp://ftp.idsoftware.com/. id's site can be very busy, so you may want to use one of their mirror sites (see Resources).

Version numbers in this article are current as of September 1998 and aren't likely to change. Quake is considered a finished product, so new versions will be released only if major bugs are found.

The shareware Quake for Windows distribution is necessary only if you don't have a Quake CD-ROM (ftp://ftp.idsoftware.com/idstuff/quake/quake106.zip).

Quake can render its graphics three ways: in an X11 window, full-screen SVGA, or hardware-accelerated OpenGL. You'll need to download the binaries only for the renderers you plan to use. (See Resources.)

QuakeWorld is a multi-player version of Quake optimized for play over the Internet. Get one of the packages listed in Resources if you plan to play on-line. Red Hat 5.x/Debian 2.x users should get the glibc version. The .rpm and .tar.gz package contents are identical. Choose one according to your distribution.

If you plan to run an Internet QuakeWorld server, select one of the dedicated server-only binaries (see Resources). Most people won't need them.

System Requirements

Installation

Start by creating the directory in which you will install Quake. The “standard” location is /usr/local/games/quake. The QuakeWorld RPM package installs its files in this directory, so it is a good idea to install here if you plan on installing QuakeWorld later.

```
mkdir /usr/local/games/quake
```

Installing From CD-ROM

If you have a very early release of the Quake CD-ROM, these instructions won't work. Please see the Linux Quake HOWTO for details on installing from older CD-ROMs.

A file on your Quake CD-ROM, resource.1, is an lha archive of all the Quake game files (lha is a file compression and archiving format like tar or zip). We will use the **lha** command (see Resources) to extract it.

Mount your Quake CD-ROM, move to your Quake directory and extract the resource.1 archive:

```
mount /dev/cdrom /cdrom #change for your system
cd /usr/local/games/quake
lha e /mnt/cdrom/resource.1
```

Your /usr/local/games/quake directory should now contain a bunch of new files and a subdirectory called /id1. The most important files for Linux Quake are in /id1, so you can safely remove everything else. If you are totally new to Quake

(or even if you are not), you may wish to keep the *.txt files for reference. On my system, I put all the READMEs that accumulate into a /doc subdirectory.

Installing the Shareware Version

The single-episode shareware version of Quake has all the features of the full version of Quake, with a couple of major limitations: you cannot play QuakeWorld (multi-player) with it, and you cannot play custom or modified levels.

Installing the shareware version of Quake is not much different than installing from the CD-ROM. Put the quake106.zip file in your Quake directory, then extract the resource.1 lha archive:

```
cd /usr/local/games/quake
unzip -L quake106.zip
lha e resource.1
```

Now save the README files (optional) and remove everything else except the /id1 directory:

```
mkdir doc
mv *.txt doc
rm -f *
```

Installing from a Pre-existing DOS/Windows Installation

If you have Quake installed under Windows or DOS on a different machine, you can transfer the files in quake/id1/ to your Linux system via FTP or some other mechanism. Keep in mind that the file names on your Linux system must be in lower case for Quake to find them, so you may have to rename them after the transfer. Also note that it may be necessary to delete your DOS/Windows installation after you do this, to remain in compliance with the terms of id's software license.

If your DOS/Windows and Linux systems are on the same machine, you have two options: copy the files from your DOS/Windows partition to your Linux partition, or link to the necessary files from Linux. Both options work equally well. You save around 50MB of disk space when you link instead of copy.

Whatever you choose to do, start by changing to your Quake directory and creating a new subdirectory called /id1:

```
cd /usr/local/games/quake
mkdir id1
```

To copy the files from your DOS/Windows partition, type:

```
cp /win95/games/quake/id1/*.pak id1
```

To create links to your DOS/Windows Quake files, type this instead:

```
cd id1
ln -s /win95/games/quake/id1/*.pak .
```

Of course, you should replace /win95/games/quake in the examples above with the correct path to your DOS/Windows partition and Quake directory.

Linux Binary Installation

Now it is time to decide which of the three Quake executables you would like to install.

- X11 Quake allows you to run Quake in a window on your X desktop. It is the least exciting client, but is a great, safe way to test your installation.
- Squake is the SVGAlib Quake client; it runs full screen on your console.
- GLQuake is the OpenGL Quake client. If you have a 3Dfx card, this is a must-have.

Download the packages you want (see the “Necessary Files” section) and extract them to your Quake directory:

```
cd /usr/local/games/quake
tar -xzf XXXX-i386-unknown-linux2.0.tar.gz
```

Sound Considerations

If you want sound from Quake, /dev/dsp needs to be readable and writable. Most distributions give it 662 (rw-rw—w-) permissions by default. The simplest solution is just to chmod 666 /dev/dsp. On most systems, the ability to read from the sound device will not pose a significant security threat. If this approach is unacceptable for your system, create a group that owns /dev/dsp and make your Quake players members of that group.

If you don't have a sound card installed or configured for your system, make sure to use the **-nosound** command-line option when starting Quake. Failure to use **-nosound** will cause Quake to exit with a segmentation fault when it tries to initialize your non-existent sound card.

X11 Quake

If you installed the X11 client, your system may need further configuration for glquake and squake, but at this point quake.x11 should be ready to go.

```
cd /usr/local/games/quake
 ./quake.x11
```

If all is well, a small window running a Quake demo should appear. You should hear sound effects and possibly music, if the CD is mounted. You can use the **-width** and **-height** command-line options to create a bigger window.

SVGAlib Quake

Both squake and glquake require SVGAlib to be running (glquake uses SVGAlib for keyboard and mouse input, in case you were wondering). SVGAlib comes with most modern distributions and must be properly configured before squake or glquake will run correctly.

libvga.config is SVGAlib's configuration file. On most systems, you will find it in either /etc or /etc/vga. Make sure the mouse, monitor and video card settings in this file are correct for your system. See the SVGAlib documentation for more details.

If you don't already have SVGAlib on your system, it is available on Sunsite (see Resources).

If you have a Red Hat 5.x or other glibc-based Linux distribution, remember that since Quake was compiled with libc5, all the libraries it links to (such as SVGAlib) must also be libc5-based. If you are going to compile a newer version of SVGAlib yourself, make sure it links to libc5 (and friends) rather than glibc, or Quake won't run.

Once svgalib is properly installed, you are almost ready to run squake. **squake** needs to run with root privileges in order to access your sound and graphics cards. One (bad) way to deal with this is always to run it as root. Responsible system administrators will cringe at this filthy suggestion. Making the Quake binaries setuid root is a more acceptable solution. Quake can then be run by regular users and still have the privileges it needs to access the graphics and sound devices. Be warned that any setuid program represents a security risk. A clever user could exploit a bug or security hole in a setuid program to gain root access to your system. If you don't run a multi-user system, this will not be a big concern.

Make squake setuid root with the following commands:

```
chown root squake
chmod 4755 squake
```

Note that you should run squake from a virtual console. It won't run from X unless you are root when you start it, and running a game as the root user is a situation to be avoided. If you are in X, do a **ctrl+alt+f1**, log in and then:

```
cd /usr/local/games/quake
./squake
```

Figure 2. Player being blown up with a rocket launcher

Figure 3. Scene showing pretty transparent water

GL Quake

Hardware-accelerated OpenGL Quake is Quake the way it was intended to be. There is no substitute—once you have experienced it, there is no going back.

To make GLQuake work, you need a 3-D card with the 3Dfx Voodoo or Voodoo2 chip set on it, the glide library, the Mesa library and SVGAlib. Getting your 3Dfx card working under Linux is a big topic, one I will discuss very briefly here. See Resources for places to find more information.

First of all, make sure SVGAlib is installed and properly configured as outlined in the previous section. Remember, glquake uses SVGAlib to get mouse and keyboard input.

Next, get and install the glide library. Glide is a library that provides an API for programming 3Dfx-based cards. If you want the Mesa graphics library to use your 3Dfx card, you must have it. Select the package(s) appropriate for your system (see Resources) and install according to the instructions on the web page.

Note that unless you download the 3Dfx-device-driver package in addition to the Glide library, you will be able to run Glide applications (like GLQuake) only as root. Install the /dev/3dfx module and you can play GLQuake as a regular user.

Once you have glide installed, try out the test program that comes with it. Remember this test program; it is a good way to reset your display if you ever have a glide application (like GLQuake) crash, leaving your screen switched off. Run this test from a VC, not X. It is possible for the test application to lose mouse and keyboard focus in X; then you'd have no way of shutting it down. Type **usr/local/glide/bin/test3Dfx** and your screen will turn blue and prompt you to press any key. After you press a key, you will be returned to the prompt.

Now you need to install Mesa, a free OpenGL-like graphics library by Brian Paul (brianp@elastic.avid.com). Luckily, you won't have to look far, because Mesa 2.6 is included with the QLQuake and QuakeWorld binaries. All you have to do is move it to the right place:

```
cd /usr/local/games/quake
cp libMesaGL.so.2.6 /usr/local/lib
ldconfig
```

If you want to upgrade Mesa to a more recent version (Mesa 3.0 should be released by the time this is printed), download the latest version from ftp://iris.ssec.wisc.edu/pub/Mesa. When installing Mesa 3.0 or higher, keep in mind that glquake is linked against libMesaGL.so.2, so you must create a symbolic link from your new libMesaGL.so.3.0 to libMesaGL.so.2 in order for glquake to find it. Also, as I mentioned earlier, remember that since Quake is a libc5-based application, all the libraries it links to must also be built with libc5. A libMesaGL linked against glibc will cause glquake to abort with a segmentation fault and possibly hang your system.

Now that SVGAlib, glide and Mesa are installed, you should be able to run glquake. Switch to a VC if you are in X (**ctrl+alt+f1**) and start glquake:

```
cd /usr/local/games/quake
 ./glquake
```

QuakeWorld

QuakeWorld is a multi-player version of Quake that is optimized for Internet play over a modem. Problems with the original Quake's network code, like excessive lag and packet loss, are reduced or eliminated in QuakeWorld.

To play QuakeWorld, you need the full, registered or retail version of Quake and a Linux QuakeWorld client. QuakeWorld clients come in the same flavors (X11, SVGAlib and OpenGL) as normal Quake, but are bundled together in one package. The prerequisites and configuration for these binaries are the same as for regular Quake, so if necessary, refer to the previous sections for help on setting up SVGAlib or glide/Mesa.

If you are installing one of the RPM QuakeWorld packages, installation should be as simple as typing the following:

```
su root
rpm -Uvh qwcl-xxxxx.i386.rpm
```

To install from the tar.gz packages, type:

```
cd /usr/local/games/quake
su root
tar -xzf qwcl-xxxx-i386-unknown-linux2.0.tar.gz
```

Four new executables (qwcl, qwcl.x11, glqwcl and glqwcl.glx) will be installed in /usr/local/quake. glqwcl.glx is a GLX application linked against standard OpenGL libraries. This should allow QuakeWorld to run with OpenGL implementations other than Mesa. The programs qwcl, glqwcl and glqwcl.glx

are installed setuid root so that the graphics devices on your system can be accessed. If you installed the /dev/3dfx driver mentioned in the GLQuake section, you can remove the setuid permissions on glqwcl and glqwcl.glx.

Once QuakeWorld is installed with your Quake files, you can start it up by typing the following:

```
./qwcl +connect some.server.address
```

Related Software

Qstat is a command-line utility created by Steve Jankowski (steve@activesw.com) that returns the status of Internet Quake, QuakeWorld and Quake 2 servers. Qstat (see Resources) is a must-have tool if you are planning on doing any Internet Quaking.

XQF (see Resources) is a graphical front-end to Qstat that uses the GTK toolkit. This is currently the best QuakeWorld/Quake 2 server browser in existence. Roman Pozlevich (roma@botik.ru) is still cranking out new versions at the rate of about one per month. If you are familiar with GameSpy for the Windows platform, this is the closest thing to it for Linux.

Resources

Bob Zimbinski is soft and pasty from years of playing computer games. When not playing games or working as a consultant for Control Data Systems, Mr. Zimbinski sleeps. He is co-author of the Linux Quake HOWTO. He can be reached via e-mail at bobz@mr.net.

Archive Index Issue Table of Contents

Advanced search

First Canadian National Linux InstallFest

Dean Staff

Issue #57, January 1999

Canada has a “Linux Day”--a novel method for getting the word out.

Saturday, September 26, 1998 was a big day for the Linux community in Canada—the First Canadian National Linux InstallFest was held.

The InstallFest was organized on a national level by CLUE (Canadian Linux Users' Exchange) to provide experienced help to those interested in installing Linux on their computers. CLUE is an organization that supports the development of local Linux User Groups and also co-ordinates events, corporate sponsorships and publicity at a national level. CLUE hopes that by enhancing association and communication amongst Linux developers, users, suppliers and the general public, it can increase the use and appreciation of Linux within Canada.

Highlights

A dozen different events were held by Linux User Groups across Canada, from Halifax to Victoria, all taking place on the same day.

The *Montréal* event, at its peak, had as many as 100 people in the room at once and by all accounts had 200 to 250 people stop by. They did 40 installs, only 20 of which were from preregistrations. They even got the crew of the local TV show *Branch* to stop by for an interview, due to air in November. Also worthy of mention is that they had guru Jacques Gelinas, author of the LinuxConf software, answering questions.

Two InstallFests were held in the *Toronto* area: one at Seneca College and the other at the University of Toronto Bookstore. The Seneca College event had a late start due to a power outage, but more than made up for it later as the unofficial count of installs was about 100. They even rolled out their Beowulf

class Linux cluster for the masses to look at and see how a few “small” Linux boxes can be turned into a “supercomputer”.

The *Manitoba* UNIX Users Group (MUUG) held their InstallFest at the University of Manitoba as a two-day event beginning on Friday. As this was their first InstallFest, they deliberately kept it small and aimed it mostly at the faculty and students of the U of M. About 140 people attended, with more than half purchasing a Linux CD, and MUUG did 19 successful installs. Attendance was greater than expected, probably due to the national news coverage the event received. At least one person came in who said he heard about the InstallFest from a segment on CTV News-1, a national news network.

The MUUG web site made mention of one more interesting story from the event. One attendee brought in a system which became known as “Franken-puter”! It appeared to be two separate cases tossed together with all sorts of spare parts the owner scrounged up, connected with a piece of coax Ethernet cable. He spent as much time swapping parts and reconfiguring on the fly as he did installing Linux. He apparently showed up at the start of the event on Friday and didn't finish until midafternoon on Saturday. Even after all that, he still hung around afterwards to help others with their installs.

The *Ottawa* InstallFest was hosted by the Ottawa Carleton Linux Users Group (OCLUG). While almost all the other events were held in a more academic setting of local colleges and universities, OCLUG had their event sponsored by NovoClub, a local retail store. NovoClub is located in a shopping mall and managed to get an empty storefront for OCLUG to use. They also arranged for display kiosks by several companies to be set up in the mall. There were training companies, a local ISP and most notably Corel Computer displaying their NetWinder. Of course, NovoClub offered specials on their very large selection of Linux products. The whole event was more like a mini-tradeshow than a typical InstallFest.

The unofficial count at the installation storefront was 250 people. This count included those who came to have Linux installed on their machines, members of the press and “just curious” folk who stopped to ask questions while wandering around in the mall.

OCLUG chose not to have people preregister; they decided to just let anyone come and register the day of the event. It was supposed to start at 10 AM and go until 5 PM. However, a line had formed by 9 AM when the mall opened and OCLUG soon ended up with a backlog of machines waiting for Linux installation. At 3 PM, they were two hours behind and had to start turning people away. By the time it was over, they had installed Linux on 50 to 60 machines and still had ten they could not finish.

Not all events were as popular as the ones listed above. The *New Brunswick* Linux Users Group had only ten people attend, with four successful installs. They were a bit disappointed with the low turnout. However, it was also homecoming week at Mount Alison University in town and a football game was in full swing at the same time as the InstallFest. They are in the process of designing a tutorial for their new users and anyone else who is interested. The *Fredericton* InstallFest was a little larger, with thirty attendees and ten installations.

Overview

The general consensus is that as a public relations event, the InstallFest was an overwhelming success. It got a lot of people asking questions about Linux, some of whom took the plunge and installed Linux for the first time. However, it was not completely successful as a technical event. By no means is this a reflection on either those who organized the individual events or the volunteers who helped with the installations—they all did a stellar job—just that no one was prepared for the magnitude of the response.

Most LUGs asked people to register prior to the event. This allowed a chance for the groups to get as many volunteers as they thought they would need. Some groups, such as the Vancouver Linux Users Group, were swamped with preregistration and had to halt registration prior to the event because they could not accommodate everyone. Even with preregistration, the day of the event was hectic. The report from Seneca College in Toronto was that their event lasted until 9 PM, and they were still unable to complete all the installs. Other events had similar reports, and despite the best-laid plans, demand overwhelmed the number of installers.

Some installs were unsuccessful, due to either time constraints or hardware compatibility issues that were not easily overcome. That said, the ratio of unsuccessful to successful installs was minimal. Overall, it was one or two to fifty. I've seen more failures than that on MS Windows installations.

Where do we go from here?

One interesting side effect of the OCLUG InstallFest was that preliminary discussions were started between Zenith Learning Technologies and Corel Computer to set up a corporate Linux training program. Also, Oliver Bendzsa of Corel Computer reported that he was as busy at the InstallFest as he was at Canada Comdex, a 3-day trade show in Toronto that drew some 50,000 people.

Dave Neill, a founding member of OCLUG, said that while grassroots events like the InstallFest are a great way to promote Linux, it is now time to start approaching local computer resellers and showing them there *is* a demand for

systems with Linux pre-installed. I work for Inly Systems, the largest independent computer reseller in the Ottawa area, and while we are now expanding the variety of Linux products we carry, we still do not offer Linux pre-installed on our machines. With at least three technicians on staff who have experience with Linux and/or UNIX installations, we could do this if people began asking for it. However, we are an exception; most resellers don't have technicians with Linux experience.

One issue that must be resolved is how and where companies can have their technicians trained. This is where training companies such as Zenith Learning Technologies come in. The fact that Zenith was at the OCLUG InstallFest shows that they realize the potential for Linux training. With such companies as Corel, Oracle, Intel and Netscape investing time and money in Linux, it won't be long before other training companies jump on the bandwagon.

Today Canada, Tomorrow the World!

Plans are already in the works for a Global Linux InstallFest next year. If you want to know more or would like to get your LUG involved, please check out the CLUE web site at http://www.linux.ca/ and contact Matthew Rice. An event of this magnitude will need lots of help organizing, so don't be shy—watch out Bill, the Penguin is on the move!

For more information on the individual InstallFest events, please visit the CLUE web site for a list of links to all the participating user groups.

Dean Staff (dstaff@echelon.ca) is a computer technician for Inly Systems and a member of OCLUG. When not at work, Dean enjoys spending time with his wife and two daughters and playing with his aquarium.

Archive Index Issue Table of Contents

Advanced search

VariCAD Version 6.2-0.3

Bradley Willson

Issue #57, January 1999

For the price, it delivers a strong value; however, a learning curve is involved.

- Manufacturer: VariCAD
- E-mail: mail@varicad.com
- URL: http://www.varicad.com/
- Price: VariCAD for Linux: $299 download via Internet$499 with support via Internet (CD-ROM included)$199 education version
- Reviewer: Bradley Willson

VariCAD is a fast, compact and economical mechanical CAD package featuring true 3-D modeling, solids and spatial analysis, 2-D to 3-D and 3-D to 2-D projection and extrusion, parametric symbol and mechanical part libraries and surface development utilities. It can be installed as a stand-alone application or on a server and networked in peer-to-peer and client-server configurations. The Linux version offers additional file security with **chown**. For the price, it delivers a strong value; however, a learning curve is involved.

Knobs, Levers and Switches

At the interface level, VariCAD resembles Microstation95 with a multitude of icons and sub-menus in both static display and windowed groups. The top menu bar is generic, with pull-down and side menus. Most of the available functions are presented in both the icons and menu bars. While you work, the menu choices and icons change functionality relative to your working environment. Menus, sub-menus, icons and sub-icons are easily customized by

editing separate text configuration files. In a networked environment, each user is able to customize his own menus and icons to his liking. The layout of the screen worked reasonably well for my work style. I was inclined to use the menu bar more at the beginning and the icons later on, once I better understood the meanings of the icon symbols.

The icons are arranged in a column matrix, loosely separated into two sections with active functions at the top and function group selection icons at the bottom. Changing the group of active functions is a simple matter of selecting one of the function group icons. Some of the function group icons also launch floating icon windows. I found the tool tips to be a great help in deciphering the icon glyphs.

The command line at the bottom of the window reminded me of AutoCAD, but that is the only resemblance. The acronyms varied from those I knew in AutoCAD. Selecting a menu item or icon causes the associated acronym to be displayed on the command line. Commands can also be entered manually using acronyms. Additional parameters are either entered as single letter, digit or symbol options, or by interactive selection of drawing and solid objects.

Figure 1. Machined Plate 2-D View

Figure 2. Machined Plate 3-D View

File, Open...

Example files are always a plus when working with a new program and VariCAD ships with several well-crafted samples of the program features. From those files, I was able to get a rough idea of performance and capabilities. Both 2-D and 3-D files loaded quickly on my 200MHz machine with the 3-D samples taking slightly longer to load, depending on the number of elements involved. Overall, I was satisfied with the graphic display quality, but VariCAD lacks the ability to export a rendered image to a standard graphics file format. To send a concept picture by e-mail, a separate utility must be used to create the snapshot file. As far as printed quality is concerned, VariCAD scores excellent. The lines are crisp and the text is clear, even on a dot-matrix printout. Since your output may vary, VariCAD includes all of the popular drawing size formats in both metric and ANSI.

Figure 3. Another Example in 2-D View

Figure 4. Another Example in 3-D View

File, New…

Working with the examples was easy, but getting started with a blank slate was another matter. As each new experience has a learning curve, I expected some level of difficulty in getting started, but I found the VariCAD learning curve to be closer to that of CATIA than to Microstation95 or AutoCAD. As a member of that group of people who jumps right into working with software without reading the manual, I ran into some problems with symbology and default settings. The default sizes of the icons were too small for my old monitor to show clearly. Once I learned how to enlarge them, the symbology became easier to decipher.

Figure 5. On-line Manual

The on-line manual was instrumental in helping me learn to modify the icon characteristics and other tasks during the review process. Pop-up tool tips were also quite helpful, but I would still like to see VariCAD offer a printed quick reference guide for the icon graphics. I am also one of those people who likes a printed page for reference, so I was happy to discover the on-line manual can be spooled to a printer or text file. I also found the search engine left something to be desired in that it did not effectively return results on multiple keyword searches. Overall, the manual appears to be complete, with a few translation, grammar and spelling errors throughout the document.

At first, I found working in VariCAD's 3-D environment to be awkward and unforgiving. The error messages that resulted from missed picks were annoying. Even CATIA will ignore missed picks and allow the user to try again without interrogation. I like the mouse-driven zoom and pan features, which make working on a large area easy. I missed the presence of an axis displayed at 0,0,0 with the vectors labeled X,Y and Z. The reliance on colors for identification of the axis vectors will make using the product difficult for color-blind users.

VariCAD recommends constructing the geometry for the solids first, then projecting the finished product into the 2-D drawing. This approach makes sense because the projected geometry does not automatically update when the parent solid is modified. Overall, the solids module is comprehensive and includes features such as moment of inertia, interference and material properties analysis in addition to solid merging, subtraction and compound surface manipulation features. I liked the “Bill Of Material” feature, because it makes it possible to specify the materials and parts “on the fly”.

VariCAD comes with extensive parametric part and symbol libraries. If you need a 2-inch long 1/4-20 bolt included in your design, it is a simple matter of defining the parameters in a dialog box and then placing the resulting part in either 2-D or 3-D modes. The modular design philosophy behind the

“Assembly” feature is becoming more popular, so VariCAD is right in stride with the trend. This feature gives you the ability to design individual components in small separate files and then assemble them in a “collector” drawing file.

Finally, VariCAD features DXF and IGES file format import and export capabilities for easy communication between other popular CAD packages.

In Conclusion

VariCAD is powerful and efficient with a long history of development. VariCAD celebrated its tenth anniversary in 1998. The people at VariCAD have elevated rapid development to a science. In the past six months, they have released three versions of VariCAD, and by the time this review is printed, they will have released a few more. October 1998 was the most recent release date, with major improvements in the area of 3-D to 2-D associativity as well as other enhancements. VariCAD is a work in progress and continues to improve with each new release.

Credits

Bradley J. Willson currently designs and troubleshoots tooling for the Boeing 777 program and fills the chair of chief cook and bottle washer for Willson Consulting Services. His friends understand and forgive his addiction to computer technology, while others wonder how he can stand the countless hours he spends staring at screens. According to Bradley, the secret is attitude—and maybe a mild case of radiation sickness. He can be reached via e-mail at cpu@ifixcomputers.com and http://www.ifixcomputers.com/.

Archive Index Issue Table of Contents

Advanced search

SciTech Display Doctor 1.0

James Youngman

Issue #57, January 1999

Display Doctor 1.0 for Linux achieves a goal even more useful than the DOS product; it uses SciTech's driver technology to provide drivers for video cards not supported by XFree86.

- Manufacturer: SciTech, Inc.
- E-mail: info@scitechsoft.com
- URL: http://www.scitechsoft.com/
- Price: $39.95 US
- Reviewer: James Youngman

SciTech has been selling their product for DOS and Windows for some time now. It is used mainly for installing VESA display services for graphics adapters. This is not an issue for Linux, since Linux programs never need to call the video BIOS.

Display Doctor 1.0 for Linux achieves a goal even more useful than the DOS product; it uses SciTech's driver technology to provide drivers for video cards not supported by XFree86. This includes cards from those manufacturers who will not release the information required in order to allow XFree86 drivers to be written.

This review will cover the Preview Release of SciTech Display Doctor 1.0 for Linux. You can download the trial version from SciTech's web site (http://www.scitechsoft.com/).

Display Doctor can be installed on any glibc2 system which already has XFree86 (3.3.2), GPM, and Tcl/Tk (7.4 or higher) installed. These requirements match one of the machines on which I run Linux. The preview is available as both a compressed tar file and as an RPM package. I chose to install the RPM package, since I was reviewing Display Doctor on a Red Hat 5.1 system.

You can install the package with **rpm -i**, but you must do this from a virtual console as opposed to a TELNET session, xterm or serial console. After the files are installed, the post-install script proceeds to set up the program interactively. This latter aspect may come as a surprise to those who are used to installing packages with RPM.

The first problem I had was that I needed to tell the install program which mouse protocol to use, even though this information was in a configuration file (an existing XF86Config and the file /etc/sysconfig/mouse on my Red Hat Linux system). The installation program immediately provided a dialog box for the purpose of fixing this, which was quite easy. After doing that, the rest of the configuration process can be navigated with the mouse.

The setup program detected the Matrox Millennium card I installed, but I had to give it the horizontal and vertical retrace specifications of my monitor (this may be because my monitor is four years old). I also have a UK-layout Microsoft Natural keyboard, which some X-server setup programs don't acknowledge, but to my surprise this went without a hitch and I could type the British pound symbol quite happily.

The final step in the configuration process is selecting the screen modes to be available for the X server. At the start of the selection process, every possible screen mode is pre-selected for you. When I started the X server for real, with **startx**, I was presented with a 1920x1080 screen mode, which unfortunately offers an aspect ratio of 16:9 and looked peculiar on my conventional 4:3 monitor. Exiting X, I removed this mode from the **ModeLine** list in the configuration file (which, oddly, is still called XF86Config). Restarting X, I was presented with a 1600x1200 screen mode—the same aspect ratio as my monitor.

In use, the SciTech Display Doctor X server worked well. From an installation point of view, it is just another X-server binary (/usr/X11R6/bin/XF86_SDD) and doesn't interfere in any way with the regular XFree86 files. I found this to be of

immense benefit and in marked contrast to the MetroX and AcceleratedX products.

The server implementation appears strikingly similar to the XFree86 server; the same set of extensions are provided and the same configuration file name and format is used (with driver name “scitech” instead of “svga” or “accel”). This is a handicap, since it prevents easy changing between X servers (XFree86 bails out when it sees the unknown driver name “scitech”). If you query the Display Doctor X server with **xdpyinfo**, it appears to be version 3.3.2 of the XFree86 server, which is quite confusing. I assume this particular oddity will disappear with the official release of the final product.

Since this is the preview release, benchmarks are not very illuminating. I look forward to benchmarking the full release of Display Doctor against the next full release of XFree86.

One thing I find most exciting about this technology is that it is just as applicable to Linux as it is to Windows. Even the same executables are used (Display Doctor installs a bunch of Windows-format 32-bit PE dynamic-link libraries in /usr/lib/nucleus). This is of interest to Linux fans, because SciTech is planning to use this same technology to bring the same level of support to KGI/GGI, SVGAlib and the forthcoming frame-buffer support currently in the Linux 2.1 kernel. According to the readme.txt file, Mesa may even benefit from the same treatment.

Unfortunately, I cannot say how Display Doctor made it possible for me to use X on a graphics card that XFree86 does not support. While it would have been nice to try this, the fact of the matter is that I have deliberately bought only hardware supported by Linux. If all Linux users did this, the market for SciTech's Display Doctor would be constricted; on the other hand, some of the things they are planning will advance the scope of the product beyond just X. I think Display Doctor is an interesting option for a bundling deal, particularly if SciTech follows through with their planned feature list.

James Youngman has recently changed jobs from VG Gas Analysis Systems to Logica. He has no significant hobbies apart from drinking real ale (see http://www.camra.org.uk/). James has been (very!) happily married to Erica for just over a year now. James claims to be intending to take up sailing again. You can reach him at jay@gnu.org.

Archive Index Issue Table of Contents

Advanced search

PartitionMagic 4.0

Roderick Smith

Issue #57, January 1999

PartitionMagic 4.0 contains an impressive array of features for partition management.

- Manufacturer: PowerQuest
- E-mail: magic@powerquest.com
- URL: http://www.powerquest.com
- Price: $69.95 ($29.95 upgrade)
- Reviewer: Roderick Smith

PowerQuest was the first company to develop a popular commercial product to modify a computer's partitions non-destructively. Developed originally for OS/2 and then ported to DOS and MS Windows, previous versions of PartitionMagic have been able to create, destroy, resize and copy FAT-16, FAT-32, HPFS and NTFS partitions, as well as convert from FAT-16 to any of the other formats. Needless to say, this has been a great boon to those with multiple operating systems, since it frees them from the tyranny of fixed partition sizes, allowing changes to disk resource allocation without the need to back up to tape, re-partition and restore. Up to version 3.05, however, PartitionMagic included no Linux-specific support. We could modify file systems for DOS, MS Windows and OS/2, but if we needed to modify Linux file systems, we had to do the backup/re-partition/restore dance or use various other workarounds, such as creating new partitions for spillover from existing partitions. PartitionMagic 4.0

promises to change this, with support for Linux EXT2 and Linux swap partition types.

Program Features

PartitionMagic 4.0 contains an impressive array of features for partition management. These features are included:

- The ability to create and delete partitions: when created, partitions can be formatted for FAT-16, FAT-32, HPFS, NTFS, EXT2 or Linux swap. Partition table entries will automatically be marked with the appropriate file system type.
- The ability to move and resize partitions: this feature applies to all supported partition types, including Linux swap. For FAT, allocation block sizes are adjusted automatically as appropriate. Unlike previous versions of PartitionMagic, 4.0's move and resize features are integrated into one. Partitions can be moved and resized in one user-interface operation and they can be resized towards or away from the beginning of the drive as well as the end. Such operations may result in two or more actual actions, however.
- Partition copy: this copies a partition from one location to another, including across physical disks, and works with all supported partition types. This makes it possible to replace a hard drive easily by copying multiple operating systems to a new drive with a single program. The copies are fast, too; I timed a copy of my Linux /home directory from one drive to another at 3:40 in PartitionMagic, vs. 5:38 using a tar pipe in Linux. The downside is that the copy seems to be doing a semi-raw copy followed by (if necessary) a resize, meaning that fragmentation is maintained on the copy.
- The Boot Magic boot loader: this utility manages the booting of multiple operating systems. In principle, it is much like LILO; when installed to a partition, it resides on a FAT partition and re-directs the boot process to another partition. It is graphical in nature, however, so may be more user friendly in operation than LILO. Boot Magic, like LILO but unlike OS/2's Boot Manager, does not require the allocation of its own partition.
- The ability to manage (but not create) IBM Boot Manager partitions: this may be of interest to OS/2 users and those who installed Boot Manager using PartitionMagic 3.0x.
- File system conversion: FAT-16 file systems can be converted to FAT-32, HPFS or NTFS (but not to EXT2). FAT-32 partitions can be converted to FAT-16 partitions.
- Drive integrity checks: this performs operations similar to those done by Linux's **e2fsck** or MS Windows' CHKDSK or SCANDISK.

- Operation from DOS or MS Windows: the OS/2 native executable has been dropped, and there is no native Linux version. Linux users can create a working system from Linux alone, however, as described below.
- Batch mode operation: a series of partition changes can be set up which PartitionMagic will then run in sequence. This allows you to do other things while your computer re-configures itself unattended. In case of an error, the batch execution will halt.
- “Wizards” to advise and help guide the user through processes such as balancing free disk space, adding a new operating system, etc.

In addition, PartitionMagic has a handful of other features that are of interest primarily to DOS or MS Windows users. For example, it has the ability to move applications from one FAT partition to another and diagnostics to help determine the best possible partition sizes for optimum FAT allocation block use.

PartitionMagic is available for download directly from PowerQuest's web site (via an on-line secure order form that asks for a credit card number) and as a retail package. I purchased my copy via the web site, using my existing 2.02 license number to obtain the upgrade pricing. The entire download is on the order of 50MB, so you will need several hours and/or a fast net connection if you choose to get it in this way. I received ample documentation in Adobe PDF format.

The program is definitely geared towards MS Windows users: when burned to CD, the files create a CD that includes an auto-install program when inserted into a MS Windows machine. Some of the more esoteric features seem to be present only in the MS Windows version of the program, but the one of greatest potential interest to Linux users, Boot Magic, requires a FAT partition to operate in the first place, so this is not a major loss. The installation files include a directory called “linux” which contains two disk images that can be copied to blank 1.44MB floppies. The first of these is a bootable OpenDOS 7.01 disk image with the main DOS PartitionMagic 4.0 executable and a few support files, including a Microsoft mouse driver. The second image includes help files for accessing PartitionMagic's help system from the first disk. Thus, those with Linux and no other OS can still use PartitionMagic by booting this floppy. It can, of course, be customized with other mouse drivers or features. Some of my testing, described below, was done with this DOS disk and some of it was done with MS Windows. Aside from the “Wizards”, the DOS and MS Windows versions of the program perform similarly.

PartitionMagic in Operation

I ran PartitionMagic through a series of tests on my system. I have a SCSI-based computer using a Symbios 53c860-based SCSI host adapter with a secondary adapter based on the Initio 9100UW chip. I ran tests with my SCSI hard disks attached to each of these adapters, with similar results both times. My hard disks include a 4GB Micropolis UltraSCSI, a 2GB Micropolis Fast SCSI-2 and a 2GB IBM UltraSCSI. Since one of the 2GB drives was used entirely for Linux swap space and temporary storage for CD-ROM creation, I was able to use it as a test-bed drive, copying file systems from the other two drives and modifying them with impunity. Most of my tests involved copying, moving and resizing EXT2 and HPFS partitions, although I also tried some operations on FAT-16 and FAT-32 partitions.

In operation, PartitionMagic 4.0 was relatively easy to use and worked as advertised—some of the time. Unfortunately, it produced errors of one sort or another on my system at least as often as it functioned correctly. Sometimes the errors would appear at the end of an operation and not have any apparent ill effect. A few times they would appear at the beginning of an operation, causing it to abort. Other times, particularly with copy operations, errors would occur at the end of an operation and abort it. I also encountered file system corruption on some tests, particularly tests involving HPFS. The one time I encountered ext2fs corruption, e2fsck was able to correct the problems.

For a number of reasons, I am certain that my physical hard drive system is not to blame for these problems. First, I did a low-level format on the IBM drive I was using as a test bed, so I am certain it was free from physical defects. Second, PartitionMagic 4.0 reported no errors on the drive when it was configured to check for them prior to each operation. Third, problems occurred on both the IBM and the Micropolis 2GB when I reconfigured my system to use it for the tests. Fourth, problems occurred with both the Symbios 53c860 board and the Initio board as the host for the hard disks. Finally, problems did not occur when I tried the same operations with PartitionMagic 2.02 (when they were possible with that version of the program).

Thus, I am quite certain that PartitionMagic 4.0 has some serious bugs. These bugs are so severe and so obvious that I find it hard to believe a reputable company would ship a product knowing these bugs existed. Therefore, I must conclude that PowerQuest did not know they existed (although I have now filed extensive bug reports with them) and that something about my system was turning up bugs in the software. Perhaps the peculiar drive geometry on my first physical disk (1018 cylinders, 133 heads, 62 cylinders), produced by the Symbios host adapter, has something to do with it; or maybe PowerQuest simply did minimal testing with SCSI systems; or perhaps their testing used

relatively simple partitioning. My drives each have at least two partitions and problems tended to turn up more frequently with three or more partitions.

In addition to the out-and-out primary function bugs, Partition Magic has a number of limitations, some of which would not even occur to many people except that the program really can do so much. Specifically:

- The Boot Magic program had problems booting a FreeBSD partition on my system.
- The program has problems with some Linux-created partition tables. Specifically, and according to the PartitionMagic documentation, Linux's fdisk creates extended partitions with logical partitions listed in order of creation, which may not be the same as the order of the partitions themselves (e.g., sda6 may appear before sda5). PartitionMagic 4.0 will choke on such partition tables, though version 2.02 doesn't seem to have any problem with them.
- There appears to be no way to merge the contents of two partitions; if you have, say, /usr and /usr/X11R6 on two partitions on one drive, you can't turn them into one partition from within PartitionMagic (although you can increase the size of /usr, reboot to Linux, copy /usr/X11R6 to the enlarged /usr, reboot to PartitionMagic, delete /usr/X11R6 and increase the size of /usr again).
- Re-sizing or moving a Linux boot partition will render the partition unbootable (assuming LILO is being used to load a kernel image off that partition). Linux users would do well to ensure that they have a floppy with a kernel image, and also a full-fledged Linux boot system with a text editor to handle any necessary changes in /etc/fstab after moving, adding or deleting partitions.
- The batch nature of the operations can result in some truly brainless series of operations. If you enter a partition resize operation, then decide you want to perform a different resize on the same partition, both will most likely be executed, although one would do. Fortunately, you can clear the entire queue of changes; however, you can't delete a single operation.

Recommendations

The above may appear to paint a rather bleak picture—to some extent, this is justified. The types of operations performed by PartitionMagic are inherently dangerous and bugs in such a program are a serious matter. On the other hand, my suspicion that the bugs manifest due to something specific about my system may mean that others may have better luck. PowerQuest also has a good reputation for producing reliable software, so I have high hopes that they will correct these problems. Assuming this happens, PartitionMagic 4.0x will be

an excellent program and a must-have utility for anyone managing multiple operating systems on one computer. The $69.95 price may seem a bit high, but if you have ever spent most of a day juggling partitions around using tape backups, removable disks or some similar mechanism, you'll recognize the appeal of being able to do that quickly and on the fly. If you own a previous version of PartitionMagic, the $29.95 upgrade price represents a true savings for any Linux user, since it radically improves on the program's utility for the Linux community.

The best possible way to use PartitionMagic seems to be as an exclusive means of managing partitions on a drive. Because of incompatibilities such as the one mentioned above with Linux's fdisk, I recommend using PartitionMagic to create all the new partitions on a disk. If you find yourself with a disk that PartitionMagic won't handle because of Linux-created logical partitions, you may be able to use Linux's fdisk to delete the offending partitions and recreate them in the correct order. If you're careful to create new partitions of precisely the correct size, your partitions will still be usable, but I strongly recommend backing them up before attempting such an operation.

If version 4.01 is not available by the time you read this, I recommend waiting for it unless you are in immediate need of Partition Magic's abilities. If you must use version 4.0, use it cautiously: back up all data before changing a partition, then run e2fsck, CHKDSK, or SCANDISK on any modified partitions immediately thereafter. (You may need to specify the **-f** option to e2fsck to be sure it runs on the partition.) A spot check of the integrity of the data after a modification would also be a good idea. Of course, these suggestions also apply to any program that does low-level operations on a hard disk, but given the problems I encountered with PartitionMagic 4.0, they apply even more strongly to it.

Roderick Smith can be reached via e-mail at rodsmith@fast.net.

Archive Index Issue Table of Contents

Advanced search

Calendar Programs

Michael Stutz

Issue #57, January 1999

Mr. Stutz introduces us to a digital method for keeping track of appointments and those important dates in our lives.

I am obsessive about keeping track of things. I used to get a free calendar each year from the local phone company. It was the perfect interface for all my appointments—a slim, spiral-bound calendar that I kept on my desktop and used to record any upcoming appointments, as well as mark the birthdays of my friends and family. Every year around the holidays I'd get a new calendar and dutifully re-transcribe the birthdays and other annual appointments into the new calendar.

A few years ago, the phone company decided to stop sending out these free calendars, but fortunately I had found a better way. By using **cal** and **calendar** programs, I can leave all my old analog date books behind.

Different versions of these utilities exist. Originally part of the BSD utilities, a GNU version called **gcal** with some advanced features has been released. I will focus this tutorial on the original BSD programs since they're widely available, as well as mention some other related programs at the end of the article.

You'll first want to find out if you have these programs installed on your system by typing:

```
which cal
```

If **which** returns a full pathname of the **cal** program, it is installed. If instead you are immediately returned to a shell prompt, then you will need to obtain and install this program. You can use **which** again to see if you have the **calendar** program installed as well; they are different programs and you will need them both.

If you use the Debian distribution of Linux, both programs are available in the **bsdmainutils** package. Otherwise, consult your distribution, or search the Linux Software Map at http://www.linuxresources.com/apps.html.

Using Calendar

calendar is a basic reminder service. It reads a file called calendar in the current directory and prints lines which start with today's or tomorrow's dates. The calendar file is a text file that can be created and edited with any text editor.

This program is more powerful than it may seem at first. The general format for a calendar file entry is the month and day to the immediate left, followed by a tab and the reminder text. The month and day can be skipped—each line beginning with a tab carries the same month and day value as the line preceding it.

There is quite a varied syntax for the month and day. To demonstrate, let's make up a sample calendar file and look at it line by line:

```
10/31   Johnny's Halloween party
Friday  Garbage day
Nov. 20 Dentist appointment, 9:30am
20/11   Mandatory staff meeting, 10:00am
January Happy new year!
        Have you made your resolutions yet?
18 *    Rent's due
```

Let's imagine that today is Friday, October 31, 1997. At the shell prompt, type:

```
calendar
```

With the sample calendar file in your current directory, you will see this output:

```
10/31   Johnny's Halloween party
Friday  Garbage day
```

The **10/31** in the first line tells calendar to print that line if it is October 31 (or the day before), and the **Friday** in the second line tells it to print (you guessed it) every Friday.

As you can see, you can also use an abbreviated form of writing out the month, as in the example line starting with **Nov. 20**. Now, on the fourth line, the mm/dd format is reversed—in this case, calendar figures that out and will print it on the 20th day of November. It's probably not a good idea to keep records in this format. If you were to write May 10th as 10/5, calendar will assume you mean the U.S. convention of mm/dd and print that entry on October 5th.

If you have just the name of a month in the first column, calendar will print the entry on the first of that month. If you eliminate that column altogether by inserting a tab and some reminder text, calendar will print the line on the date

of the preceding entry. Thus, on New Year's Day, our example calendar will output:

```
January Happy new year!
        Have you made your resolutions yet?
```

Finally, substituting an asterisk for the month will print that entry on the stated date each month. In our example, a reminder to pay the rent will be displayed on the eighteenth of each month.

Keeping Your Dates Separate

Since it is pre-processed by **cpp**, the C preprocessor, calendar recognizes include files. This allows you to keep and use special lists, such as a personal list of birthdays, without cluttering up your main calendar file. The calendar program comes with a set of such files in /usr/lib/calendar/ to get you started:

- **calendar.birthday**: birth and death dates of many famous historical figures
- **calendar.computer**: important dates in the history of computing
- **calendar.holiday**: known (and not-so-well-known) holidays
- **calendar.music**: important dates in music, especially mainstream rock-n-roll
- **calendar.christian**: Christian holidays
- **calendar.history**: many historical events
- **calendar.judaic**: Jewish holidays
- **calendar.usholiday**: standard US holidays

When you use an include statement, calendar first searches the directory it was called in, and then looks in /usr/lib/calendar/. Including this line in your file,

```
#include <calendar.usholiday>
```

means calendar will first look in the current directory for such a file. If not present, it will then check /usr/lib/calendar/, and if the appropriate file is found, include it.

One of the functions of my old paper calendar was to record and keep a record of important events in my life. I was able to reproduce this function by keeping this information in files named calendar.yyyy, such as calendar.1997. Looking through the files in order gives me a chronological record of major events in my life, and if I ever wanted to see what I was doing around today's date in a certain year or years, I could add include statements for the appropriate calendar.yyyy files in my calendar file.

Automating Calendar

You can run calendar whenever you like, but it might be more useful to put it in your profile file (~/.bash_profile if you use the bash shell). Then calendar will run each time you log in to the system.

I keep my personal calendar file in the doc/etc/ subdirectory of my home directory, so I would include the following line in my profile:

```
cd /home/m/doc/etc/; calendar; cd
```

Putting the same line in your .bashrc file (again, only if you use the bash shell—others are different) also works to run calendar each time you start a shell.

Sometimes even *this* isn't enough—if your machine is on all the time and you haven't been starting any new shells or xterms, you might miss a reminder. So you could schedule a **cron** job to run calendar each day, e-mailing the output to you as a reminder.

Using cal

The **cal** program displays a text calendar. If you call it without any options by typing:

```
cal
```

the current month will be displayed on the terminal like this (assuming September, 1998):

```
   September 1998
 S  M Tu  W Th  F  S
       1  2  3  4  5
 6  7  8  9 10 11 12
13 14 15 16 17 18 19
20 21 22 23 24 25 26
27 28 29 30
```

It's pretty no-frills, but can work on any terminal. To get a calendar of the whole year, call cal with the year as argument.

```
cal 1950
```

Typing:

```
cal -y
```

will print a calendar for the current year.

cal can also display any arbitrary month. If you want to see the month of December 1999, for instance, use:

```
cal 12 1999
```

in which case you'll see:

```
   December 1999
    S  M Tu  W Th  F  S
             1  2  3  4
    5  6  7  8  9 10 11
   12 13 14 15 16 17 18
   19 20 21 22 23 24 25
   26 27 28 29 30 31
```

Other Calendar Programs

While cal is quite useful for creating simple calendars, sometimes when printing you might want a nicer output. **pcal** is a program that creates very nice calendars in PostScript. Its options are very similar to that of cal. It is available from http://garbo.uwasa.fi/unix/pcal.html.

Other programs that handle calendar functions in a more graphically-intense way (and can be used only in X) include **ical**, at http://www.research.digital.com/SRC/personal/Sanjay_Ghemawat/ical/home.html and **plan**, found at http://www.in-berlin.de/User/bitrot/plan.html. There are many variants of the basic UNIX calendar programs—check the Linux Software Map for more.

With this overview of the power and flexibility of these simple calendar programs, you too can leave your analog calendar systems behind.

Michael Stutz is a writer whose first novel, Sunclipse, is freely distributed under GNU GPL copyleft—just like Linux—and is on the Web at http://dsl.org/m/doc/lit/sunclipse.html. He can be reached via e-mail at stutz@dsl.org.

Archive Index Issue Table of Contents

Advanced search

Linux as a PACS Server for Nuclear Medicine

Cheng-Ta Wu

Issue #57, January 1999

Linux is being used in a Taiwan hospital as a server for medical images and as a firewall.

Many Linux users are not likely to be familiar with nuclear medicine, but it plays a major part in the medical field today. I am a physician and will describe my experience with Linux in the Nuclear Medicine Department of Chang Gung Memorial Hospital, Kaohsiung, Taiwan. I hope the information will be useful for novice Linux users (I was a novice two and a half years ago).

Before describing my Linux experience in nuclear medicine, I have to mention the background of medical-image standards including PACS, Interfile and DICOM. No standard existed among the image formats of different CT (computerized tomography), MR (magnetic resonance) and gamma camera vendors before 1985. After recognizing the need for standards to facilitate multi-vendor connectivity and PACS (Picture Archiving and Communication System), the American College of Radiologists (ACR) and the National Electrical Manufacturers Association (NEMA) proposed the first standard ACR/NEMA 1.0 in 1985, ACR/NEMA 2.0 in 1988 and then ACR/NEMA 3.0 (well-known as DICOM 3.0).

DICOM 3.0 is the current standard, and almost all vendors implement DICOM 3.0 in their new product lines, although many other old image instruments are still not DICOM 3.0 compatible. Another standard format in the nuclear medicine field is Interfile. Like DICOM 3.0, Interfile is a specified file format, but, unlike DICOM 3.0, Interfile is not a communication protocol. The network protocol of DICOM 3.0 is built on top of TCP/IP, so DICOM clients can query/retrieve image data from a DICOM server and can also store image data on the server through the Internet if properly connected. Interfile is simply an interim file format of nuclear medicine. If nothing else, it proposed a standard for gamma camera vendors to follow, but what about the future? DICOM is the best way to go. If you have an interest in medical image formats, please refer to

the Medical Image Format FAQ (http://idt.net/~dclunie/medical-image-faq/html/index.html), maintained by David Clunie, M.D., for further information.

In the Diagnostic Nuclear Medicine department, after intravenous injection or oral ingestion of radionuclides, patients are put under a gamma camera to have pictures taken. A gamma camera picks up gamma-rays emitted from the radionuclides to make up the images. There are many gamma camera vendors in the market. The following is a list of them and the operating systems they use:

- Siemens: MacOS 7.x
- Elscint: OS/2
- GE, Picker, SMV, Toshiba and others: UNIX
- Other: Proprietary OS (Some old gamma cameras such as GE Starcam, old Toshiba, et al.)

The network protocol for Macintosh is AppleTalk, for OS/2 it is NetBIOS and for UNIX it is TCP/IP. Due to the prevalence of the Internet, all operating systems used today also support TCP/IP. Network protocols for computers are used for communication in the same way language is used by people. Under the same network protocol, different computers can exchange data with one another.

Chung Gung Memorial Hospital, Kaohsiung, Taiwan, is a medical center with more than 1,000 beds. One IP address is assigned to the Nuclear Medicine Department. We have three gamma cameras. The oldest one is the GE STARCAM which uses a proprietary OS, one Siemens ICON triple-head gamma camera and one Elscint Varicam dual-head gamma camera. The Elscint Varicam is a brand new gamma camera, installed before I left the hospital.

After learning about Linux two and a half years ago, I realized it would be the perfect server for gamma cameras. Since Linux is a UNIX clone, it offers TCP/IP networking. In addition, Netatalk (AppleTalk protocol for UNIX) allows Linux directories to be mounted by Macintosh, and SAMBA (SMB is NetBIOS over TCP/IP) allows OS/2, Windows 95 and Windows NT to use the services at Linux and vice versa. TCP/IP, AppleTalk and SMB (server message block) are all available in Linux; thus, Linux can communicate with all gamma camera computers except some old and proprietary ones. I will describe later how to solve the problem of retrieving the image data from proprietary computer operating systems.

At the time I set up Linux as a PACS server for our Nuclear Medicine department, our Siemens ICON could export Interfile image files but didn't offer the DICOM function. The Elscint Varicam offered DICOM functions and the GE Starcam offered neither Interfile nor DICOM. Using the image translation

software GAMMACON from MITA, the problems of the proprietary image file data were solved.

GAMMACON is a program that runs under the MS-DOS environment. It reads and writes different proprietary, Interfile and DICOM image files to and from 8-inch, 5.25-inch or 3.5-inch diskettes and the network or hard disk. GAMMACON uses a security hardware key attached to the PC printer port to prevent software piracy. After modifying the configuration file of the Linux DOS Emulator, GAMMACON runs smoothly under Linux. Combined with the network capacities of Linux, we could translate the GE Starcam image files archived on 3.5-inch floppy disks into Interfile, and then process them on the Siemens ICON or the Elscint Varicam. Due to the limitations of GAMMACON, only one program could be run at a time. The security hardware key is locked by GAMMACON. After the needed programs are installed, Siemens ICON could easily mount the Linux shared directories from the CHOOSER via Netatalk. Elscint Varicam OS/2 could also mount the Linux shared directories as a network disk (via SAMBA). For other gamma cameras that use UNIX, such as Picker or ADAC, mounting the directories via NFS is routine work.

To read the GE Starcam files off-line, I could bring the floppy disks from the GE Starcam to Linux, then use GAMMACON to translate the GE Starcam files into Interfile, DICOM or GIF format. It sounds perfect. Even for gamma cameras that are not DICOM compatible, we still offer a convenient way to solve the image file exchanges in nuclear medicine by using the specific functions of the software from different vendors to process the images. Before I left Chung Gung Memorial Hospital last year, I worked on the DICOM Query/Retrieve (DICOM client) function on Linux—a hard job for me, since I have no programming background. The Swansea University in Wales offers the DICOM server service on Linux; perhaps others do too.

Finished? No, I mentioned earlier that one IP address is assigned to our Nuclear Medicine department. I installed two NE2000 compatible cards in this Linux PC PACS server and recompiled the Linux kernel to enable the IP masquerade function. One network card is used with the IP address we were assigned and the other network card is assigned to an internal network address. All the gamma camera computers, the Macintoshes and the Windows 95 PCs in our office belong to the internal network and can access the Internet seamlessly through the Linux IP masquerade. The Linux IP masquerade forms a firewall to prevent invasion from the Internet. The limitation of one IP address is no longer present.

Combined with the Apache WWW server and mSQL, we use the GIF-format nuclear medicine images converted from GAMMACON to make all the

diagnostic reports available as HTML documents available to the registered medical doctors.

Hylafax is a free fax server for Linux, and it also supports many other operating systems such as Macintosh and MS Windows. Several colleagues of mine are quite happy with Hylafax, since they can easily send a fax from their PCs. You may wonder why I didn't mention DNS or a mail server. Our hospital set a proxy server to allow only the WWW browser to access the external world and prevent hacker destruction, so we must exclusively use the mail server offered by our hospital. They set the proxy and mail server to the same SUN Ultra SPARC.

Is Linux difficult? No. Take me as an example. I was a total computer newbie two and a half years ago, and yet I used Linux in my daily work. I am sure others could do the same. I would like to thank all of the people who devote their efforts to Linux and free software.

The author, **Cheng-Ta Wu** was a nuclear physician and is now in general practice at Feng Shan, Kaohsiung County, Taiwan. He lives with his parents, two brothers and a dog. He is interested in traveling, music and swimming. If you come across him while he is traveling, please mention you are a Linux user. He can be reached via e-mail at chengtaw@mail.euphoria.com.tw.

Archive Index Issue Table of Contents

Advanced search

Caching the Web, Part 1

David Guerrero

Issue #57, January 1999

Improve your users' browsing and save your bandwidth by using proxy servers to cache web pages.

The web is everywhere. Everyone uses it. Everyone talks about it. But in this less-than-perfect world, you know there are problems. Bandwidth is a problem. Web document latency (the time a document takes to arrive at your browser once its URL is requested) is a problem. As more of your bandwidth space is used, latency of documents retrieved from the Internet increases. Bandwidth is expensive, perhaps the most expensive element of an Internet connection.

Despite the fact that the web is growing fast, the same documents get requested and the same web sites are visited repeatedly. We can take advantage of this to avoid downloading redundant objects. You would be surprised to learn how many of your users read the NBA.COM web pages, or how many times the GIFs from AltaVista cross your line.

Even if you know nothing about web caching, you are probably using it with your web browser. Most common browsers use this approach with the documents and objects you retrieve from the Web, keeping a copy of recent documents in memory or disk. Each time you click on the “back” button or visit the same page, that page is in memory and does not need to be retrieved. This is the first level of caching, and the technique can be expanded to the entire web.

The basic idea behind caching is to store the documents retrieved by one user in a common location, and thus avoid retrieving the same document for a second user from its source. Instead, the second user gets the document from the common place. This is very important when you deal with organizations in Europe, where most of the inbound traffic comes from the other side of the Atlantic, frequently across slow links.

The main benefit of this approach is the fact that your users' browsing is now collaborative, and an important number of the documents your users retrieve are served in a very small period of time. In a medium-sized organization (with between 50 to 100 users), you can serve up to 60% of URL requests from the local cache.

The difference between a browser cache and a proxy-cache server is that the browser cache works for only one user and is located in the final user workstation, while the proxy-cache server is a program that acts on behalf of a number of web browser clients, allowing one client to read documents requested by others earlier. This proxy-cache server is located in a common server that usually lies between the local network and the Internet. All browsers request documents from the proxy server, which retrieves the documents and returns them to the browsers. It's the second level of caching in an organization. Figure 1 shows this type of network configuration.

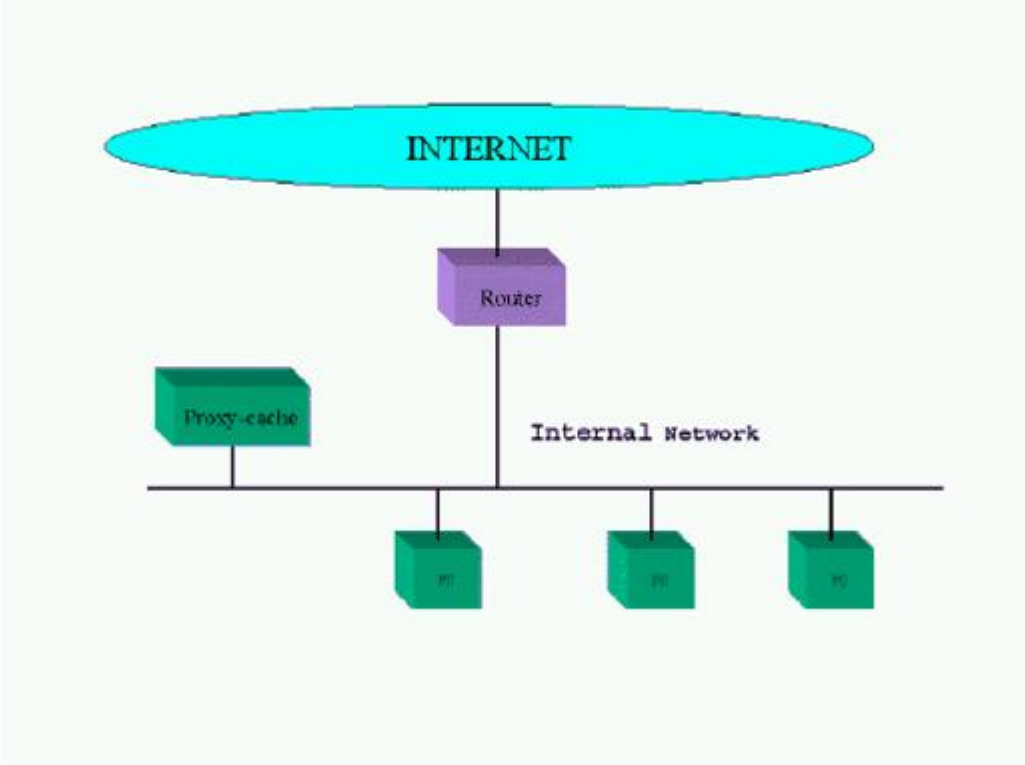


Figure 1. Proxy-cache Server Network Configuration

A proxy-cache is not just a solution to the bandwidth crisis; it is also desirable when a network firewall is needed to guarantee the security of your organization. In this case, the proxy-cache sits on a computer accessible from all local browsers, but isolates them from the Internet at the same time. This computer must have two network interfaces attached to the internal and external networks and must be the only computer reachable from the Internet. Figure 2 illustrates such a configuration. The proxy-cache server must be accessible only by internal systems to ensure that no one on the Internet can

access your internal documents by requesting them from the proxy-cache. I will discuss access control to the proxy-cache later in this article.

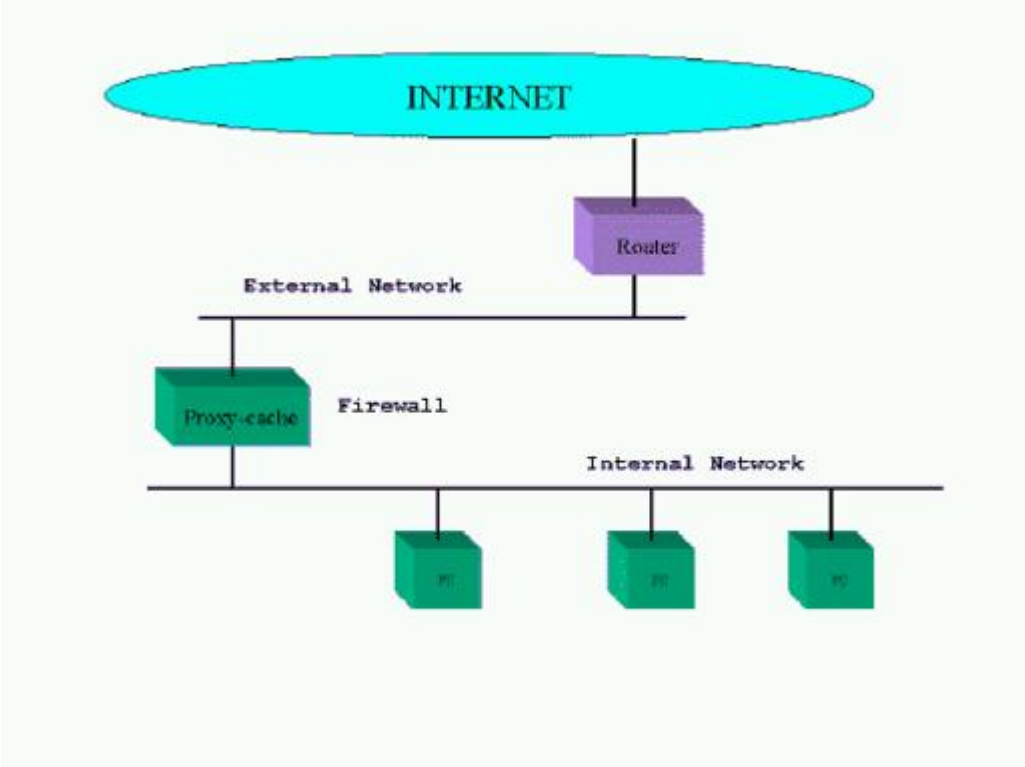


Figure 2. Proxy-cache Network Configuration with Firewall

Multi-Level Web Caching

One step forward from this approach is the concept of a cache hierarchy, where two or more proxy-cache servers cooperate by serving documents to each other. A proxy-cache can play two different roles in a hierarchy, depending on network topology, ISP policies and system resources. A neighbor (or sibling) cache is one that serves only documents it already has. A parent cache can get documents from another cache higher in the hierarchy or from the source, depending whether it has more parent or neighbor caches in its level. A parent cache should be used when there are no more opportunities to get the document from a cache on the same level.

Choosing a good cache topology is very important in order to avoid generating more network traffic than without web caching. An organization can choose to have several sibling caches in its departmental networks and a parent cache close to the network link to the Internet. This parent cache can be configured to request documents from another parent cache in the upstream ISP, in case they have one (most do). Agreements can be made between organizations and ISPs to build sibling or parent caches to reduce traffic overload in their links, or to route web traffic through a different path than the regular IP traffic. Web caching can be considered an application-level, routing mechanism, which uses

ICP (Internet Cache Protocol) as its main protocol. Figure 3 is an example of how an organization can implement multi-level web caching.

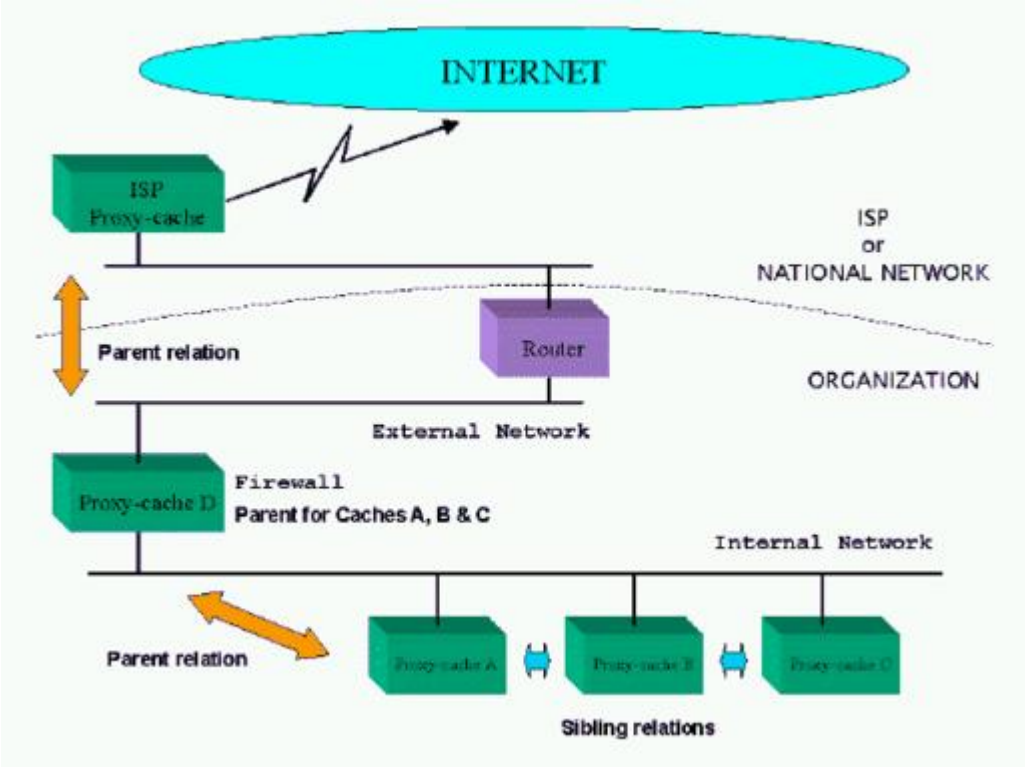


Figure 3. Multi-level Web Caching Organization

What's ICP?

ICP, Internet Cache Protocol, is a protocol used for communicating among web caches. A lightweight protocol built on top of UDP, ICP is used to locate specific web objects in neighboring caches. Most transfers of objects between caches are done with the TCP-based HTTP protocol, but making the decision of where to retrieve an object must be done with a simpler and faster mechanism. Other information needed is which caches are down or have congested links.

One cache, in order to find the best location from which to download an object, sends an ICP request packet to all of its siblings and parent caches, and they send back ICP replies with a HIT or MISS code. A HIT code means this cache has the object and agrees to serve it. A MISS code means it doesn't have the object. Thus, the cache now knows who has the object it needs, and, combining this information with other factors such as round-trip times of each response, can perform the cache selection and make the request via HTTP to its choice. If all the caches reply with MISS packets, it requests the document from its parent cache. An ICP request/reply exchange should occur in a second or two, so the latency increases this time for the browser, but this is usually not noticed by the end user.

If the object requested via ICP is small enough, it can be included in the ICP HIT reply, like an HTTP redirect, but this is not a very common situation. Of course, ICP is needed only in a multi-level cache environment with multiple siblings and parent caches. Using ICP is not necessary in situations like the ones in Figures 1 and 2. When only one cache is involved, or when one cache always requests documents from the same higher-level cache, ICP would only add unwanted overhead.

To cache or not to cache?

At this point, we must realize that not all objects in the web are cacheables. Most FTP files are, as well as most static web pages, but a large number of CGI-generated web pages (dynamic documents) are not. This kind of document is non-cacheable, because it is different each time you request it. Two good examples of this kind of object are access counters and live database queries. Caching a reply from a flight reservation system is senseless, since the next query will most likely return more up-to-date values. Other kinds of documents which should not be cached include SSL documents (securely transmitted documents).

OK, cache, but for how long?

Even if you do not have a proxy-cache server, you must be aware of the effects other proxy-cache servers are causing on the Internet. You may be publishing information on your web server that other caches are storing and serving for more time than you probably want. This is particularly true if you periodically update your site and it's important to you that a final user never gets out-of-date pages or graphics.

A document in a cache server can have three different states: FRESH, NORMAL and STALE. When an object is FRESH, it is served normally when a request for it arrives without checking the source to see if the object has been modified since its last retrieval. If it's in NORMAL state, an **If-Modified-Since GET** request is sent to the source, so the cache server downloads the object from the source only if it has changed since its last retrieval. A STALE document is no longer valid, and it's retrieved from the source again.

Normally, when a web server sends a document, it adds an HTTP header called Last-Modified containing the date the object was created or last modified. This data is used by cache servers to heuristically calculate how much time may pass for the object to still be considered FRESH. Usually, a proportion of the time elapsed between the date the document was last modified and the date when the document was received is used. A normal proportion is 10% to 30% of this time. If this proportion is set to 20%, a document modified 10 days ago will remain in the cache only two days before being checked for changes.

Webmasters who frequently update their information need more control over the time their documents remain unchecked in web caches. In this case, the Expires HTTP header in the documents served by your server can be used to indicate when this document must be dropped by any cache server. This header explicitly gives the caches the expiration date of a document. A valid RFC1123 time format should be used with this header, for example:

```
Expires: Mon, 25 Aug 1997 10:00:00 GMT
```

This header can be generated easily in CGI scripts or the mod_expires module included in Apache 1.2. For example, the following Apache directives (in a <Directory> </Directory> or a .htaccess) would do it:

```
ExpiresActive On
ExpiresByType image/gif A432000
ExpiresByType image/jpeg A432000
ExpiresByType text/html A10800
```

The Expires header is activated for all subsequent documents with a value of five days for JPEG and GIF images and three hours for HTML documents.

If you have documents which should never be cached in any server or browser, use the HTTP header called:

```
Pragma: no-cache
```

Of course, a cache may expire an object sooner, based on site configuration, lack of free disk space, LRU (less recently used) policies, etc., but it can never cache an object beyond its **Expires** time.

Next month, we will discuss Squid, the best free software solution for building proxy-cache servers.

Resources

David Guerrero is a system and network manager for the Spanish Ministry of Education and Culture and an independent consultant. He has been using Linux since the .98plNN days. When not working, he likes to spend time with his love Yolanda, travel, play guitar and synths, or go out with his “colegas”. He can be reached at david@boe.es.

Linux for Macintosh 68K Port

Alan Cox

Issue #57, January 1999

“I don't care if space aliens ate my mouse” or a case study in both the technical and human issues in porting the Linux OS to a new M68K target platform.

Several barriers to a Linux for Macintosh 68K port exist. The first is that Apple does not want other operating systems on its machines. While almost all of the workings of a PC can be learned from books, almost nothing is written about the Apple Macintosh. Sometimes Macintosh specifications and technical notes fill in the blanks; at other times, it is necessary to apply a great deal of guesswork and experimentation to figure out the hardware.

The second barrier is a human one. Most Macintosh machines were not sold to the technical market, and average Macintosh users aren't terribly interested in a “real operating system” for their computers. Nevertheless, a sizable technically oriented Macintosh user community does exist, with a lot of Macintosh hardware to go with it (probably more than any other non-Intel Linux platform). A further reason has been provided by Apple, whose quaint advice for owners of 68K machines now appears to be “buy a new computer”.

The third barrier to a Linux port is less obvious and is hidden by a lack of documentation. Certain folks have speculated that embarrassment is the main reason for Apple Computer releasing so little documentation. In general, Macintosh platforms have positively Stone Age design features. For example, the interrupt controllers on a Macintosh II are a pair of 6522 VIA chips, intended for use with the 8-bit 6502 processor. Bad hardware design makes for poor performance, unless carefully handled. The complete lack of DMA (direct memory access) is even less helpful. Apple seems to think no DMA is a feature on most machines and actually have a technical note stating “I used to be a teenage DMA junkie”, which seems to be a justification of their rather comical hardware design.

Getting Started

To get a port started, the first item needed is hardware. I had most of this (a 5MB MacII, cast off from the office as too slow for practical use). Initially, I felt safe helping to work out the directions for the Linux port, as this system lacked an MMU (memory management unit) and was therefore unable to run any proposed Linux port.

Rob Pelkey started on some very basic Linux work for the Macintosh, but needed a boot loader to load the Linux OS and kick it off. On #linux on the LinuxNet IRC network, Jes Sorensen (the keeper of Linux68K), I and several others got into a few discussions about the port and what would be required. After a lot of digging, we managed to gather some basic information on the Macintosh68K, then filled in further areas by investigating the excellent detective work the OpenBSD/Macintosh team had done in getting BSD limping along on Macintosh machines. Further information came to light from the Linux on OSF Mach port sponsored by Apple. We discovered that Apple continued to use the same 8-bit microcontrollers, or emulations of them, and had not redesigned the systems materially for the new processor.

Everything seemed perfectly fine. I had a Macintosh box to laugh at (and we used it occasionally to fail to duplicate problems Macintosh users had with CymruNet), we could kick ideas around and I had no MMU in my Macintosh, so I couldn't possibly help write any code.

By this point, Rob's effort had stalled badly, as he lacked the time to write the boot loader needed to run Linux and was working on passing courses and other sundry items. No worry—either someone would eventually take over the project, or he would finish his courses. Then Frank Neuman sent me an MMU for the MacII and someone else donated a pair of Ethernet cards—whoops, no more excuses.

Learning MacOS

Having fitted the MMU to the Macintosh without blowing it up, I tried to get MacOS to run with virtual memory. This is supposed to be simple—click on the memory tool and select 32-bit, virtual memory on. But no, my memory control didn't have a 32-bit option, let alone a virtual memory one. I stared a bit, then checked on a more modern Mac downstairs to be sure I had the right screen. The other Macintosh which was running the same MacOS version had the required option; mine didn't.

This was my first experience with the horrors of the Mac. While UNIX says “I'm sorry you can't do that”, MacOS has exactly two error messages. It either goes “eep?” or the setup box is simply not there until 12 other unidentified items

have been installed and three apparently unrelated dialog boxes have been completed. Mine was an error of the latter category.

Apple shipped the MacII with the ability to upgrade to include an MMU chip; therefore, they sensibly shipped it with a system ROM incapable of running with the MMU enabled. Brilliant—just don't design anything mission-critical, please. Fortunately, Apple had concealed on their web site a small tool which patches the ROM entry points so that it can run in 32-bit mode.

Okay, so all I had to do was download the tool, install it and be done—not so simple. To get the program, I needed the Ethernet to work. I ended up using **kermit** to transfer 700KB of Ethernet installer onto the Macintosh. After four hours of fighting with the completely alien Macintosh archiver tools, I had the machine talking AppleTalk shares to a Linux box using Netatalk, as well as insight into why Macintosh people meeting a PC for the first time look as if they'd just discovered alien life forms.

An hour later, I had figured out how to unpack Macbin files and the Macintosh was in 32-bit mode and admitted the MMU was present and functional.

Building and Booting Linux

The next stage in the operation was to figure out how to boot a Linux kernel image on the Macintosh. NetBSD and OpenBSD use a boot loader which loads a.out format executables into the memory of the Macintosh, shuts the Macintosh down, moves it to address 0 and jumps to it. I quickly decided I didn't want to write a boot loader. The OpenBSD loader was almost pure MacOS wizardry at a level far beyond my abilities. Not to worry—it soon became apparent that the OpenBSD loader could be persuaded to load Linux too. A true loader could wait.

The next problem was building a Linux kernel image that would link and (while most likely not doing anything useful) at least serve as something to feed the OpenBSD booter. Linux is built using the GNU tool chain, which supports the building of cross compilers. It is thus possible to compile and build 680x0 binaries on an ordinary Intel-based PC. It took a couple of builds to get **gcc** and the GNU binutils generating almost the right code. Linux a.out executables have a two-byte header different from the OpenBSD ones, and the OpenBSD boot loader checked those bytes. Rather than rebuild the entire tool chain again, I wrote a simple tool to fix the headers.

Most of Linux/M68K was quite content to build for a Macintosh target. I filled in everything that complained with dummy routines—for Mac keyboards, mice, display, etc. until it all compiled. Because of the well-designed abstraction layers in the Linux/M68K kernel, this was quite easy to do. I now had a

completely useless, do-nothing, Macintosh kernel that the OpenBSD loader would load and which then promptly crashed the machine as I expected.

The Linux/M68K project had faced up to the challenges of supporting multiple types of 680x0-based computers within the same port, well before I got involved. As a result of the need to support both the Amiga and Atari systems, clear layers of abstractions are present. Adding an additional M68K target consists mostly of filling in platform-specific blank fields. A port to a completely new processor would have been far more challenging than this one.

For the Macintosh case, I filled in various, mostly blank function handlers. After finally getting the thing to link, I ended up with a kernel that was hard-coded for a 5MB 68020-based Macintosh with FPU and a display at 0xF9000000. It had no interrupt controllers, no disk controllers, no keyboard and no mouse. Anything else I could find was also hard coded. However, it linked and that was the important thing. Having done a bit of reading up on the innards of the console drivers (and much interrogation of Jes), I wrote a fairly simplistic back end for the generic console driver on the Macintosh. As it turned out, the very simplistic approach reflected the Macintosh hardware I had, which was a completely unaccelerated bitmapped display supporting 640x480 in 4-bit colour.

Paint It Black

A Linux 68K kernel starts with a partially shared piece of initialisation code written in 680x0 assembler and using almost all the most Gothic and peculiar features of the architecture. This initialisation code also sets up the memory management and caching, and touches everything no one normally knows about. The 68020, 68851, 68881 combination of chips used in the Macintosh II is obsolete and Motorola didn't carry documentation on this device. I did know two things which, in theory, were enough to debug and figure out what was going on. First, I knew the base address of the screen memory; second, I knew the address that the code would begin executing. The very first routine I put in the startup code painted the screen a revolting blue colour. After about 15 boots and some staring at the source code, I had a Macintosh that booted to a blue screen, waited a short while, then crashed.

In many ways, this was the single hardest item to get going. When dealing with a completely unknown system environment with no idea what is around the code, debugging is extremely tricky. Real commercial hardware people use logic analysers—I didn't have that option. I learned several things in the process; notably, that Macintosh screen memory is not located where the hardware claims it is until the MMU is set up. I also made the amazing discovery that the rounded corners on the Macintosh display are drawn in software.

Over the next few weeks, the Macintosh went through an assortment of debugging stripes and coloured patterns as I inched a few lines at a time through the initialisation assembler code, fixing it bit by bit and gradually mapping in the needed hardware. Eventually, the kernel hit the magic **start_kernel** function in the C code without crashing on the way.

Consoling Yourself

Hitting start_kernel is the beginning of the easy road; at least on a PC, text-mode consoles are now present instead of stripes. So theoretically, hitting start_kernel on a Macintosh should have meant that getting the kernel to initialise a text console and begin showing useful debugging information was close. Nothing could have been further from the truth.

After several attempts to get the console up, I wrote some routines to print penguins and Macintosh logos on the screen (this was easier than text). Each significant point the kernel reached added a penguin to the display, and a failure point before the console came up printed a given number of burning Macintosh logos. While hardly as good as print statements, this was good enough to rapidly locate several bugs in the processing of options passed by the boot loader (little things like apparently having 0KB of memory upset the Linux memory initialisation). The code would get to the beginning of the console setup and die.

To get past this point, I had to fill in support for the 4-bit packed pixel displays that were used by the Apple Macintosh “Toby” display card. The generic bitmapped console drivers for the 680x0 port supported a wide variety of pixel formats and naturally excluded the one I needed.

Had I known at the time, I could have simply switched the machine to Mono in the display preferences, but I didn't know that action physically switched the card into a monochrome mode. Adding 4-bit packed pixel wasn't too difficult. I left the somewhat scarier 2-bit packed pixel support for later, hoping someone else would write it. The console code is also very modular on the 680x0, and these console layers (abscon, fbcon) are now used by most non-Intel ports. It is reasonable to assume that it will be driving all the ports by the 2.3 kernel series.

The machine still crashed mysteriously and all evidence pointed to a structure getting stamped on. I put guard values on either side of it and checked that they were not overwritten; I moved the structure in memory; I tried everything I could think of in order to stop it from being apparently corrupted. (No joy, no change.) After a bit of head scratching, I added code to check that the values were okay at boot and at initialisation of each subsystem. The value was wrong at the start of the C code; it was also wrong at the start of the assembler.

This looked as if the boot loader was corrupting data, yet this made no sense, since the loader would corrupt the same location, not pick on a specific variable wherever it might be located. Eventually, I used the GNU **objdump** tools to look at the binary I was loading. It turned out the GNU linker was at fault and in some places was loading a completely bogus address for relocation.

A new linker and the magic words “Calibrating Bogomips” appeared on the screen, followed by a hang, then much rejoicing. In many ways, the time lost to the linker bug was not that bad. Eyeballing the code in search of the mystery bug, I had fixed some twenty or thirty other serious bugs in a vain attempt to find the illusionary real bug.

I wasn't too worried about the Bogomip calibration hanging. It is hard to calibrate time before the interrupt routines and, in particular, the timer interrupt routines have been written. I commented it out and after a short while the rest of the code booted to the point of saying “Panic:unable to mount root file system”. A reasonable situation, as it had exactly no device support except the screen.

Filling In the Blanks

Getting the machine to the point where everything appears to boot is by no means a completion of the first steps of a porting project. This stage is when you finally appreciate the real problems and the scale of work remaining to be done.

The most important items to fill in were those that dealt with the most basic system resources: interrupts, memory and the I/O buses. The interrupts and several I/O subsystems are handled by a pair of 6522 VIA chips, 8-bit controllers from the Stone Age. These chips themselves are documented and their locations are known, even if some of the connections to their I/O pins are a mystery. A certain amount of mapping work and other detective information showed that the VIA chips provided the all-important system timer ticks, handled the keyboard at an extremely low (and at the time undeciphered) level and provided interfaces for external interrupts from the bus controllers.

Several other pins appear to do things such as turn the Macintosh off. Even now, we don't know what everything on the VIA chips does or if all the pins have a real use. It also turned out I got the easy end. The later Macintosh machines replace the second VIA with a device known as RBV (RAM-based video), which contains a bad emulation of a VIA chip and various other components in one piece of glue logic.

Basic interrupt handling on a Macintosh is relatively clean. A great deal of attention has been paid to keeping interrupts that need a fast response at a

higher priority than time-consuming processes. That works well under MacOS, but Linux tends to take rather too binary a view of interrupts, especially in the drivers. Certain interrupts are wired in strange ways, presumably to save components; the SCSI interrupt, for example, is wired through a VIA but is effectively upside down compared to the other interrupt sources. Apple saved an inverter by using as an interrupt signal the fact that the VIA can handle either direction of state change.

I ended up with two layers of interrupt handling, which were mostly hard coded. Unlike a PC, the Macintosh interrupts are hard wired. Only the Nubus (plug in) cards change positions, and they all share one interrupt which sets bits in a VIA register to indicate the real interrupt source.

Nubus proved quite entertaining. The documentation is weak and written from the viewpoint of someone building a card for a Macintosh. It took about a week before the boot-up code would scan and report a list of which Nubus slots were occupied and the names of the devices. Once it worked, the Nubus turned out to be an extremely well-designed system with features much like PCI. Each slot is allocated a set of memory resources and can raise an interrupt. A ROM allows the OS to read each device for identification and driver information. The ROM also contains other “useful” data, including icons for the device. At the moment, these are not visible under Linux, but the intention is to support /proc/nubus/[*slot*]/icon.xpm at some time in the future.

Mapping Ethernet Cards

The Daynaport card I had been given was very close to several PC designs. The 8390 Ethernet chip and block of RAM on it made that quite clear. There are, however, 224 possible locations for the chip and memory within each Nubus slot space.

Finding out where the device was hidden required building a collection of kernels which searched the 24 bits of address space looking for two things. First, it looked for areas of memory which could be read and written; second, it looked for areas like those which had the additional property of giving different results when read back. The 8390 chip has several control registers; by playing with these, it is possible to reliably identify the chip. (This same code is used to probe for NE2000 and WD80x3 cards in Linux for PC.) On the Macintosh, the RAM was easy to find but the 8390 did not show up.

Having played with the RAM behaviour a bit, I discovered that the memory was mapped to alternate 16 bits in its address space. That is, if you wanted to read it, you had to read two bytes, skip two bytes, read two bytes, etc. A bit of further experimentation revealed that the Ethernet controller registers occurred every

fourth byte, that the RAM occurred every other pair of bytes and was 16-bits wide and that the Ethernet controller saw the 16-bit wide memory as 8-bit wide.

This sort of technique worked for mapping a large number of devices and address spaces and helped to discover the location of additional devices in the Apple I/O spaces. We still don't know enough to drive the Apple sound chip and the “Integrated Woz Machine” (floppy disk controller), but we do know where they are located.

Rooting for NFS

When you need to start testing a system by booting into user space, you need a file system. The NFS root file system is extremely attractive for this and has been used for most ports. The NFS (Network File System) makes transaction requests at the level of files rather than disk blocks. This has the saving grace that errors in the new port cause transactions to get rejected. If you are trying to debug a new port and a SCSI controller driver at the same time, you will instead spend much of your time reformatting and reinstalling the disk from which you are attempting to boot. Using NFS limits the possibilities for errors and makes it easier to add and edit files as you attempt to make the machine work.

The initial installs were done with a set of **tar** files, known as “watchtower”, for the M68K. Watchtower is extremely outdated but is small and easy to unpack. Since the goal was to get a shell prompt, the age of the binaries was not a serious worry. Watchtower also demonstrates another strength of Linux/M68K —all the ports run the same binaries. Instead of having to cross compile and debug all the binaries for the Macintosh, I was unpacking and booting a file system set up for installation on a Commodore Amiga.

With a few modifications to the drivers and several small bug fixes to the kernel code, the applications began to run. As most of the code we needed to add for a new M68K platform was drivers and setup code, once things began to work, most applications sprang to life. It took a couple of tweaks to get floating point to always behave, but once done, I was able to boot the machine fully multi-user but without keyboard, mouse or hard disk support.

It took almost a month before anyone else got the kernel to boot on their own machine. A lot of debugging removed some rather bad assumptions that had “escaped” the code cleanup. Gradually, other MacLinux 68K machines began to pop into being. This is an extremely important step for any project, as it allows others to contribute effectively. Michael Schmitz wrote the SCSI drivers and much of the keyboard and mouse support. He is now adding IDE. Numerous others have tested and debugged code on many varieties of Macintosh and even made it work on some.

Conclusions

While any new port is difficult, the structure of the Linux M68K kernel tree is very well-designed and delivers on its intention to allow easy portability between M68K targets. Several sections of this code are rightfully being used cross-architecture as well as cross-platform.

Making a free software port work seems to be about having a small number of people willing to take the project the first half of the way. Once this point is reached, the project gathers momentum of its own accord, even when it is something as pointless as Linux on a Macintosh II.

Lack of documentation is only a hindrance. It will not prevent determined people from exercising their basic rights to use and operate property they bought and now own. Instead, it reflects badly on the vendor who is trying to be a nuisance. If the only documentation on the keyboard interface is entitled “Space aliens ate my mouse”, someone will still find it.

Always be the second operating system ported to an undocumented platform. The sterling work done by the OpenBSD/Macintosh team was a huge help to the Linux project. I'm also happy to say that even though half of the world may spend their time arguing on Usenet advocacy groups, the relationship between the Linux and BSD Macintosh teams has always been one of mutual co-operation. Together, we advance our detective work and knowledge of the Macintosh platforms to the good of all Macintosh users dumped and orphaned by Apple.

Thanks

Alan Cox started hacking on Linux 0.95. He's since discovered he doesn't like working for small non-Linux companies and especially not for big ones, so he now runs Building #3, a Linux contracting company primarily working for Red Hat software. He can be reached at alan@snowcrash.cymru.net.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Creating a Web-based BBS, Part 1

Reuven M. Lerner

Issue #57, January 1999

Ready to create your own virtual community? Here's how to begin.

For a period of time last year, the buzzword on the Web was “community”. Everyone wanted to build a virtual community, allowing people to interact on-line in much the same way as they interact in real life.

While virtual communities were (and are) overhyped, it is true that the Internet has produced a number of such on-line groups, many of whose members have never met in person. If you're reading this magazine, you probably participate in at least one e-mail list, chat system or Usenet newsgroup. Indeed, Linux would probably not be the success it is today were it not for communities of developers and users sharing information with each other via the Internet.

Several methods are available for creating an on-line community, beginning with the oldest and best-known, an e-mail list. Setting up a mailing list is relatively easy, and only minimal resources are necessary to keep a list running. Another popular option is a Usenet newsgroup, which uses a similar format but a different distribution mechanism than e-mail.

Still another option is a web-based bulletin board system. While such systems are neither as flexible nor as powerful as Usenet or e-mail lists, they do offer a number of advantages. They are resistant to spam, can be easily integrated into other aspects of a web site and give visitors to the web site a chance to participate in discussion without having to register. Many commercial web sites now offer bulletin boards for their users, in the hopes of turning their site into a truly interactive and two-way experience, rather than another distribution medium for their content.

Starting in this issue, we will take a three-part look at how to create a simple bulletin board system of our own. This project was suggested by reader Dwight Johnson and also influenced by my creation of an “At the Forge” home page

that will include examples of the programs presented in these columns, as well as a central place for readers to discuss the programs.

This month, we will look at the basic guts of the bulletin board system to be used on the ATF site. As you will see, I have decided to keep the software and the BBS very simple, without certain advanced features such as hierarchies and threading. However, it should not be difficult to add these features to the software, or to use this as a base for a more advanced system. Next month, we will add enough features to make this a serviceable BBS. Finally, in the third part of this series, we will look at ways in which we can add a number of useful features to the system.

Designing the BBS

The first consideration is the look and feel of the BBS, since that will force our hand on a number of other issues. As I indicated above, it is my goal to keep this software as simple as possible. I decided to keep discussions in a non-hierarchical manner. Each message belongs to a single thread within the BBS. We will not keep track of replies or allow sub-threads. Messages within a thread will be presented in chronological order, from the newest message to the oldest.

The user will thus have several possible options at any given point: starting a new thread, posting a new message to an existing thread, listing the current threads, or looking through the messages in one thread.

While I briefly considered storing messages in ASCII text files, I quickly decided to use a relational database. A database makes it easier to handle future expansion, since more features can be provided by adding one or more columns to a table. Databases also free us from having to worry about file formats, locking and other problems which inevitably occur when we use ASCII text files.

My database of choice is the “mostly free” MySQL. The programs will be written in Perl and will use Perl's database interface, known as DBI. See the “Resources” sidebar for pointers to information about any or all of these.

If you have been following this column over the last few months, you may be surprised to see that I have implemented it using simple CGI programs. I could have used mod_perl, a module that embeds a Perl binary inside of the Apache HTTP server. I could also have used HTML::Embperl, the templating language we explored in this column's previous two installments.

However, reality is often the compelling factor and the web space provider I use has not yet installed mod_perl. These programs should run just fine under

Apache::Registry, the modules for mod_perl that provide emulation of the CGI standard.

Creating the Tables

If we are going to store information in a relational database, the first technical decision involves the database itself. What information do we want to store, and how do we want to store it?

Because we are storing messages and threads, I designed the system with two tables, ATFThreads and ATFMessages. Each message, including information about the author and the posting date, is stored in ATFMessages. Each message in the table points to a single thread in ATFThreads, allowing us to sort messages by thread.

Here, for instance, is the definition of ATFThreads:

```
CREATE TABLE ATFThreads (
    id SMALLINT UNSIGNED AUTO_INCREMENT
        PRIMARY KEY,
    subject VARCHAR(255) NOT NULL,
    author VARCHAR(60) NOT NULL,
    email VARCHAR(60) NOT NULL,
    text TEXT NOT NULL,
    date DATETIME NOT NULL,
    UNIQUE(subject)
 );
```

Each thread is stored in a single row of the database, uniquely identified by its **id** column, which we define to be a **SMALLINT UNSIGNED**. (We are thus allowed 65,535 different topics, which should suffice for now.) By declaring the column to be AUTO_INCREMENT, we are asking MySQL to give the **id** column a new value each time we insert a new row. By declaring it to be the PRIMARY KEY, we indicate that the **id** column will uniquely identify a row.

The other columns are fairly self-explanatory: **subject** contains the subject of the thread, while **author** and **email** contain the thread creator's name and e-mail address, respectively.

Each thread has an opening message that starts the discussion; it is stored in the **text** column in a column of type **TEXT**. **TEXT** fields can contain amounts of text larger than the 255-character maximum given to us by **VARCHAR** columns. **VARCHAR** columns are stripped of trailing whitespace, sparing us from at least one housekeeping chore when working with the database.

Finally, we give each thread a **date** column in which we record the creation date and time with a **DATETIME** element. We also ensure that the human-readable subject line for the thread is unique with the **UNIQUE** keyword at the end of the

table definition. This prevents us from having two threads named “Problems with MySQL”, for example.

Now that we have seen how to create ATFThreads, we can define ATFMessages. The two are quite similar, the main difference being a reference to a thread ID:

```
CREATE TABLE ATFMessages (
    id MEDIUMINT UNSIGNED AUTO_INCREMENT
        PRIMARY KEY,
    thread SMALLINT UNSIGNED NOT NULL,
    subject VARCHAR(60) NOT NULL DEFAULT
        "No subject",
    date DATETIME NOT NULL,
    author VARCHAR(60) NOT NULL DEFAULT
        "Mr. Nobody",
    email VARCHAR(60) NOT NULL DEFAULT
        "atf@lerner.co.il",
    text TEXT NOT NULL
 );
```

Once again, we create a column with an auto-incrementing primary key named **id**. Different tables can have identically named keys just as different hashes can. If we are referring to both tables in a single query, we can distinguish between the two by using the **table.column** syntax, as in **ATFMessages.id** and **ATFThreads.id**.

Notice how we have used the **DEFAULT** keyword to assign default values to each of the elements. Truth be told, the way the database-handling programs are written makes it unlikely we will ever see these defaults. (Empty strings are passed to the database as empty strings, rather than as NULL values. To get a true NULL, we must pass an undefined scalar.) However, it is always a good idea to build multiple checks into your programs just in case one of the other levels does not work in the way you expected. This can also help us track down problems; if we notice that many users are identified as “Mr. Nobody”, we can assume something has gone wrong with our posting software.

We can create the tables by entering the above SQL commands at the interactive **mysql** prompt. Once they have been created, we are ready to start working on the programs.

Common Program Elements

The programs in this project share a number of elements. Each starts with a series of **use** statements:

```
use strict;
use diagnostics;
use CGI;
use CGI::Carp qw(fatalsToBrowser);
use DBI;
```

The first, **use strict**, prods us into making our variable references explicit, either by creating them as lexicals (with the **my** statement) or with the **use vars** statement. I have chosen to create all variables as lexicals, but if you were interested in putting common variable definitions into an external file, you might want to consider making them globals.

Next, we invoke **use diagnostics**, which tells Perl to give us information from the **perldiag** manual page if and when there are problems with our program. I find **use diagnostics** to be an invaluable debugging tool when working with web applications, since it often points to a foolish mistake I have made. This, along with **use strict** and the **-w** flag, makes programming in Perl much less error-prone.

We then load the **CGI::Carp** module, which overrides the built-in **Carp** module with routines of its own that make for more accurate messages in our HTTP server's error log. We also import **CGI::Carp::fatalsToBrowser**, which sends an error message to the user's browser if and when an error occurs. This allows us to use the standard **die** statement without having to worry about whether we have already sent the HTTP “Content-type” header. Sending a message to the user's browser without such a header almost always causes an error message to be displayed.

Each program in the BBS also defines a number of variables: **$database**, **$server**, **$port**, **$username** and **$password**. These variables are used to log into the database with DBI; by setting them at the top of the program, you can modify them as necessary without having to change hard-coded strings.

Each program also turns off buffering, so that information is sent to the user's browser as soon as the program sends it to the appropriate file handle. Normally, saying

```
print "<P>Hello</P>";
```

does not send **<P>Hello</P>** to the user's browser. Rather, it places the string in a buffer. When the buffer is filled, its contents are shipped off to the user's browser. This is more efficient, since the computer can copy a lot of data at once, rather than spending its time entering and exiting from the routines that handle file operations. However, it also means the user must wait to see results. We can turn off buffering by setting the built-in Perl variable **$|**:

```
$| = 1;
```

Finally, each program connects to the database with the standard DBI routine:

```
my $dbh =
 DBI->connect("DBI:mysql:$database:$server:$port",
 $username, $password);
```

If the connection succeeds, we receive a database handle (dbh) in return and store it in **$dbh**. If **$dbh** is false, however, we should report an error, since it means the connection did not work:

```
die "DBI error from connect:", $DBI::errstr
    unless $dbh;
```

We can do the same thing when preparing a query. The result from **$dbh->prepare** is a statement handle (sth). When defined, **$sth** is an object that itself accepts methods. When **$sth** is undefined, the statement preparation failed:

```
my $sth = $dbh->prepare($sql);
die "DBI error with prepare: ", $sth->errstr
    unless $sth;
```

We can execute our statement with **$sth->execute**, which works in much the same way as **$dbh->prepare**. The difference is that the result code is a simple value, rather than an object:

```
my $result = $sth->execute;
```

In some programs, we test the value of **$result** and use **die** to report an error:

```
die "DBI error with execute: ", $sth->errstr
    unless $result;
```

In others, we use **$result** to decide whether to continue with the program or to print a more user-friendly error message:

```
if ($result)
{   # do something
}
else
{   # indicate an error
}
```

Finally, we always disconnect from the database at the end of our programs:

```
$dbh->disconnect;
```

This is not truly necessary, since DBI and Perl close all such connections when the program exits. However, if you are running with **-w**, a message will be inserted into your error log each time a program exits without disconnecting from the database nicely. We do this in order to keep our error log free of spurious details.

Creating and Viewing Threads

Since each message must belong to a thread, we will first look at how a thread is created. A thread is no more than a single row in the ATFThreads table, so our thread-creation program will be fairly simple.

The three listings referred to in this article are available for anonymous download at ftp.linuxjournal.com/pub/lj/listings/issue57/3193.tgz. They are not printed here due to space considerations.

Add-thread.pl (Listing 1 in the archive file) uses the contents of an HTML form to insert a new row into ATFThreads. However, it performs some additional manipulation as well, to ensure that the data will be retrievable in a useful way.

We can use either single or double quotes around text strings in SQL queries. Double quotes are used by DBI for parameters and thus exclude the possibility of using quotation marks. We therefore use single quotes around our text strings. However, this raises the issue of how to pass single quotes to the program. A simple solution is to perform a substitution on each of the text strings generated by the user. For example:

```
$value{"subject"} = $query->param("subject");
$value{"subject"} =~ s/\'/\'\'/g;
```

We can do even better by using the built-in **$dbh->quote** method, which quotes a text string for us. **$dbh->quote** decides whether to use single or double quotes and also handles special characters, such as quotation marks and question marks, with ease. We use a **foreach** loop to quote each of the elements:

```
# Get the form parameters
foreach my $element (qw(subject text author
    email))
{
$value{$element} =
    $dbh->quote($query->param($element));
}
```

Once we have done this, we can be sure that **$value{$element}** is suitable for insertion into the database.

We also perform several substitutions on the “text” HTML element, which contains the text that starts the thread. To begin, we remove all HTML tags, so as to prevent people from linking to all sorts of crazy sites. While it might be desirable to allow people to include HTML in their postings, it could also lead to chaos if formatting commands were inserted. I decided to be slightly draconian and disallowed all HTML. We do that by removing everything between **<** and **>**:

```
$text =~ s/<.*?>//sg;
```

Notice how we use Perl's non-greedy operator ***?** instead of ***** to remove the HTML tag. If we were to use ***** and the line had two HTML tags, Perl would remove everything from the first **<** through the final **>**. We use the **/s** modifier to tell Perl that **.** includes all characters, including new lines. Without **/s**, **\n** would not be included in **.**, which means a two-line tag such as

```
<a
 href="http://www.cnn.com/">
```

would be ignored.

We then make sure new lines are treated correctly, first removing multiple new lines and then replacing them with HTML paragraph markers:

```
$text =~ s/\r\n/\n/g;
$text =~ s/\r/\n/g;
$text =~ s|\n\n|</P>\n<\P>|gi;
```

Once it has performed all of these tasks, add-thread.pl creates the SQL query that will insert the new thread into ATFThreads:

```
my $sql = "INSERT INTO ATFThreads ";
$sql .= " (subject, text, author, email, date) ";
$sql .= "VALUES ($values, NOW())";
```

We insert the date of the thread for future use, but also so that we can sort the threads in the order of their creation.

The program which lists threads, appropriately named **list-threads.pl** (Listing 2 in the archive file), uses a SELECT query to retrieve all of the rows from ATFThreads:

```
my $sql =
"SELECT id,subject FROM ATFThreads ORDER BY subject";
```

After performing an **$sth->execute**, it checks to see how many rows were returned. If none were returned, we indicate that no threads have yet been created. If threads exist, we iterate through the results with **$sth->fetchrow**, which places the query result into **@row**. We can pull out the elements of **@row** and print a list:

```
if ($sth->rows)
{
print "<ul>\n";
while (my @row = $sth->fetchrow)
{
 print "<li> ";
 print "<a href=\"/cgi-bin/view-thread.pl?";
 print "$row[0]\">$row[1]</a>\n";
}
print "</ul>\n";
$sth->finish;
}
```

Users are presented with an alphabetical list of thread titles, each of which is a hyperlink to **view-thread.pl** (Listing 3 in the archive file), described below. As you can see, the argument to view-thread.pl is the **id** value of the thread, the defined primary key.

Conclusion

Next month, we will complete the design and implementation of our basic BBS, adding the ability to create messages and search through the system. Until then, consider dropping by a working implementation of this software at http://www.lerner.co.il/atf/, where you can trade ideas with other readers of this column.

Resources

Reuven M. Lerner is an Internet and Web consultant living in Haifa, Israel, who has been using the Web since early 1993. His book Core Perl will be published by Prentice-Hall in the spring. He can be reached at reuven@lerner.co.il. The ATF home page, including archives and discussion forums, is at http://www.lerner.co.il/atf/.

Archive Index Issue Table of Contents

Advanced search

Letters to the Editor

Various

Issue #57, January 1999

Readers sound off.

Thanks for the On-line Reference

Over a year ago, you ran an article by John Little (“Setting up a Sun SPARCstation”, October 1997) on getting a SPARC up and running on Red Hat 4.2. Shortly after reading it, I came across an old SPARC IPX at the local computer graveyard that essentially needed a CMOS battery to function properly. I snapped it up and never got around to installing Linux on it until recently. Since I've moved, I couldn't find the original article and was overjoyed when I found it in its entirety on your web site. That information along with the Red Hat Powertools 5.1 disc allowed me to get the IPX up and running in under an hour.

Thanks again for all the great information with no fluff! —Leon Hauck
leon@progcpu.com

Future of Linux Followup

I just received the October issue of *Linux Journal* and was alarmed to discover that I had omitted from my “Future of Linux” report the most important “Resource” link of all: that of the on-line version of the article itself, which I continue to update almost weekly:

http://pobox.com/~newt/reports/linux-19980714-top.html

Recent developments include Linux product announcements from Caldera (Netware), Citrix and Sybase; a *San Francisco Chronicle* report that the last of the large databases without a Linux port, IBM's DB2, will be announced for Linux by the end of September; and news about Dell's Linux pre-installations, a number of Open Source announcements and a new Linux quarterly in French: *Linux Magazine France*.

More important, perhaps, is the fact that the dozens of links embedded in the text (e.g., for the Top 500 supercomputer list or the AP1000+ multi-computer) are available. —Greg Roelof snewt@pobox.com

Re: July 1998 Issue

It's ironic that when I was reading Griffin Caprio's letter in the October issue, I had the July issue in my briefcase to take to work in order to photocopy two articles for some coworkers. Please do not let the prospect of big circulation lead you to water down the content of *LJ*. I have seen other computer magazines go that way, and abandoned them. *LJ* is now the only one to which I still subscribe. —Tom Kuiper kuiper@jpl.nasa.gov

Office Suite Review

In the article “Applixware vs. StarOffice” that appeared in the October *LJ*, Mr. Butzen states that his criteria for testing included portability, specifically the “... ability to import and export files to Microsoft Office.” While he touched on this issue briefly with his experience exporting to Word 6.0, he neglected a major issue in this regard.

To my knowledge, neither suite's presentation package will export to Microsoft PowerPoint format. While this may seem like a minor point to some, it is a huge issue for those of us who use presentation software as a routine part of our jobs. Most of us are not fortunate enough to have access to PC/projection systems running Linux and the appropriate presentation software, let alone fortunate enough to find someone willing or able to create 35mm slides from either of these formats. While exporting presentations to Windows Meta Files is supported, this is a poor substitute compared to having the project saved in a format that is easily edited and displayed, particularly when one is far from his friendly Linux box.

I have been a StarOffice 4 user for about six months. While I find it a bit slow, the major factor preventing me from completely abandoning Microsoft Windows95 and Microsoft Office is StarOffice's inability to export presentation files to a more “universal” format. I'm surprised that with Linux's popularity in the scientific community and the common use of presentation software in those endeavors, this subject has not received more attention. —Frank Lynch, MD flynch@statecollege.com

October STP Column

I enjoyed the October “Stop the Presses” column, but felt it was rather dated by the time it hit my mailbox, given the recent actions by Oracle, Sybase and IBM(DB2). Have you considered running the column on the *LJ* web site instead,

so as to keep the column from becoming too out of sync with current Linux events?

I've been working in the database industry for the past 2 1/2 years now, so I naturally feel that the database market will be critical to Linux's future success. As such, I've kept a close eye on the Informix, Oracle, Sybase and IBM announcements of late. Although the Informix-centric information was most excellent, another article covering the recent adoption of the Linux platform by all the major database names would be most welcome. (I'd be interested in finding out whether my own suspicions about the meaning of these trends are shared by others.)

Regardless, I'm planning on photocopying the “Stop the Presses” article and posting it outside my door as the latest salvo in my long-running Linux-PR campaign I've been waging within my company.

Many thanks, and keep up the good work! —Peter Kuklaf ruviad@coil.com

We do sometimes put the “Stop the Presses” column on the web as soon as it is written, usually in our on-line e-zine *Linux Gazette*. However, that did not happen for this particular column —Editor

Redirecting to Console 9

In my previous letter you published in the “Letters to the Editor” column, I referred offhand to “redirecting output to Console 9, as described in a previous issue of *Linux Journal*”. I have received quite a few inquiries as to which issue that was in and how to do it.

The issue was #31, the column was “Novice to Novice: Keyboards, Consoles and VT Cruising” by John M. Fisk and the page was 17, the section heading “Putting that Unused VT to Work”. However, you don't have to look up that issue, as it is quite simple. Just add the following line to /etc/syslog.conf:

```
*.*         /dev/tty9
```

and all your syslog messages are sent to console 9 as well.

Thanks, *Linux Journal*, for publishing that tip back in November 1996. It has made a lot of system administration tasks easier. —Cynthia Higginbotham cyhiggin@pipeline.com

Intel and Red Hat

I have been using Linux since 1992 and have built up a successful business providing offices with an effective cheap alternative to expensive Microsoft servers and products. Up until the beginning of this year, it did not matter which distribution you used; everything worked just fine. Okay, each distribution worked slightly different, but they all used the same libraries and kernel. When I read in the computing press that the big software houses were starting to support Linux, I knew that before long, Linux would rival MS on the same level for both back end and desktop.

However, within a few weeks of Intel buying a stake in Red Hat, I read that some products will work only with the Red Hat version of Linux. We must stop this now, or in two years we will have a situation of Intel-Hat becoming the next Microsoft with an 85% stake in the Linux market. Last week I installed a mail, Samba and web proxy server into an office in London. I was asked, “Is this Red Hat you are installing?” I said “No, it is Linux.”

I believe we must push for a common standard. Any product released for Linux must work on all distributions. Please don't let Intel-Hat muscle their way in or we will be in the same position in two to three years as we are with Microsoft now. —Robert Weeks robert.weeks@csfp.co.uk

Review of Applixware vs. StarOffice

I read *LJ* here in Germany and have often read good reviews in your magazine. Anyway, this special review “Applix vs. StarOffice” by Fred Butzen in the October issue is somewhat incorrect.

StarDivision released StarOffice 4.0 some time ago, and three service packs have already reached the users. Since 4.0, they are no longer using Motif but their own StarView GUI which has a Windows look and feel. Speed, features and reliability have all been improved greatly since the 3.0 version. By now, the Linux StarOffice 4.5 preview release is out and can be tested. This preview got released even before the other 4.5 preview releases had been shipped. You can even buy an official version of StarOffice 4.5 including the handbook and a sheet with notes about the differences between the Windows and Linux versions. They are actually selling the 4.0sp3 version now, with the guarantee of sending you the new 4.5 CD-ROM as soon as it is no longer a preview/beta version. —Holger Lehmann lehman_h@informatik.fh-hamburg.de

Archive Index Issue Table of Contents

Advanced search

1998 Atlanta Linux Showcase

Norman M. Jacobowitz

Issue #57, January 1999

One of the most notable aspects of this year's show was the variety of attendees, over 2000 in all.

The 1998 Atlanta Linux Showcase was held October 23 and 24 in downtown Atlanta, GA. Thanks to Greg Hankins and the rest of the Atlanta Linux Enthusiasts, it turned out to be yet another tremendously successful Linux gathering.

One of the most notable aspects of this year's show was the variety of attendees, over 2000 in all. Visitors spanned the gamut from faithful Linux geeks to the merely curious. Coupled with the wide variety of exhibitors and speakers, these guests made the 1998 ALS one of the most diverse crowds ever for a Linux show.

Among the 60 participating vendors were most of the usual suspects, such as Red Hat Software, Caldera, Debian, S.u.S.E. and SpellCaster. Adding to the excitement of the show was the presence of a few noteworthy newcomers. Oracle gave Linux users a sampler CD-ROM of their database system, and Informix also presented their database applications. Corel showed off both the NetWinder and WordPerfect 8 word processor for Linux. San Mehat, NetWinder developer, even showed off his pet project—a ten-processor Beowulf cluster that he carried around in one hand. Even Schlumberger had a booth where they talked about their smart card embedded with the Linux OS (see “Muscle Flexes Smart Cards into Linux” by David Corcoran, *Linux Journal*, August 1998).

The show was tied together with a one-gigabyte backbone set up by Cabletron. Registration was run on Linux-based PCs. In fact, one Microsoft employee we talked to felt very much in the minority. Hopefully, he ordered a Cobalt Qube or NetWinder to take home.

Other exhibitors included Jeff Bates and Rob Malda of slashdot.org. The GNOME team was out in force, showing us the fruits of their labors. Miguel de Icaza gave a talk on the GNOME project. Several retailers were represented, such as Linux Mall, Linux Central and the Linux General Store.

Many leaders of the GNU/Linux revolution were on hand. Eric S. Raymond was there making his always-brilliant philosophical observations about the power and importance of Open Source software. Also present was Richard M. Stallman: the man who can be said to have started it all.

As expected, a few important announcements were made, in particular by keynote speakers. Allen Miner, Vice President of Strategic Business Development at Oracle Corporation, gave the first keynote speech. Mr. Miner reasserted Oracle's commitment to Linux and announced several key partnerships with other Linux vendors. Look for Linux to figure more prominently in Oracle's future.

Dr. Michael Cowpland of Corel talked about the future of his company's relationship with Linux. He also announced that Word Perfect 8 for Linux would be available as a free download for personal use in the near future. The rest of Corel's productivity applications should be available on Linux by early 1999. Corel also recently announced plans to team with Red Hat to port more software to their NetWinder.

The surprise hit of the show was the caffeinated Penguin Peppermints handed out by *LJ* representatives, including yours truly. Folks lined up three rows deep at the booth to pick up these treats.

To no one's surprise, this year's Atlanta Linux Showcase was yet another testament of Linux's growth towards complete world domination. As the list of independent software vendors, value-added resellers and retailers supporting Linux grows each day, the outlook for the future of Linux is quite optimistic.

Norman M. Jacobowitz is a freelance writer and marketing consultant based in Seattle, Washington. He can be reached at normj@aa.net.

Archive Index Issue Table of Contents

Advanced search

New Products

Amy Kukuk

Issue #57, January 1999

Applixware 4.4.1, FootPrints v 2.2, Linux Office Suite 99 and more.

Applixware 4.4.1

Applix, Inc. has announced the release of Applixware 4.4.1 for the Linux platform with a new filtering framework that has been optimized for MS Office document interchange and Y2K compliance. Applixware includes Applix Words, Spreadsheets, Graphics, Presents and HTML Author. This Linux version also includes Applix Data and Applix Builder as standard modules. Applixware for Linux is available directly from Applix and from its partners, including Red Hat and S.u.S.E. The product is available for $99 US.

Contact: Applix, Inc., 112 Turnpike Road, Westboro, MA 01581, Phone: 508-870-0300, Fax: 508-870-0300, E-mail: applixinfo@applix.com, URL: http://www.applix.com/.

FootPrints v 2.2

UniPress Software, Inc. has announced the release of FootPrints version 2.2, a web-based help desk software package. FootPrints is designed to utilize the Internet to track problems and solutions and make that information available to users over the Internet or Intranet. This version includes e-mail access (not requiring access to the web) and a facility to load existing data into FootPrints projects. Version 2.2 includes a web forms-based Data Import Facility which

allows existing data to be imported into FootPrints projects. The existing data can be in a database, spreadsheet or any other source. The FootPrints Starter Pack (including three licenses) is priced at $1995. Quantity, site license, reseller and educational pricing are also available.

Contact: UniPress Software, Inc., Phone: 732-287-2100, E-mail: info@unipress.com, URL: http://www.unipress.com/.

Linux Office Suite 99

nux Office Suite 99

S.u.S.E., Inc. has announced the release of Linux Office Suite 99 which includes a spreadsheet, word processor, presentation graphics, database, fax program and other business applications. The product includes Applixware 4.4.1 and integrates Applixware with the powerful ADABAS D 10.0 database system, enabling users to import data from the ADABAS D database into Applix Spreadsheets. It also contains the KDE and GNOME graphical desktops, S.u.S.E. fax, the personal edition of the backup utility ARKEIA 4.0, the GIMP graphics program and many other features. Suggested retail price is $79.95 US.

Contact: S.u.S.E., Inc., 458 Santa Clara Avenue, Oakland, CA 94610, Phone: 510-835-7873, Fax: 1-510-835-7875, E-mail: info@suse.com, URL: http://www.suse.com/.

Etherminal J

IGEL LLC has announced the release of Etherminal J, a thin client desktop device. Etherminal J incorporates Netscape Communicator version 4.05 and a complete set of UNIX connectivity tools in its own flash memory. Storing and running these software modules locally keeps network bandwidth requirements at a minimum. IGEL's Flash Linux is a compressed UNIX-compatible, flash memory accessible operating system. It is a POSIX-conformant, multi-threading, multi-user operating system. Based on the Linux kernel, it offers the largest number of available device drivers and applications. It supports Internet and Java. The list price for Etherminal J is $800 US.

Contact: IGEL*USA, 31 Stonecroft Dr., Suite 105, Palmer, PA 18045-2812, Phone: 610-258-4290, Fax: 610-258-1289, URL: http://www.igelusa.com/.

DupliDisk

Arco Computer Products, Inc. has announced the release of the DupliDisk-Direct, a real-time backup device. Due to its small size and unusual design, which allow it to be plugged directly into one of the computer's onboard IDE

controllers, the DupliDisk-Direct requires neither a PCI nor an ISA bus slot. The DupliDisk is a RAID 1 device capable of mirroring up to two pairs of IDE, EIDE or U/DMA drives of any capacity. If a hardware failure disables one of the mirrored drives, the DupliDisk automatically shifts operations to the functioning drive. Users have the choice of either a bay-mounted or rear-mounted external panel for the visual display. The product is priced at $250 US.

Contact: Arco Computer Products, Inc., 2750 North 29th Ave., Hollywood, FL 33020, Phone: 954-925-2688, Fax: 954-925-2889, E-mail: arco@arcoide.com, URL: http://www.arcoide.com/.

CommuniGate Pro Server

Stalker Software, Inc. has announced its commercial release of CommuniGate Pro, a unified messaging server which supports multiple platforms. CommuniGate Pro communication and access modules provide all types of messaging services required for today's applications. The SMTP module sends and receives messages via the Internet using ESMTP with DSN and multichannel queuing. The POP and IMAP modules implement the standard POP3 and IMAP4rev1 protocols as well as most known extensions to these protocols. The HTTP module supplements these modules with web access to user e-mail, eliminating a need for mailer applications on client computers. The web interface is customizable to include anything from logos or banner advertisements to company announcements.

Contact: Stalker Software, Inc., 655 Redwood Highway, Suite 275, Mill Valley, CA 94941, Phone: 415-383-7164, Fax: 415-383-7461, URL: http://www.stalker.com/.

WebSENSE

Apexx Technology, Inc. has announced the release of WebSENSE web filtering software, simple file sharing and spam filtering software for the TEAM Internet 100 series family. These features provide customers with the complete tool set to control and manage Internet usage. The integration of WebSENSE with TEAM Internet enables small businesses to control Internet browsing by blocking, monitoring and filtering individual web browsing to non-business-related web sites. Simple File Sharing allows TEAM Internet to act as a simple networking server on the LAN, giving employees the ability to retrieve, store and collaborate on files. Spam filtering software automatically filters unwanted e-mail and helps small businesses eliminate expensive network bandwidth while reducing employee time spent reviewing and managing unproductive material. The TEAM Internet solution is based on the Linux kernel. Pricing starts at $1695 US.

Contact: Apexx Technology, Inc., 506 South 11th Street, Boise, ID 83702, Phone: 800-767-4858, E-mail: sales@apexxtech.com, URL: http://www.apexxtech.com/.

Parallel Computing Toolkit

Wolfram Research has introduced parallel computing support for Mathematica. The upcoming Parallel Computing Toolkit will add parallel programming over a network of heterogeneous machines to the long list of programming paradigms supported in Mathematica. It implements many parallel programming primitives and includes high-level commands for parallel execution of operations such as animation, plotting and matrix manipulation. The Parallel Computing Toolkit builds on Mathematica's symbolic programming language. It is written entirely in the Mathematica language and uses Mathematica's standard MathLink protocol to communicate between any number of Mathematica kernels.

Contact: Wolfram Research, Inc., 100 Trade Center Drive, Champaign, IL 61820-7237, Phone: 217-398-0700, E-mail: info@wolfram.com, URL: http://www.wolfram.com/.

Correction

In the November issue's “New Products” column, it was incorrectly stated that Interbase 5 for Linux was free. (The beta version was a free download.) The correct price of the product is available by contacting sales@interbase.com.

Archive Index Issue Table of Contents

Advanced search

Best of Technical Support

Various

Issue #57, January 1999

Our experts answer your technical questions.

umask Trouble

I'm trying to use **umask** to set permissions in a directory, but it doesn't allow me to set execute. I use

```
umask a=rwx
```

and when I create a new file the permissions are **-rw-rw-rw-**.

Can you give me a quick explanation of this command? Thanks. —Ernesto Jardim, ernesto@ipimar.pt

umask doesn't set permissions; it uses a mask to clear existing file permissions. The umask is also used by the shell to set initial file permissions on a newly created file. Specifically, permissions in the umask are turned off from 0666. The default umask is commonly 022 (in octal notation). In binary it is 000 010 010 which is equivalent to **----w--w-**. When a file is created, the default permissions are **rw-rw-rw-** (666) and after the **umask** is applied, they will be **rw-r--r--** (644). To set permissions, use the **chmod** command.

IDE/ATAPI Support?

I have two technical questions that I can't seem to solve by reading HOWTOs.

1) Has anything been done for the IDE/ATAPI version of the Iomega Zip drive? Every HOWTO I have read seems to cover only the SCSI and the parallel port versions.

2) I have an HP ScanJet 5P scanner, with complementary Symbios one device SCSI controller. When I boot Linux, it says it doesn't detect any SCSI hosts. Is this normal and what is the reason behind it? —Henk Verleye, henk@sophis.be

1) Newer kernels (like 2.0.35) support IDE/ATAPI removables. Just include IDE/ATAPI FLOPPY support and recompile the kernel.

2) Frankly, I don't know if this type of SCSI controller is supported, but if it is, make sure the ncr53c8xx SCSI driver is compiled into the kernel.

Switching Hard Drives

I have one hard drive for Linux Red Hat 5.0 and one for Windows and want to switch them. Linux is on hda1 and Windows is on hdb1. hdb1 is the faster of the two, and I want to move Linux to it and put Windows on hda1. I know how to do the Windows part, but how do I duplicate everything on hda1 to hdb1? hdb1 is a bigger hard drive and has more than twice the speed of hda1. —Jon, LordShroom@hempseed.com

First boot Linux, then mount hdb1 under /mnt with **mount /dev/hdb1 /mnt**; then, if one partition is all you need to copy, type the following:

```
cp -a --one-file-system / /mnt
```

Wait for the copy to finish, then type **umount /mnt**. If you have more than one file system you want to copy, you have to repeat this for each partition. Now you need to change /etc/lilo.conf so that LILO boots from hdb1 instead.

Mounting a Zip Disk

I am using Red Hat 5.1 and am having some difficulty mounting a Zip disk formatted in Windows 98. The file system is not FAT32; it is FAT16. I can easily mount a Linux EXT2 Zip, but not the Windows 98 one. I'm not sure if I have the relevant information in my FSTAB—maybe someone can tell me what I need. I've used commands like:

```
mount -t msdos
```

I've tried many variations of this with no success. Is there something I'm missing? The **man mount** help seems informative, but yields no solutions — Edward Heshka, heshka@idirect.com

The default partition used on a Zip disk under DOS/Windows is the fourth partition. Don't ask me why! Add entries similar to these to your /etc/fstab:

```
/dev/sdc1 /zip ext2 noauto,rw,user,nosuid,sync
/dev/sdc4 /zipdos msdos noauto,rw,user,nosuid,sync,mode=0777
```

Make sure the mount points exist and you use the correct SCSI device. Check the messages during bootup if you're not sure. Now you can mount a DOS Zip disk with **mount /zipdos** and an EXT2 Zip disk with **mount /zip**.

Sharing Directories

I'm fairly new to Linux. I have succesfully installed Red Hat Linux 5.1 on my laptop and have configured X appropriately. I have made appropriate network settings and I want to use network shares (i.e., directories) that exist in my company's Windows NT domain. Any suggestions would be greatly appreciated. Also, we use MS Exchange for our e-mail and I have had limited success in configuring a POP3 client to hit the server. Thanks in advance. —William B. Winslow, bill.winslow@atkearney.com

One word: SAMBA. You can find information on SAMBA at http://www.samba.bst.tj/samba/samba.html. Also, read the review in *Linux Journal* of John Blair's book *SAMBA: Integrating UNIX and Windows* to see if it is a resource you are interested in using.

Shutting Down

I am using Red Hat 4.2. I would like to give a user who is not root the permission to shut down the system. The man page says, “write the name of the user in the file /etc/shutdown.allow”. Unfortunately, this has no effect, i.e., the user gets the message “must be root” after typing **shutdown**. —Thomas Okon, okon@math.tu-dresden.de

The only way I know of for any user to correctly shut down a Linux system is to be physically present at the keyboard and press **ctrl-alt-del**. This key sequence has the effect of running shutdown from init(8). This is the default behavior and all /etc/shutdown.allow does is to restrict **ctrl-alt-del** even more to specifically named users.

Updating Web Site

I am using Red Hat 5.0. How can I write a script that compares two directories recursively, one on the localhost, the other on an FTP site, then upload only the newer files to the FTP site? I wish to easily update my web site which is getting quite large and difficult to update manually. —Grim_Sweeper@softhome.net

The good news is the solution is already available. The bad news is that you will have to configure it to fit your needs. I'm talking about the mirror package

available at ftp://src.doc.ic.ac.uk/packages/mirror/mirror.tar.gz. This is an excerpt from the man page:

Mirror was written for use by archive maintainers but can be used by anyone wanting to transfer a lot of files via ftp. Regardless of how it is called, mirror always performs the same basic steps. It connects to the remote site, internally builds a directory listing of the local target directory, builds one for the remote directory, compares them, creates any subdirectories required, transfers the appropriate files (setting their time-stamps to match those on the remote site), creates any symbolic links, removes any unnecessary objects and finally drops the connection.

Archive Index Issue Table of Contents

Advanced search

WWWsmith: Installation and Configuration of FreeBSD

Sean Eric Fagan

Issue #57, January 1999

Here's how to set up a web server using another freely available operating system, FreeBSD, a high performance, mature, UNIX-like system.

FreeBSD is a popular (and free) Unix-like operating system, available from the Internet and on CD-ROM (chiefly from Walnut Creek CD-ROM). In this respect, it shares much with Linux, which is admittedly more popular and better documented.

Unlike Linux, FreeBSD is (as the name implies) derived from the popular BSD variant of Unix; many features considered standard with Unix these days originated at the University of California, Berkeley. These features include (among others) networking and long file names; the networking code, in particular, is mature and high performance. (One of the busiest sites on the Internet is http://wcarchive.cdrom.com/, aka ftp://ftp.cdrom.com/ aka ftp://ftp.freebsd.org/ aka http://www.freebsd.org/; it runs FreeBSD and pumps out data at an average of more than 2.5MB per second, every second of every day.)

In this article, I will describe the process of installing FreeBSD on a LAN, and configuring it to work as a web server, all using free software. Although Unix is not traditionally a user-friendly operating system, FreeBSD does have a usable installation process (provided you read the documentation and have a rough idea what you are doing) and requires very little maintenance.

Before installing FreeBSD, you need to be prepared. First, you need to know how you will choose to install the system: via CD-ROM, NFS or FTP. CD-ROM installs are the easiest and fastest; FTP is the most commonly-used. This requires that the computer you are installing on have access to the Internet (either via LAN or PPP/SLIP).

Hardware Requirements

FreeBSD needs at least an 80386-level processor, with at least 4MB of RAM and about 150MB of disk space. Note that FreeBSD currently requires at least 5MB of RAM for installation—but can get by with 4MB post-install. Most popular disk controllers are supported, including (E)IDE and several SCSI controllers. The machine I am using is a 33MHz 80486, with 16MB of RAM and a 202MB IDE drive. It also has an UltraStor 34f VLB SCSI controller with a CD-ROM drive attached, and a Novell NE2000+ Ethernet card, configured at IRQ 5 and **0x280**.

For PCI systems, almost any DEC 21x40 and 21x41-based Ethernet card will suffice, and both the Adaptec and NCR SCSI controllers are well-supported. The NCR is considerably cheaper and is well-suited to low-load systems. There is some debate as to whether the new versions of Adaptec's SCSI cards are worth the money for high-end systems, due to recent changes Adaptec has made. There is also new support for DPT SCSI cards, including their RAID controller, which may be desirable in some circumstances.

Some IDE CD-ROM drives, and proprietary CD-ROM interfaces, are also supported. The support for those is not as good as for the SCSI. This is true because while a SCSI driver may be quite complex, the command set is very standard, which is not yet the case for IDE CD-ROM drives.

You will also need a boot floppy. The boot image is available at ftp://ftp.freebsd.org/pub/FreeBSD/<release>/floppies/boot.flp and on the CD-ROM as /floppies/boot.flp. The release I used was 3.0-970522-SNAP. If you are creating the boot floppy under a Unix-like system, you would use **dd** to create the image. For example, under FreeBSD, type:

```
dd if=boot.flp of=/dev/rfd0a bs=18k
```

A similar command is used on other Unix systems. If you are creating the floppy under MS-DOS, you will need the rawrite.exe file, which is located in .../tools/rawrite.exe on both the FTP site and CD-ROM. Create the floppy by typing:

```
..     ools\rawrite boot.flp a:
```

I installed the 3.0-SNAP release, which is available on CD-ROM; it is essentially a development snapshot, and hence isn't as stable or mature as the other releases.

Preparing to Install

Before beginning the installation, at least read the release notes. The recommended files to read are INSTALL.TXT, README.TXT and RELNOTES.TXT, all in the release's root directory.

Write down hardware information, such as the disk geometry (heads, sectors, and cylinders)—although this is not truly necessary, it can be useful. Also the configuration of any ISA cards, such as SCSI and Ethernet. Last, since the machine is going to be on a LAN, you should write down the host name, domain name, IP address, default router IP address and name server (DNS) IP address; this will save you a lot of frantic searching later. Note that if you are doing an FTP or NFS install, you need the same information.

Beginning Installation

Installation is begun in the same way as any other equivalent system: put the boot floppy or CD-ROM in the drive. Press **enter** at the **Boot:** prompt; if you don't type anything, it will time out and boot automatically.

First a scrollable menu is presented to let you decide whether or not to configure the kernel. You can choose to skip the configuration step, or you can enter either a visual or line-oriented configuration program. (I recommend the visual mode, of course.)

The kernel configuration process allows you to disable or reconfigure most device drivers; this is invaluable if you have a device card that is configured slightly different from what FreeBSD has been told to expect. Some devices require destructive probes (meaning that probing for one may confuse or disable another device); if you know which devices are not in your system and disable all of those, probes will be less of a concern. Please note that PCI devices are not, currently, configurable—since they are configured on the fly, there is no conflict, and do not need to be re-configured or deleted.

In my case, I disabled all of the mass-storage devices that I did not have, including the Adaptec 154x driver and the second Western Digital controller. The Western Digital driver, **wdc**, controls (E)IDE, ESDI, MFM and RLL hard disk drives. The probe sequence for one of these controllers takes a considerable amount of time, so disabling the second one, _wdc1_, speeds up the boot process measurably.

The visual configuration process is fairly self-explanatory and takes only a few seconds to go through. However, it is not, in most cases, truly necessary. An example of when it would be necessary: if my Ethernet card had not been configured at IRQ 5, I/O port **0x280**, memory address **0x0d8000**, I would need to either reconfigure the card or change what the FreeBSD kernel expected. If you accidently delete a driver, you can reconfigure it by switching to the “Inactive Drivers” section by pressing **tab** and pressing **enter** to re-enable it.

After you've finished the kernel configuration, press **Q**, answer the question that appears and watch the system boot. On a slow system, you can watch the

kernel messages being issued and ensure that all of the desired devices have been found. Or, you can press the scroll-lock key when they begin to scroll, and when the kernel is done probing, you will be able to scroll the display up and down using the arrow keys and page up/page down.

You will now be presented with a text menu (in color, if you are on a color CGA or VGA monitor). (See Figure 1.)

Figure 1. The Initial Install Menu

Read the Documentation

The first item in the menu is “Usage”, which explains how to move through the menu system and which keys do what. This is a must-read for any first-time installer. Press **enter**, and you will be presented with the “HOW TO USE THIS SYSTEM” screen. (See Figure 2.)

Figure 2. Selecting Item 1 in the initial menu brings up this screen.

The next menu item is “Documentation”, which provides a brief overview of FreeBSD, the supported hardware, installation guide, etc. These files are available on the CD-ROM's root directory, as well as in the release's root directory in the FTP location.

Choosing Your Options

The third menu item is “Options”, and mostly applies to non-CD-ROM installs—NFS and FTP. In particular, if you need to use an FTP name other than **ftp** (e.g., anonymous or even a non-anonymous account name). (See Figure 3.)

Figure 3. Installation Process Options

The easiest way to get started is to choose the “Novice” installation method (the fourth item of the main menu). The first thing this does is partition the disk for you, using a screen-oriented **fdisk** program. The “Express” method isn't as verbose with explanations—and is probably the best way to install if you've done FreeBSD installs before. (See Figure 4.)

Figure 4. Express Install Screen

For simplicity's sake, I chose to use the entire disk for FreeBSD by typing **A**—it then asked if I wanted to have a “true partition” entry. This is necessary if the disk will be used in a mixed-OS, dual boot machine (e.g., both DOS and FreeBSD). Since the machine in question will only be used as a web server, I

answered no. (See Figure 5.) Note that if you are using BIOS geometry mapping, this may very well be required. As always, type **Q** when done.

Figure 5. Disk Geometry Screen

Partitions vs. Slices

FreeBSD can work with DOS-style partitions, and it can use its own partitions as well. FreeBSD calls the former “slices” in order to avoid confusion, although it doesn't necessarily succeed. In general, BSD partitions reside inside DOS-style partitions (aka “slices”). The normal name for a disk is <device><unit><partition>, e.g., wd0a; the slice is added after the unit, and before the petition. For example, wd0s1e would be the first slice (starting at 1, not 0), fifth partition within that slice, of the first IDE drive. FreeBSD can automatically partition the slice for you; on my 202MB drive, it chose:

```
/       32MB
swap    41MB
/var    30MB
/usr    98MB
```

You can choose your own sizes, of course. I chose the defaults which are quite reasonable.

After deciding on the layout of the disk, the next step is to choose which type of system to install. The options range from minimal to complete, with most people selecting something in between. For this install, the most likely type would have been “Basic”, which would install the basic FreeBSD system; however, I also prefer to configure my kernel to edit out unnecessary devices, so I chose the “kernel developer” package—this is the basic package, with compiler tools and the kernel sources. When installed, it used up approximately 130MB of disk space.

When selecting the package (by pressing the space bar), you are immediately asked if you want to install the DES packages. This is desirable, as you can share password file entries with traditional Unix systems this way. However, the default FreeBSD password encryption scheme (MD5 checksumming, actually) appears to be stronger than DES. Note that you are not supposed to install DES unless you are in the USA or Canada due to export restrictions, although the packages are included on the CD-ROM.

In addition to the basic DES package (the static and shared libraries), you can choose to install Kerberos (an authentication suite developed at MIT), as well as the sources to each. Although I generally use Kerberos, I did not install it on this machine, as space was getting tight and configuring Kerberos is not easy.

The install program then asks if you want to install the ports collection; this is fairly small (about 10MB), but since space was so tight I did not install it. There is more about ports and packages later later.

At this point, you are presented with the “Choose Distributions” menu again; if you are satisfied with your choices, press **return** to continue, otherwise, choose the distribution type you wish and continue.

Installation Media Choices

Figure 6. Media Choice Screen

The next choice is what kind of media to use for the install. (See Figure 6.) I chose to use the CD-ROM method, as it is faster, easier and more convenient than the others. However, you can also install via NFS and FTP (and passive FTP —this is required if you are behind a firewall that has been configured by a paranoid administrator). For FTP installs, it uses the account name chosen in the “Options” section mentioned previously. Last, you can also install via an existing file system (e.g., an MS-DOS file system), floppy or tape. To use tape, you must have one of the tape drives supported by FreeBSD—mostly SCSI tapes, but also Wangtek and a couple of others.

If you choose to do an FTP install, you have to select the site to grab the files from—the default is the “Primary Site”, which is ftp://ftp.freebsd.org (aka http://wcarchive.cdrom.com/). There are also mirrors around the world.

When doing an FTP or NFS install, you also need to configure the networking interface. You're presented with all of the networking interfaces that the system found—any networking cards it recognized, as well as SLIP, PPP and the parallel-port IP interface (PLIP). Help is available at the “Network interface information required” menu by pressing **f1**. One quick note: the SLIP and PLIP options assume that the connection will be a hard-wired connection—if you need to connect using a modem, PPP is the only possible method.

After selecting the network interface (e.g., ed0), you will need to tell the install program the host name and domain name, default router (aka the “gateway”), name server, IP address and any extra options. Note that the gateway and name server fields need to be IP addresses, not host names. You will need to enter this information again, when doing post-install configuration.

If you selected the PPP interface, you will be asked to configure it. This requires knowing what baud rate to use (it defaults to 115200) and the IP address of the remote side. By default, it uses the gateway address, if you supplied it; you can also tell it to use “0”, which will allow it to be negotiated as part of the PPP connection setup. After you've done all this, you are then told to switch to VTY3

(the third virtual console screen), where the PPP program has been started. From there, you need to connect to the PPP server you are using (e.g., dialing the modem, entering account and password information, etc.).

After that, all that was necessary was to wait while the system installed. On my slow 486DX-33, with an IDE drive and a double-speed SCSI CD-ROM drive, it took 16 minutes to install all of the packages.

System Configuration

The install program then asks if you'd like to configure the network devices and, if so, which ones. This is identical to what was done if a network installation of any sort was done. In the case of the install I did, there was only one interface to configure: ed0. FreeBSD prompts for host and domain names, network gateway, name server and IP address. The netmask defaulted correctly, although you can change it if necessary. There is also a box for “Extra options”—some cards may require link-level options to choose which interface pair, e.g., BNC or Twisted, to use.

The next questions asked are about, Samba, IP forwarding, anonymous FTP, and NFS configuration. Of these, the only one I chose to configure was anonymous FTP, as this is sometimes useful for a web server. If my network had more (or, for that matter, any) Windows systems, Samba would allow file and printer sharing. If the machine were going to be my router, I would have enabled IP forwarding.

The last three system configuration questions are system console configuration (e.g., screen saver, font, keyboard map, etc.), time zone and mouse. This particular machine does not have a mouse; if it did, it would be possible to enable text cutting and pasting.

Installing the Web Server

The last thing to do is install any desired packages. FreeBSD has quite a considerable set of packages and ports; this is, in fact, one of the most attractive attributes of FreeBSD, in addition to its high performance.

Ports vs. Packages

Ports and packages are very similar; the only difference is in what is included in the file. A package is a gzipped tar file containing all of the files needed, along with some description and checksum files. A port, on the other hand, consists of patches, and a pointer to the location on the Internet of the main files. Many ports are built on the local system after applying source patches. Some,

however, are “ports” because they are commercial programs and cannot be distributed via CD-ROM or Walnut Creek's FTP site.

The only package I chose to install was the Apache package in the WWW category. This took only a few seconds to install from CD-ROM, and it then went on to finish system configuration: additional accounts, setting the root password and registering. (Registering sends e-mail to the FreeBSD project and is not necessary. It does help the project, though.)

Once all that is done, the installation process is complete, and you can exit to reboot. When your machine comes back up, your FreeBSD system should now be on the network.

Post-install Configuration

As mentioned above, I always configure my kernels to trim away any unneeded devices. This is similar to what was done during the visual configuration process but is done by compiling the kernel, it results in a smaller kernel, requiring less memory.

The FreeBSD Handbook describes this process in detail. The Handbook is available on the FreeBSD web page, at http://www.freebsd.org/ and is also installed in /usr/share/doc/, in HTML. In simplest terms, you do the following commands:

```
cd /sys/i386/conf
cp GENERIC <machine name>
vi <machine name> #edit the file and exit vi
config <machine name>
cd ../../compile/<machine name>
make depend all
cp kernel /kernel
reboot
```

The complicated part is in the editing of the configuration file. After dealing with the visual configuration utility, the configuration file should not be all that complicated. (It is documented in the Handbook.) You can use the **dmesg** command to see which devices were found and which were not. By default, the installation leaves a copy of the generic kernel in /kernel.GENERIC; you can boot this, or any other kernel, by typing the name of the kernel at the Boot: prompt.

In addition to removing or configuring devices, system parameters can also be configured this way. One such parameter, **maxusers**, controls how much memory the kernel allocates to certain resources—the maximum number of processes, open files, and time events are all calculated based on **maxusers**. Another parameter that may need to be changed is **MAXMEM**—due to BIOS limitations, FreeBSD only recognizes up to 64MB of RAM by default (or 16MB on some very old systems), and MAXMEM (specified in KB) tells it to use more.

For example, on a machine with 256MB of RAM, which is expected to have a heavy load, the following lines in the configuration file might be used:

```
maxusers        100
options         MAXMEM="(256*1024)"   # 256MB
```

Once again, after editing the appropriate configuration file, run **config** and then **make**.

Making the Web Server Useful

The Apache package installs the configuration files into /usr/local/etc/apache, and the default configuration files have a document root of /usr/local/www/data. By creating an index.html file in that directory, the web server is now up.

For me, the machine was completely installed, configured and acting as a web server on my LAN in about two hours. Most of that time was spent waiting for the kernel to recompile; it took 90 minutes on this machine (it takes about six and a half minutes on my 133MHz Pentium)—and the system was working as a web server during that period.

Conclusions

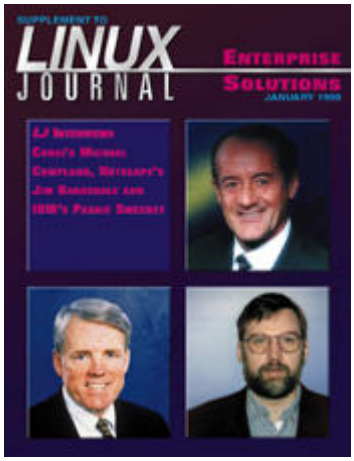
I have installed FreeBSD several times. The process is fairly painless, largely intuitive and very quick when done from a CD-ROM. My main objection is that it lacks a help option for many of the dialog boxes or menus; this can make it difficult to know what to do if you are new to Unix. However, ignoring that, the install went smoothly and required no knowledge of Apache configuration or installation on my part. If I hadn't chosen to reconfigure the kernel, I would have had a fully-functioning web server within about 30 minutes of beginning installation.

Resources

Sean Fagan has been a BSD contributor for many years. He lives in San Jose with a psychotic cat who insisted on being mentioned in this article. He can be reached at sef@kithrup.com.

Archive Index Issue Table of Contents

Advanced search



Enterprise Solutions Supplement

Features

LJ Interviews Corel's Michael Cowpland *by Marjorie Richardson*
A talk with the President of Corel, a man and a company fully comitted to Linux.

LJ Interviews Netscape's Jim Barksdale *by Marjorie Richardson*
Netscape's CEO talks about Open Source and Linux and why his company supports both.

LJ Interviews IBM WebSphere's Paraic Sweeney *by Marjorie Richardson*
IBM now supports Open Source by shipping products with the Apache web server.

News & Articles

Linux as POS for Pizza Business *by Steve O'Connor*
Here's a new use for Linux—selling pizzas—and about time, too.

Linux-Kontor Accountancy Package *by Joachim Schaaf*
Mr. Schaaf describes the concept and current development stage of this free program for the commodity market.

Product Review

xxl: A Spreadsheet for Linux *by Larry Ayers*
The intent of xxl is to produce a graphical spreadsheet which is both uncomplicated and easy to learn and use.

Departments

From the Editor
Linux and Enterprise: A Winning Combination *by Marjorie Richardson*