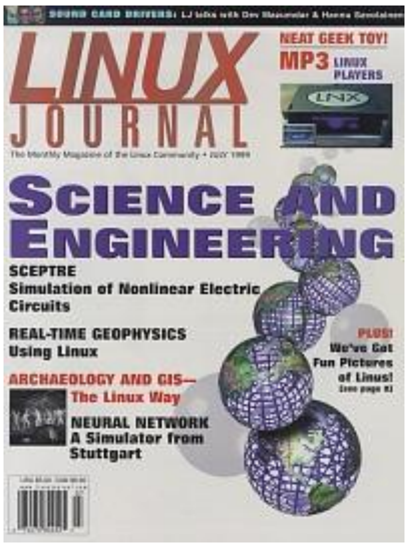
Advanced search

Linux Journal Issue #63/July 1999



Focus: Science & Engineering

Marjorie Richardson

Issue #63, July 1999

Linux has been embraced from the beginning by research departments in universities, business and government.

New technology has always found a home in the science labs of universities and the research departments of both scientific and engineering companies. Schools and new companies looking for their niche in the marketplace often have restricted budgets and a need for highly robust systems. For both these reasons, Linux has been embraced from the beginning by research departments in universities, business and government.

I have always had in interest in science. My degree is a B.S. (math and physics), one of my hobbies is astronomy, my job for many years was programming geophysical applications, and my husband is a physicist. As a result, our Science & Engineering issue is always one of my favorites. I enjoy reading about the cool ways Linux is being put to use in these fields. And this issue is no exception—we even have an article about geophysics.

Ed Petron's article on teaching computers to think brings science fiction to everyday reality. This new method of programming is a quite a step from the usual algorithmic approach. Speaking of science fiction, take a look at the pictures of Fermilab and the great article by Jon Hall.

Inside, Wolf-Rainer Novender describes SCEPTRE, a simulation tool for electric circuits, and on the Web, Alasdair McAndrew gives us a comprehensive tutorial for the mathematical tool, MuPad. Also in “Strictly On-line” are articles about using GPS technology to do precision farming and calculating underground water quality using parallel algorithms. Two unique uses of Linux that I would never have dreamed up.

As long as students continue to be exposed to Linux at school in their science labs, Linux will continue to make inroads into engineering and scientific applications.

Caldera Lizard

Ransom Love of Caldera dropped by our office in April to give us a copy of their new release, OpenLinux 2.2. I installed it on a test machine to see if the much-touted “new and easy” Lizard install truly worked. Basically, it did, but I had one problem: the install was hanging while probing for the SCSI device, a 2-channel UW Adaptec on-board controller. A message from Caldera support recommended I use the boot parameter **er=cautious**. I did, and it worked. I also had to use the custom install option and define partitions, since the machine I was using already had the Be OS installed. If the OS had been Windows, Lizard would have automatically built the partitions using PartitionMagic. Even with the custom install, the entire procedure took about ten minutes and I had a working Linux system with KDE, WordPerfect8, Star Office and other goodies. It was so fast, that by the time it offered to let me play Tetris, the installation was complete. When Linux detractors say Linux needs an easy install, this is what they want. We'll have a full review next month.

Marjorie Richardson, Editor in Chief

Stuttgart Neural Network Simulator

Ed Petron

Issue #63, July 1999

Exploring connectionism and machine learning with SNNS.

Conventional algorithmic solution methods require the application of unambiguous definitions and procedures. This requirement makes them impractical or unsuitable for applications such as image or sound recognition where logical rules do not exist or are difficult to determine. These methods are also unsuitable when the input data may be incomplete or distorted. Neural networks provide an alternative to algorithmic methods. Their design and operation is loosely modeled after the networks of neurons connected by synapses found in the human brain and other biological systems. One can also find neural networks referred to as artificial neural networks or artificial neural systems. Another designation that is used is “connectionism”, since it deals with information processing carried out by interconnected networks of primitive computational cells. The purpose of this article is to introduce the reader to neural networks in general and to the use of the Stuttgart Neural Network Simulator (SNNS).

In order to understand the significance of the ability of a neural network to handle data which is less than perfect, we will preview at this time a simple character-recognition application and demonstrate it later. We will develop a neural network that can classify a 7x5 rectangular matrix representation of alphabetic characters.

Figure 1. Ideal Letter “A”

In addition to being able to classify the representation in Figure 1 as the letter “A”, we would like to be able to do the same with Figure 2 even though it has an extra pixel filled in. As a typical programmer can see, conventional algorithmic solution methods would not be easy to apply to this situation.

Figure 2. Blurred “A”

Neural Network Principles of Operation

A neural network consists of an interconnected network of simple processing elements (PEs). Each PE is one of three types:

- Input: these receive input data to be processed.
- Output: these output the processed data.
- Hidden: these PEs, if used in the given application, provide intermediate processing support.

Connections exist between selected pairs of the PEs. These connections carry the output in the form of a real number from one element to all of the elements to which it is connected. Each connection is also assigned a numeric weight.

PEs operate in discrete time steps **t**. The operation of a PE is best thought of as a two-stage function. The first stage calculates what is called the net input, which is the weighted sum of its input elements and the weights assigned to the corresponding input connections. For the **j**th PE, the value at time **t** is calculated as follows:

$$net_j(t) = x_1(t)w_{1,j} + x_2(t)w_{2,j} + \cdots x_n(t)w_{n,j} = \sum_i x_i(t)w_{i,j}$$

where **j** identifies the PE in question, **xi(t)** is the input at time **t** from the PE identified by **i**, and **wi,j** are the weights assigned to the connections from **i** to **j**.

The second stage is the application of some output function, called the activation, to the weighted sum. This function can be one of any number of functions, and the choice of which one to use is dependent on the application. A commonly used one is known as the logistic function:

$$F_j(x) = \frac{1}{1+e^{-x}}$$

which always takes on values between 0 and 1. Generally, the activation **Aj** for the **j**th PE at time **t+1** is dependent on the value for the weighted sum **netj** for time **t**:

$$A_j(t+1) = F_j(net_j(t))$$

In some applications, the activation for step **t+1** may also be dependent on the activation from the previous step **t**. In this case, the activation would be specified as follows:

$$A_j(t+1) = F_j(A_j(t), net_j(t))$$

In order to help the reader make sense out of the above discussion, the illustration in Figure 3 shows an example network.

<u>**Figure 3. Sample Neural Network**</u>

This network has input PEs (numbered 1, 2 and 3), output PEs (numbered 8 and 9) and hidden PEs (numbered 4, 5, 6 and 7). Looking at PE number 4, you can see it has input from PEs 1, 2 and 3. The activation for PE number 4 then becomes:

$$A_4(t+1) = F_4(net_4(t)) = F_4(x_1(t)w_{1,4} + x_2(t)w_{2,4} + x_3(t)w_{3,4})$$

If the activation function is the logistic function as described above, the activation for PE number 4 then becomes

$$A_4(t+1) = \frac{1}{1 + e^{-(x_1(t)w_{1,4} + x_2(t)w_{2,4} + x_3(t)w_{3,4})}}$$

A typical application of this type of network would involve recognizing an input pattern as being an element of a finite set. For example, in a character-classification application, we would want to recognize each input pattern as one of the characters A through Z. In this case, our network would have one output PE for each of the letters A through Z. Patterns to be classified would be input through the input PEs and, ideally, only one of the output units would be activated with a 1. The other output PEs would activate with 0. In the case of distorted input data, we should pick the output with the largest activation as the network's best guess.

The computing system just described obviously differs dramatically from a conventional one in that it lacks an array of memory cells containing instructions and data. Instead, its calculating abilities are contained in the relative magnitudes of the weights between the connections. The method by which these weights are derived is the subject of the next section.

In addition to being able to handle incomplete or distorted data, a neural network is inherently parallel. As such, a neural network can easily be made to take advantage of parallel hardware platforms such as Linux Beowulf clusters or other types of parallel processing hardware. Another important characteristic is fault tolerance. Because of its distributed structure, some of the processing elements in a neural network can fail without making the entire application fail.

Training

In contrast to conventional computer systems which are programmed to perform a specific function, a neural network is trained. Training involves presenting the network with a series of inputs and the outputs expected in each case. The errors between the expected and actual outputs are used to adjust the weights so that the error is reduced. This process is typically repeated until the error is zero or very small.

Training methods vary greatly from one application to the next, and there is no single universal solution. As an example, we turn to a general discussion of what is known as gradient descent or steepest descent. Here, the error for each training pattern is quantified as:

$$E_p = \frac{1}{2}\sum_{k=1}^{M} \delta_{pk}^2$$

where **Ep** is the error for pattern **p** and

δ
pk is the difference between the expected and actual outputs for pattern p and output PE **k**. The error **Ep** can also be thought of as a scalar valued function of the set of connection weights in the network:

$$E_p = F(w_1, w_2, \cdots, w_n$$

With this in mind, minimizing **Ep** involves moving the weights in the direction of the negative gradient **-**

∇
Ep. The weight change for a selected weight **wi** can be calculated as:

$$\Delta w_i = -\eta \frac{\partial E_p}{\partial w_i}$$

where **nu** is some constant between 0 and 1. The function F is typically chaotic and highly nonlinear. That being the case, the actual gradient component may be a very large value that may cause us to overshoot the solution. The constant **nu** can be used to suppress this.

I have included the above discussion mostly for the benefit of readers with some basic knowledge of multi-variable calculus. For others, it is really only important to know that, through an iterative process, a neural network is adapted to fit its problem domain. For this reason, neural networks are considered to be part of a larger class of computing systems known as adaptive systems.

Prototyping with SNNS

Although the actual implementation and operation of a neural network can be accomplished on a variety of platforms ranging from dedicated special-purpose analog circuits to massively parallel computers, the most practical is a conventional workstation. Simulation programs could be written from scratch, but the designer can save much time by using one of several available neural network prototyping tools. One such tool is the Stuttgart Neural Network Simulator (SNNS).

A Demonstration of SNNS

The easiest way to get started with SNNS is to experiment with one of the example networks that comes with the distribution. One of these is a solution to the character recognition problem discussed at the beginning of this article.

Upon invoking SNNS, the manager (Figure 4) and banner (Figure 5) windows appear.

Figure 4. SNNS Manager

Figure 5. SNNS Banner

Selecting the file option from the manager window presents the user with a file selection box (Figure 6).

Figure 6. SNNS File Selection

The file selector uses extensions of .net, .pat, etc. to filter the file names in the selected directory. We will load the letters_untrained.net file, since we want to see the training process in action. We will also load the letters.cfg configuration file and the letters.pat file which contains training patterns.

After the files are loaded, selecting the display option in the manager window will present the user with a graphical display of the untrained network (Figure 7).

Figure 7. SNNS Untrained Network Display

This window shows the input units on the left, a layer of hidden units in the center and the output units on the right. The output units are labeled with the letters A through Z to indicate the classification made by the network. Note that in the Figure 7 display, no connections are showing yet because the network is untrained at this point.

Figure 8. SNNS Control Window

Selecting the control option from the manager window presents the user with the control window (Figure 8). Training and testing are directed from the control window. Training basically involves the iterative process of inputting a training vector, measuring the error between the expected output and the actual output, and adjusting the weights to reduce the error. This is done with each training pattern, and the entire process is repeated until the error is reduced to an acceptable level. The button marked ALL repeats the weight adjustment process for the entire training data set for the number of times entered in the CYCLES window. Progress of the training can be monitored using the graph window (Figure 9).

Figure 9. SNNS Graph Window

In the graph window, the horizontal axis shows the number of training cycles and the vertical axis displays the error.

Figure 10. Partially Trained Network

Figure 10 shows a partially trained network. In contrast to the untrained network in Figure 7, this picture shows some connections forming. This picture was taken at the same time as Figure 9. After enough training repetitions, we get a network similar to that shown in Figure 11. Notice that the trained network, when the letter A is input on the left, the corresponding output unit on the right is activated with a 1.

Figure 11. Trained Network

As a quick check to see if the network can generalize, a modified version of the training data set is tested with one of the dots in the A matrix set to zero instead of one, while an erroneous dot is set to one instead of zero. Figure 12 demonstrates that the distorted version of the letter A is still recognized.

Figure 12. Test of Distorted A

As pointed out in the section on training, many different methods can be used to adjust the connection weights as part of the training process. The proper one to use depends on the application and is often determined experimentally. SNNS can apply one of many possible training algorithms automatically. The training algorithm can be selected from the drop-down menu connected to the control window.

SNNS reads network definition and configuration data from ASCII text files, which can be created and edited with any text editor. They can also be created by invoking the **bignet** option from the manager window. **bignet** enables the creation of a network by filling in general characteristics on a form. Refinements can be made by manually editing the data files with a text editor or by using other options within SNNS. Training and test data files are also plain ASCII text files.

Other notable features of SNNS include:

- Remote Procedure Call (RPC)-based facility for use with workstation clusters
- A tool called **snns2c** for converting a network definition into a C subroutine
- Tools for both 2-D and 3-D visualization of networks

Getting and Installing SNNS

Complete instructions for getting and installing SNNS can be found at http://www.informatik.uni-stuttgart.de/ipvr/bv/projekte/snns/obtain.html. Prepackaged binaries are also available as part of the Debian distribution. Check http://www.debian.org/ for the Debian FTP site nearest you.

Summary

Neural networks enable the solution of problems for which there is no known algorithm or defining set of logical rules. They are based loosely on neurobiological processes and are thus capable of decisions which are intuitively obvious to humans but extremely difficult to solve using conventional computer processes. They are also less brittle and prone to failure than conventional systems due to their distributed nature. Their inherent parallelism provides opportunities for highly optimized performance using parallel hardware. The Stuttgart Neural Network Simulator (SNNS) is a powerful tool for prototyping computer systems based on neural network models.

Resources

Ed Petron is a computer consultant interested in heterogeneous computing. He holds a Bachelor of Music from Indiana University and a Bachelor of Science in computer science from Chapman College. His home page, The Technical and Network Computing Home Page at www.leba.net/~epetron, is dedicated to

Linux, the X Window System, heterogeneous computing and free software. Ed can be reached via e-mail at epetron@leba.net.

Archive Index Issue Table of Contents

Advanced search

Archaeology and GIS—The Linux Way

R. Joe Brandon

Trevor Kludt

Markus Neteler

Issue #63, July 1999

A description of an archaeology project making use of the freely available geographic information system GRASS.

Since the days of Heinrich Schliemann's search for Troy, archaeologists have been confronted with the dilemma of how to record the spatial characteristics of archaeological data, and once recorded, how to analyze those data. This spatial information has historically been recorded on paper maps of varying accuracy and scales. When researchers wanted to perform analyses of these data, they were required to spend hours, if not days, transposing this information to new paper maps and making arduous measurements by hand. This time-consuming process is nearing its end as researchers have moved to take advantage of geographic information systems (GIS) for spatial analysis. A GIS is best thought of as a dynamic database for spatial data. Fortunately for budget- and quality-conscious researchers, one of the oldest and most robust GIS packages, GRASS, is available for free on Linux.

In the late 1970s, the method of storing archaeological data began to slowly change from paper to digital. As access to affordable computers became more common, archaeologists could see the advantages of digitally storing their spatial data for analysis. During this era, several commercial vendors began to develop primitive GIS systems in an attempt to meet the needs of researchers for storing spatial data. Unfortunately, these products worked only with vector data, completely ignoring the utility of raster data for analysis. The Construction Engineering Research Laboratory (CERL) of the U.S. Army Corps of Engineers found this oversight in the commercial packages too great and decided the solution was to build their own UNIX-based GIS which would meld vector and raster analysis capabilities. Because this code was written with federal dollars, it was made freely available to the public. From this project, the GIS package Geographical Resource Analysis Support System, more commonly known by the acronym “GRASS”, was born. From its beginnings in 1982, GRASS has grown into a powerful GIS package that rivals expensive commercial packages, yet remains free to the public.

The Project

The McGregor Guided Missile Range, located on Ft. Bliss in western Texas and southern New Mexico, is just a bit smaller than the state of Delaware. As part of the army's long-standing commitment to effective management and research, Ft. Bliss hired the Archaeological Research Center (ARC) at the University of Texas-El Paso to do a 100% pedestrian survey of 180 square kilometers of the McGregor Range. Principal Investigator Tim Church and Ft. Bliss Lead Archaeologist Galen Burget realized that utilizing old-fashioned survey methods, which are primarily paper-based, would not be feasible for a survey of this size. Furthermore, the research team was keenly interested in utilizing spatial environmental and ecological information collected by other scientists at the post for a landscape analysis.

Archaeologists/computer jockeys Trevor Kludt and R. Joe Brandon were hired to help develop the most efficient method of dealing with the anticipated high volume of data, and to insure this data would be compatible with existing ecological data. R. Joe also directed the archaeological field crew during the project to monitor the feasibility of the data acquisition methodologies being developed. This hands-on approach allowed Trevor and R. Joe to tweak the methodology for collecting data so that all aspects of the project were integrated seamlessly.

The research team decided from the outset that a flexible and powerful GIS package was critical if we were to accomplish our goals. Considering that virtually no money was in the budget for additional hardware or software and the whole show would have to be run on a single mid-level Pentium PC (Gateway P-133 with 2GB hard disk and 32MB memory), we were in a bind.

After considering a number of commercial GIS packages, we determined that GRASS running under Linux was the answer. GRASS offered the full suite of sophisticated GIS functions this project demanded, while Linux provided us with a wide range of tools to tackle the myriad customizations we knew were inevitable. Once the decision was made, we had Linux (Red Hat 4.0) up and running in no time, then downloaded the compiled binaries of GRASS (v4.2). This was all done for about what the field crews would spend on refreshments after a day surveying in the desert heat.

Knowing from the outset that our field data were to be integrated with raster (cell)-based environmental and ecological data, the research team decided to develop a raster-based field methodology for the survey. At first blush, this decision seemed very practical. In practice, it turned out to be the biggest challenge of the project. Traditional field methods focus primarily on defining the boundaries of archaeological concentrations (sites) and are therefore vector-based. The methods we were developing, focusing on the distribution of materials within the grid units or “cells”, were raster-based. How could we put these individual raster units together to define our sites?

For speed, flexibility and consistency, we decided to develop computer routines to do this monotonous work for us. As we needed these routines quickly, initial prototyping was done using Qbasic while Linux and GRASS were being set up. Eventually, we translated these routines into a comprehensive Tcl script suite and a number of Bash shell scripts. These scripts allowed us to automate the importation of the field-collected data, dynamically define site boundaries, provide reports on site inventories, import data into GRASS, run a suite of spatial tests and comparisons, and output the finished site and project maps. Without the power and stability of Linux and the interoperability of GRASS, these tasks would have taken a significant amount of time.

Figure 1. Single Survey Unit Results

Fieldwork proceeded rapidly. Each of the 180 square kilometer survey areas was divided into a grid of 16m2 “cells” which matched the size of the analysis raster cells within GRASS. Figure 1 shows the results of a single km2 survey unit, displaying artifact distribution, site boundaries and a 100-meter grid. The site boundaries were created using the GRASS module **r.to.vect**. The map output was created in a script file that can be sent to the screen or the printer. The size of the cells, 16m on a side, was a trade-off between the desire for precision (smaller units) and efficiency (larger units). Field crews would traverse these survey areas using 1:3,000-scale aerial photographs with the 16m2 grid superimposed on the photo. The detail of these air photos allowed the crew chief to navigate to near sub-meter precision quickly and consistently. As the crews walked through the survey areas, crew members would call out when they had spotted an artifact or feature, and the crew chief would then call back the corresponding row and column number for that “cell”. This information was then noted on a data recording form. In the lab, this information was checked for accuracy and errors, entered into a DOS-based data file (on a 386 with 40MB hard disk and 2MB memory salvaged from the scrap heap), then migrated to the Linux environment.

Figure 2. GRASS-generated Results

As the project unfolded, it became clear that the sheer volume of incoming data would give us no rest. Each day, the six field crews were covering thousands of cells on the ground and sending in mountains of data. Figure 2 shows the display results generated by GRASS for a small portion of the study area. This displays the variety of survey units, roads, site boundaries, artifacts and features. The map output was created in a script file that can be sent to the screen or the printer. Since development of the data processing routines coincided with the start of fieldwork, we quickly developed a backlog of data to process. And since each of the 180 survey areas was treated the same, we counted on the power of scripting redundant tasks to increase our efficiency in keeping up with the data stream. Once our routines were tested, debugged and on-line, we caught up quickly and were able to keep pace with the field crews. In the end, we accomplished what had at times seemed impossible. We are convinced the powerful scripting and developmental tools available under Linux, coupled with the sophisticated GIS routines available under GRASS, enabled us to successfully tackle the difficult and interesting challenges encountered during our project. Equally important, we were able to complete this project on time and under budget while creating one of the most dynamic spatial archaeological databases anywhere. In retrospect, we realized that even with a greater budget, we could not have fared better than we did utilizing the strengths of Linux and GRASS.

Future Directions

Although we were able to automate many of the tasks associated with survey work, we were never able to realize the full potential of the Linux/GRASS combination to automate virtually all of these tasks in concert with each other. We feel that a future project could be designed building on the work completed so far. This project could include equipping field crews with differentially corrected GPS receivers and data loggers to record and correct field data in real time. These data could then be downloaded at the end of the day and fed into the Postgres database engine which would provide seamless data entry and manipulation. Text formatting packages such as nroff/troff or LaTeX would be used to automate the production of site-specific reports and documentation. GNUplot could be used to produce the necessary graphs and charts. GRASS would be used for spatial display analysis.

Linked via scripts, we envision a time when field data collected in this manner could be downloaded, and within an hour, fully formatted and detailed site reports and maps would begin to pop out of the printer. Considering that with traditional methods the time gap between the input of raw data and the finished site-specific documentation is most often measured in weeks or months, we feel that this next level of integration is not only feasible, but advisable. By harnessing the tools Linux has made freely available, we believe there is much to offer researchers in this arena. We are glad to see development for both Linux and GRASS is so actively promoted by their respective communities and look forward to utilizing these tools to their fullest potential.

GRASS Applications for Other Researchers

Although the project described above is of interest mainly to other archaeologists, the availability of a powerful GIS package for Linux is of interest to a larger segment of the Linux community, notably geographers, geologists and other researchers dealing with spatial data.

Development of GRASS languished for several years in the mid-90's after funding for future development at CERL came to a halt. There was concern among the international GIS community that GRASS, the father of raster analytical GIS packages, might finally be on its way to that great clippings bag in the sky. That fear was ended in November 1997 when the “GRASS Research Group” was established at Baylor University in Texas. This group pledged to continue development of GRASS. Interested in the ongoing maintenance and development of the GRASS GIS platform, they received the copyright for GRASS from CERL.

Coinciding with this renewed support for GRASS was the meteoric rise in the numbers of Linux GRASS users around the world, coupled with the availability of CPUs with sufficient speed to handle the volumes of data robust GIS package can deliver. The authors feel it has been the growth of the international Linux community that has been a major catalyst in giving GRASS a new lease on life. As Baylor continues to blaze new trails, a large audience of users eagerly awaits each update. Today, development is actively continuing, much of it pushed forward by users in the Linux community. Due to the stability, speed and power of Linux and its ability to turn even a modest PC into a strong UNIX workstation, GRASS has found a place in the Linux user community.

Since 1998, Linux GRASS has been developed by Markus Neteler, and the code is freely available at http://www.geog.uni-hannover.de/grass/ and several mirror sites. Additional UNIX flavors of GRASS are available from this site. Monthly updates of the GRASS 4.2.1 packages are published out of Hannover. Beside error corrections, these updates include new modules from independent GRASS programmers that have been collected from the international community and integrated into the system package.

A Great GUI for GRASS

GRASS was the last major GIS package to be completely command-line driven. A recent major improvement in GRASS has been the creation of a platform-independent graphical user interface based on the freely distributed Tcl/Tk libraries. The main work for this undertaking has been by Jacques Bouchard in France. This interface, TCLTKGRASS, gives users a clean GUI interface for GRASS commands and modules, as well as access through the standard command line. This GUI compliments the key strength of GRASS—it operates at the system level, allowing users to utilize the full suite of Linux utilities in conjunction with GRASS.

Current Linux GRASS Version

Figure 3. Sub-Pixel Analysis Results

The current release of GRASS 4.2.1 now comes with more than 350 modules. In addition to standard GIS functions for raster, vector and point data, it offers statistical and image processing capabilities. Markus has recently written an algorithm, called "spectral mixture analysis", that performs sub-pixel analysis and true coverage degrees in percent. Figure 3 shows the results from a module for GRASS that allows the user to perform sub-pixel analysis on vegetation and soil types with the spectral mixture analysis module. The left map shows the covered soils in percent. The right map shows the difference between a remotely sensed image and ground-truthed imagery. Black indicates no difference, red indicates high difference. Known field errors account for the red in the middle. This module can be used for geological analysis as well as detection of minerals based on their spectra. The vector lines in this image were created using the module **r.line** on the remotely sensed image, which was much faster than digitizing. This figure was created using **xfig**, a free drawing tool with good map scale features. **xfig** is supported in GRASS through a vector and raster exporting module. GRASS users can also rectify off-nadir aerial images taken with a hand-held camera from a small plane and convert them into quasi-orthophotos usable for area or distance calculations or mapping purposes. An overview of Linux GRASS, the tutorials and sample data are also available from Markus's University of Hannover, Germany, web site at www.geog.uni-hannover.de/grass/welcome.html.

GRASS 5.0—The Next Generation

On Feb. 1, 1999, Baylor and Hannover simultaneously released the GRASS 5.0 beta. This was the first major upgrade to GRASS in a number of years. The demand was so high that downloads at Baylor accounted for 20% of the entire university Internet traffic. To date, several thousand unique downloads of Linux GRASS have been made from the Hannover site, which is currently transferring 5GB a week. This volume is steadily increasing. A stable release of GRASS 5.0 is planned for early summer—by the time you read this. Currently, bug reports are being collected and fixes applied to the code. The code is also being modified to make GRASS 64-bit compliant.

Figure 4. NVIZ, a 3-D Visualization Tool

The most significant new features of GRASS 5.0 are floating-point support in raster format and an improved sites format. Because all raster modules have to be rewritten, this is a major step in GRASS development. For the stable 5.0 version, several other projects are on the list. The NVIZ tool, a robust three-dimensional visualization tool, is currently being ported to Linux and other UNIX systems from SGI to aid in viewing 3-D GRASS data with raster/vector/sites overlaying features (Figure 4). This tool allows users to display 3-D raster data as stacked layers, with raster, vector and sites draped as overlays. It will be a very useful (and much sought-after) tool for data visualization.

To encourage user development of GRASS routines, the GRASS 5.0 programmer's tutorial will be available very soon. In the near future, the growth of GRASS will include introducing a new vector format. This is part of the long-term plan to continue the evolution of GRASS to a true 3-D/4-D GIS system. It will incorporate an improved 3-D raster format and new 3-D vector formats.

Linux GRASS is Spreading

A friend recently summed up the future of GRASS and Linux quite well. He was leaving a research position to go work for a state agency using their expensive off-the-shelf GIS and remote-sensing packages; however, he had also worked with GRASS on a Sun box. Before he left, he sat down with me to get some information on setting up a Linux box on his office Wintel machine. As he said, “I need to be able to get the real spatial work done on a stable platform that is not going to crash; those commercial packages on Wintels are just too buggy.” When I sent him a copy of this last paragraph now that he has been in the office for some months, he responded, “You can say that again and again and again....” One more sign that the Linux juggernaut continues on its way into the workplace.

R. Joe Brandon (rjoe@cast.uark.edu) is completing his M.A. in archaeology at the University of Arkansas, where he works at the Center for Advanced Spatial Technologies (CAST). He first began working with GRASS in early 1993. He runs MkLinux at home. He enjoys spending time relaxing and traveling with his fiancée Kristy, playing with his cats, B&W photography and web development.

Trevor Kludt (tkludt@unm.edu) is working on his dissertation in archaeology at the University of New Mexico. His interests include GIS and spatial modeling applications in archaeology and development of new methodologies which take advantage of the power of GIS. Trevor has been using Linux and GRASS since 1995. When not doing archaeology, he likes to travel or tend his medicinal herb garden.

Markus Neteler (neteler@geog.uni-hannover.de) recently received his M.Sc. in Physical Geography at the University of Hannover, Germany. The focus of his work is on GIS/Remote Sensing using GRASS. Markus is also a member of the GRASS Research Group at Baylor University. GRASS on a Linux box is his recommendation for researchers today and into the future. Playing oboe in a classical symphony orchestra compensates for his computer work.

Archive Index Issue Table of Contents

Advanced search

Advanced search

Real-Time Geophysics Using Linux

Laura Connor

Charles B. Connor

Issue #63, July 1999

How the geophysical industry is using Linux for processing of magnetometer data.

Whether searching for gold or oil, or assessing the geologic hazards of a site, geophysics is a high-stakes business. Geophysical exploration involves learning more about the earth's interior by making measurements at the surface using instruments such as seismometers, gravity meters and magnetometers. At Southwest Research Institute, our team of geologists and geophysicists works worldwide to assess volcanic hazards using geophysical techniques (Figure 1).

Figure 1. Geologists Team

We discovered that Linux provides a robust, low-cost system for visualizing and processing magnetometer data in real time. Application of this real-time, data processing/visualization system enables us to create better geophysical maps for use in volcanic hazard assessments. This technique illustrates the potential for Linux applications in a variety of geophysical surveys.

Figure 2. R-T-V Base Station

Our magnetic mapping system consists of three components: a mobile instrument platform, a GPS base station and a real-time visualization (R-T-V) base station (Figure 2).

The usual instrument platform includes a cesium-vapor magnetometer carried by one person and GPS and telemetry equipment carried by another (Figure 3), but alternatives are possible. We have made magnetic anomaly maps by securing this instrumentation to a mountain bike and pedaling across the survey area (Figure 4). Currently, we are planning to mount the instruments on an unmanned aerial vehicle (UAV) to fly surveys at low altitudes (Figure 5). This flexibility in survey design becomes possible because all data processing, data storage and data monitoring tasks can be handled at the R-T-V base station.

Figure 3. Telemetry Equipment

Figure 4. Equipment Attached to Bike

Figure 5. Unmanned Aerial Vehicle

Geophysical Application

Anomalies in the earth's magnetic field are produced by variations in the magnetic properties of rocks within the earth's subsurface. Magnetic surveying is a particularly good method of mapping subsurface volcanic rocks because these rocks have strong magnetic properties. Creating a map of magnetic anomalies helps to locate buried volcanos and volcanic intrusions that are not seen in the surface geology (Figure 6). Therefore, accurate magnetic mapping represents a critical step in determining the volcanic history of a region as part of a volcanic hazard analysis.

Figure 6. Magnetic Anomaly Map

A geophysical instrument called a magnetometer is used to collect the data needed to generate a magnetic anomaly map. Magnetometers measure small changes in the intensity of the earth's magnetic field. A geophysical survey crew transports the magnetometer across a study area, while continually taking magnetic measurements. During our surveys, the crew also carries a global positioning system (GPS) to track the locations of these measurements. Magnetic and GPS data are radio-telemetered to a base station where software, running under the Linux operating system, is used to visualize the magnetic data as they are collected. This base station also monitors the location of the magnetometer, then processes and stores these data. With real-time visualization, the survey crew gains quick insight into the magnetic anomalies being mapped. Using this information, survey teams can concentrate their time and energy mapping areas that reveal significant details about the subsurface geology. Thus, real-time visualization of magnetic measurements can greatly facilitate the search and discovery process inherent in geophysics.

The cesium-vapor magnetometer is very sensitive, capable of measuring a change of one part per million in the intensity of the earth's magnetic field. This magnetometer can sample at very high rates, 1Hz to 50Hz. Such high data-collection rates are useful because we interface the magnetometer with a real-time kinematic differential GPS, capable of determining the mobile instrument platform's position every second to within several centimeters. This high accuracy is accomplished by using a stationary GPS base station to radio-telemeter location corrections to the GPS carried by the mobile survey team (Figure 7). Magnetic readings become interleaved with GPS location readings as the magnetometer's data collection facility stores the data from both instruments.

Figure 7. Mobile Survey Team

Implementing Real-Time Visualization

Simplicity and reliability are significant elements of the R-T-V base station. Physically, the base station consists of a 9600 baud, 2 watt radio modem, interfaced via a serial port connection to a 166MHz laptop computer with 24MB of RAM running Linux. The computer and radio are both powered by a 12-volt battery. The radio requires an antenna to provide line-of-site communication, sometimes augmented by a repeater, with the mobile survey team's radio. The R-T-V base station program waits for data at the serial port and selects, for visualization and processing, only those data strings tagged appropriately by the magnetometer and transmitter radio. Non-tagged or unrecognizable transmissions are discarded.

A number of events are programmed to occur as data arrive at the R-T-V base station:

- The GPS RF data transmission is associated with the corresponding magnetometer RF data transmission. This step involves interpolation of the GPS data to calculate the exact position of the magnetometer sensor.
- The actual field transmissions are logged to a file and can be viewed via a “chat” (text) window on the laptop's display.
- Regional effects on the magnetic readings are removed using the International Geomagnetic Reference Field correction, and the values are displayed on a scalable strip-chart showing the change in magnetic values over time.
- The positions of the survey team are transformed from latitude and longitude and plotted as x-y coordinates on a scalable, two-dimensional map using a Universal Transverse Mercator (UTM) map projection. These applied data corrections simplify the task of data interpretation and map generation for the survey team.
- Finally, the converted data are saved to disk.

Figure 8. Setup Screen

Figure 9. A Look at the Data

XForms, a graphical user interface (GUI) toolkit based on Xlib for the X Window System, was chosen as the GUI builder for the visualization. The use of XForms' strip-chart object and two-dimensional plot object facilitated rapid program development. The WYSIWYG (what you see is what you get) interactive form designer made it possible to visually construct each layout for the displays (Figures 8 and 9). A timed call-back routine specifies when to check the serial port for data. Mouse events are handled between serial port reads. This allows for simultaneous screen updates and GUI control, with no noticeable delay to the user. During a survey, magnetic reading failure at the magnetometer is indicated by the stopping of updates to the strip-chart display, and an invalid GPS position is indicated by a color change on the two-dimensional location map.

Use of the R-T-V base station has mitigated many of the difficulties that frequently accompany geophysical surveys. Prior to using the R-T-V base station, our surveys relied on downloading saved magnetic-GPS data from the magnetometer for later post-processing and map generation. Post-processing took up to an additional month after returning home from the field before a magnetic map could be realized and the quality of the collected data evaluated. Sometimes the most interesting magnetic anomaly was just meters away from the survey area, but there was no way of knowing this until weeks later after post-processing. Equipment and power failure remain an ongoing frustration when using batteries and multiple cable connections in the desert and other desolate environments. Real-time monitoring allows for quick problem diagnosis by the whole team, not just the one or two team members struggling to complete the survey. This is especially important if the surveyor has little experience with the equipment, the survey conditions are rugged, or the equipment happens to be mounted on a UAV or a bicycle. For example, one common problem is the loss of differential correction transmissions from a GPS base station. When this happens, the mobile GPS receiver cannot calculate accurate positions, and magnetic readings not precisely located are useless. This situation is often not apparent to the mobile part of the survey team. At the R-T-V base station, we specifically monitor differential GPS quality and can alert the mobile team immediately when a problem occurs.

In practice, our Linux system is able to read RF transmissions smoothly, process and feed data to the displays, and write to the data files in a seamless flow. It is even possible to switch to another virtual desktop and perform various editing or file management tasks while the real-time processing is taking place. Switching back to the real-time display is equally smooth. We were quite satisfied with the performance of our R-T-V system. Much of its success can be attributed to the stability of the Linux operating system. Future enhancements include visualizing multiple surveys concurrently, adding color-coded symbols to the location map to indicate magnetic values, loading previous survey information, performing a real-time magnetic drift correction on the data, and processing transmissions from other geophysical instruments.

Resources

Acknowledgments

Laura Connor is currently working on her M.S. in Computer Science at the University of Texas at San Antonio. Laura's desire to collaborate and Chuck's interest in innovation inspired the development of the R-T-V base station. The collaboration is successful because of their desire to be way out there as well as an eagerness to make things work.

Chuck Connor, a senior research scientist at Southwest Research Institute, received his B.S. in geology from the University of Illinois in Urbana and his M.S. and Ph.D. from Dartmouth College in New Hampshire. He can be reached via e-mail at cconnor@swri.edu.

Archive Index Issue Table of Contents

Advanced search

SCEPTRE: Simulation of Nonlinear Electrical Circuits

Wolf-Rainer Novender

Issue #63, July 1999

A look at an automatic circuit analysis program through engineering-based examples.

SCEPTRE is an automatic circuit analysis program capable of determining initial condition, transient, frequency and steady-state response of electrical networks and dynamic systems. It is considered a general-purpose tool for engineers and can be used to assist in the design and analysis of various circuits. The problem of interest is described in a free-format, easy-to-learn, engineer-oriented language. The basic set of network elements consists of linear and nonlinear resistors, capacitors, inductors, voltage and current sources, and mutual inductances.

In addition, SCEPTRE has a “Defined Parameter” mode allowing it to solve state equations, which might be written to describe non-electrical systems. This outstanding feature allows the user to couple electrical networks with differential equation systems giving a great deal of versatility, especially for interdisciplinary problems like mechatronics.

If the problem of interest cannot be described by the built-in functions, the user can describe his problem as an ultima ratio via a FORTRAN program. The program uses the state variable method. Other highlights are the model library, reruns with automatic varied network quantities, extended DC options (sensitivity, worst-case, Monte Carlo, optimization), and several integration routines for accurate and fast solution of even “stiff” problems.

Example Problem

To give the uninitiated some idea of the input data required to accommodate a simple but practical problem, consider the electrical schematic of the Darlington pair in Figure 1. The SCEPTRE version of the circuit appears in Figure

2 with the assumption that a transistor equivalent circuit named 2N706A has been stored in the model library at some previous time.

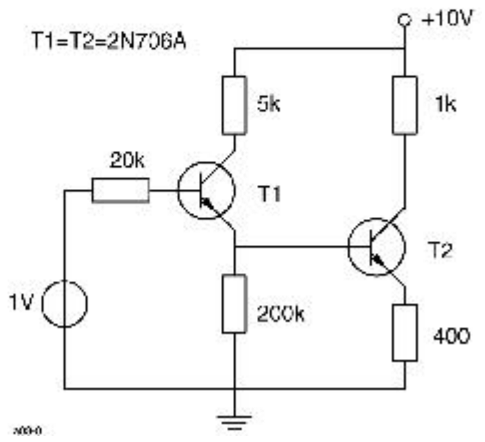


Figure 1. Schematic Diagram of Darlington Pair

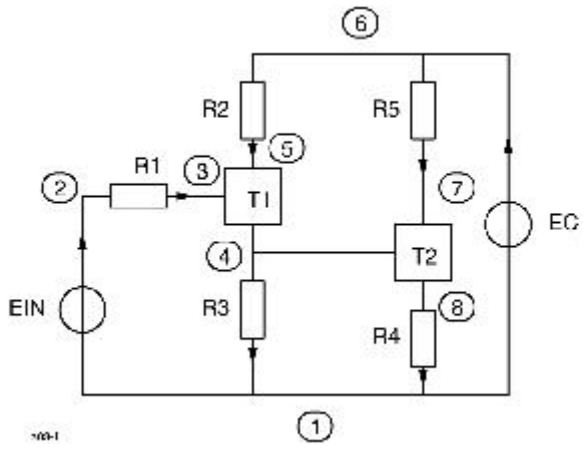


Figure 2. SCEPTRE Form of Darlington Pair

The problem is to determine the output voltages across R3 and R4 and the power requirements of this circuit under nominal conditions and after the current gain of the first-stage transistor has been degraded to various levels. Defined Parameters will be used to calculate the power consumption (PEC, PEIN). The rerun feature accommodates the additional runs required for the degraded alpha version (P1T1). The SCEPTRE input is shown in Listing 1.

Listing 1. SCEPTRE Input

SCEPTRE Use

This section describes the preparation of circuits and the basics of the circuit description language for better understanding of the examples.

Circuit Preparation

Assuming the user wants to analyse an electric circuit, the following steps should be considered:

- Draw an equivalent circuit consisting of resistors, capacitors, inductors, mutual inductances, voltage sources, current sources or models containing these elements. All of these elements may be linear or nonlinear.
- Assign to each node an arbitrary alphanumeric name.
- Give each element in the circuit a unique name, where the first character classifies the component according to Table 1.

Table 1.

- Choose a consistent set of parameter units. SCEPTRE assumes no set of units for the element values. It operates directly on the numbers supplied by the user. For numerical reasons, users should select a set of units so that the voltage and current responses expected in the solution are as close to unity as possible. A consistent set of units is one which satisfies the following equations simultaneously: $U = I R$, $t = R C = L/R$.
- Choose an arbitrary current flow in each passive circuit element, indicate the direction of positive current flow in each current source and voltage source.

SCEPTRE statements are entered under major headings and subheadings. The most important headings used are the following:

```
MODEL DESCRIPTION
CIRCUIT DESCRIPTION
  ELEMENTS
  DEFINED PARAMETERS
  OUTPUTS
  FUNCTIONS
RERUN DESCRIPTION
END
```

The actual problem is described under the major heading **CIRCUIT DESCRIPTION**. For the temporary definition or permanent storage of models, the corresponding input data go under the heading **MODEL DESCRIPTION**. The **RERUN DESCRIPTION** heading is always used whenever the rerun feature is exercised. All changes to the master network must appear under this heading.

The network topology of a circuit must be written under the subheading **ELEMENTS** in the general form:

```
element, fromnode - tonode = value
```

in case of mutual inductances:

```
  M..., Lname1 - Lname2 = value
```

value may be a number, defined parameter, reference to a table, mathematical expression using any intrinsic FORTRAN function, reference to a statement function or user-defined FORTRAN function.

Controlled sources (**E**..., **J**...) may depend on any circuit quantity (linearly dependent sources). Mainly for the analysis of semiconductor devices, current sources may depend on diode tables or built-in diode equations (primary dependent current sources) or even on other primary dependent current sources (secondary dependent current sources). In this way, the simulation is carried out in a cost-effective manner. SCEPTRE determines the correct sequence for the circuit calculation to avoid computational delays normally introduced by these nonlinearities. The controlled sources will run in an AC analysis without modification.

For the Monte Carlo, worst-case and optimization calculations, resistors and sources may be provided by bounds.

Mathematical expressions are entered in two ways. The direct way has the general form

```
  X name (mathematical_definition)
```

and is preferred when handling an equation, which is used only by a single reference. Using the mathematical sequence for several purposes, the equation form

```
  Q name (par1, par2, ... ) = ( ... )
```

is the more efficient way. It serves as a statement function and must be entered under the subheading **FUNCTIONS**.

The same subheading holds the table entries. Data are entered as point pairs with the independent variable in increasing algebraic order. To produce step functions, it is permissible to supply two consecutive independent-variable values that are equal but have different dependent-variable values.

Reference to an element voltage and element current is accomplished by prefixing the element name with the single character **V** and **I**, respectively. These references may be used as output variables under the subheading **OUTPUTS**, expressions, equations, tables and defined parameters. One of the most powerful features in SCEPTRE is the user-defined parameters under the subheading **DEFINED PARAMETERS**. The general form is

```
  P name = value
```

where value denotes the possible entries mentioned before. If the user specifies the time derivative

```
   DP name  = value
```

SCEPTRE calculates the integral:

```
   P name = ò DP name  dt
```

This allows the solution of simultaneous differential equations, which need not be coupled to any network quantity. By canonical substitution, transfer functions to any order can be evaluated. Thus, the subheading **ELEMENTS** is not mandatory. For the AC analysis, the defined parameters may be complex-valued.

A variety of run controls can be entered under the subheading **RUN CONTROLS** and used to control the simulation to a fine degree.

Simulation

The simulation is carried out in two phases. In the first phase, SCEPTRE analyses the input data and, if the input statements are free of errors, generates the appropriate FORTRAN subroutines which solve the network equations. In the second phase, these subroutines are compiled and linked with modules from the SCEPTRE library and eventually with user-defined routines. Finally, the binary to be executed contains only those modules needed by the actual problem. Moreover, the working buffers are dimensioned accordingly.

During execution, the actual simulation time (or frequency in the AC analysis) with the requested output quantities is monitored (no graphics), but may be suppressed to considerably speed up the simulation. The results are written into a direct access file.

A postprocessor called NGP is available, which reads the direct access file and enables the user to select the requested quantities. NGP then calls GNUPLOT for the final graphical representation.

Despite its apparent complexity, to get full use of SCEPTRE, only a text editor is needed along with two system commands to start the simulation and the postprocessing.

Program Organization

It was the intention of the author to keep the organization of SCEPTRE as simple as possible without losing its power. The shell scripts for SCEPTRE and NGP, which control the flow of all actions, are kept simple, allowing the user to modify them according to his requirements. NGP comes with the source code. Thus, it is possible to adapt NGP to graphic software other than GNUPLOT. Moreover, the monitoring interface routine, which is called after each time or frequency step, is also available as source and may be modified for on-line graphics.

Requirements and Availability

The version of SCEPTRE now available runs under any Linux kernel. It needs a FORTRAN compiler and GNUPLOT. The X11 Window System is not required. The binaries of SCEPTRE, including libraries, NGP (source), auxiliary routines (source), English and German documentation and a set of examples are available at ftp://novilux.fh-friedberg.de/pub/sceptre_linux. The archive also contains a suitable GNUPLOT version and a FORTRAN compiler.

History

SCEPTRE was originally developed by IBM Federal Systems Division, Owego, New York, for the Air Force Weapons Laboratory, Kirtland AFB, New Mexico, in 1966 (see Resources 1-5). The development was assumed by GTE Sylvania, Waltham, Mass., in 1972. The program could be obtained from AFWL for certain mainframes.

Examples

Typical applications for the use of SCEPTRE are circuits with semiconductor devices. Equivalent circuits for these devices are available with any desired complexity (see Resources 5-8).

To demonstrate the wide range of SCEPTRE applications, I will present two examples from different areas. Other examples are included with the listings in the file ftp://ftp.linuxjournal.com/pub/lj/listings/issue63/3008.tgz.

1. High Voltage Impulse Generator

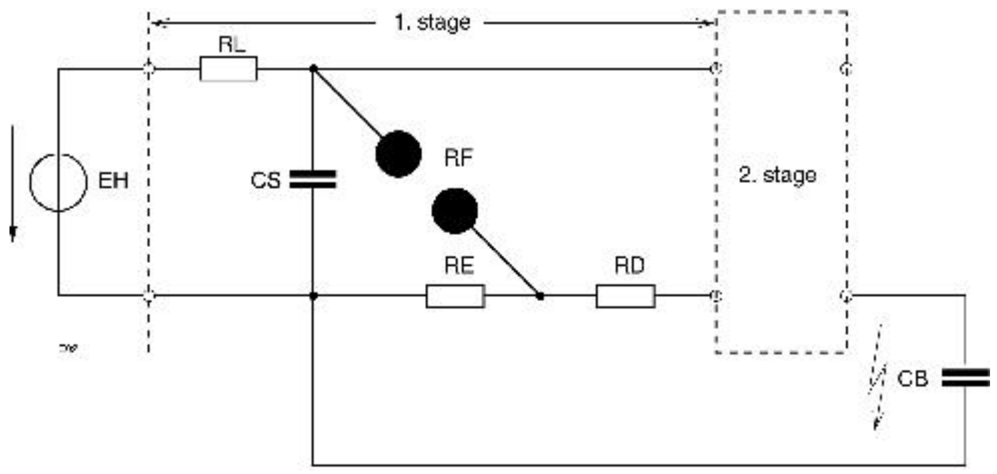


Figure 3. Schematic of a Two-stage Marx-Surge Generator

Voltage pulses of large amplitude are generated when capacitors initially charged in parallel are series-connected with the aid of spark gaps. Such a Marx-surge generator is illustrated in Figure 3. For testing high voltage equipment, e.g., a power transformer, the shape of the generated impulse must meet certain requirements (rise and breakdown time). The shape is influenced mainly by the resistors **RE** and **RD**. To save computing time, the capacitors **CS** in stages 1 and 2 are initially charged via **INITIAL CONDITIONS** to the voltage of **EH** (making **EH** superfluous).

As the stages are identical, the **MODEL DESCRIPTION** is used and each stage is included with a simple statement in the **CIRCUIT DESCRIPTION**. To avoid ambiguity of the element names in the main circuit, SCEPTRE includes the model designation as a suffix (here S1 and S2) to the component names. For simplicity, the spark is replaced by a constant resistance **RF**, but any other nonlinear function may be applied. For this example, the set of units chosen were **kV, A, k omega**, nF, mH and **mu**s. The voltage at the capacitor **CB** is referred to as **VCB** and is shown in Figure 4. The SCEPTRE input is shown in Listing 2.

Listing 2. Marx-Surge Generator Input to SCEPTRE

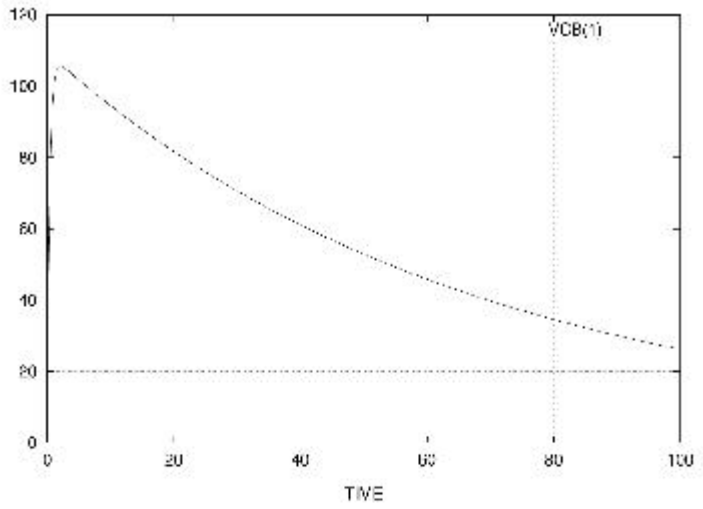


Figure 4. Output Voltage VCB

2. Rotational system

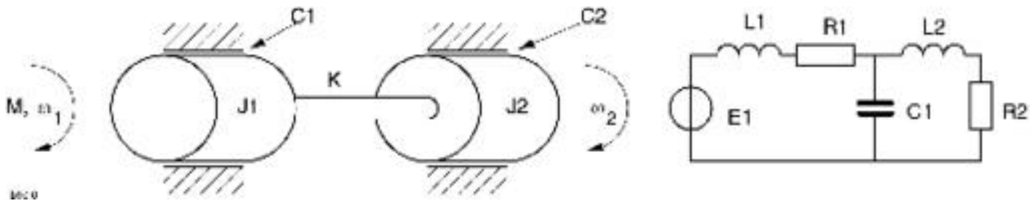


Figure 5. Rotating System with Two Degrees of Freedom, Electrical Analog of System

Figure 5 shows a rotational system with two degrees of freedom. Two bearings with polar moments of inertia **J_1** and **J_2** and viscous frictions **c_1** and **c_2** are coupled through a shaft with a spring constant **k**. A driving torque **M** is applied to the left bearing. It is desirable to find the angular velocities **omega_1** and **omega_2** of the bearings and the torque M_w on the shaft.

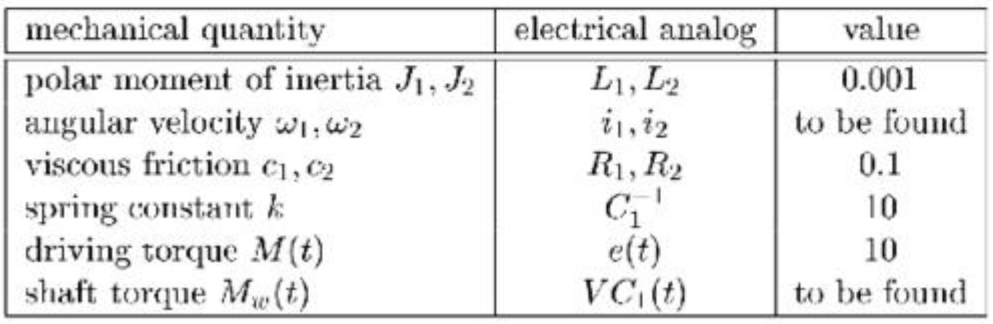
One approach is to derive directly the mechanical relationship, which yields in the following differential equations (see Resources 3):

$$\begin{aligned}\dot{x}_1 &= -\frac{c_1}{J_1}x_1 - \frac{1}{J_3}x_3 + \frac{M}{J_1}\\ \dot{x}_2 &= -\frac{c_2}{J_2}x_2 + \frac{1}{J_2}x_3\\ \dot{x}_3 &= kx_1 - kx_2\end{aligned}$$

with the substitutions

$$\begin{aligned}\omega_1 &= x_1\\ \omega_2 &= x_2\\ M_w &= x_3\end{aligned}$$

This set of equations can be entered directly under **DEFINED PARAMETERS**.



mechanical quantity	electrical analog	value
polar moment of inertia J_1, J_2	L_1, L_2	0.001
angular velocity ω_1, ω_2	i_1, i_2	to be found
viscous friction c_1, c_2	R_1, R_2	0.1
spring constant k	C_1^{-1}	10
driving torque $M(t)$	$e(t)$	10
shaft torque $M_w(t)$	$VC_1(t)$	to be found

Another method for solving this problem is converting the non-electrical system to its electrical analog as shown in Figure 5. The corresponding mechanical and electrical analogs for this problem are shown in Table 2.

<u>Listing 3. Rotating System Input to SCEPTRE</u>

The SCEPTRE input to solve these two methods simultaneously and independently is shown in Listing 3. The defined parameter **PERAB3** has been introduced to monitor the absolute error between the equivalent quantities **PX3** and **VC1**. The functions of time for these quantities are shown in Figure 6. The absolute error **PERAB3** remains less than 1.5E-15 during the complete simulation.

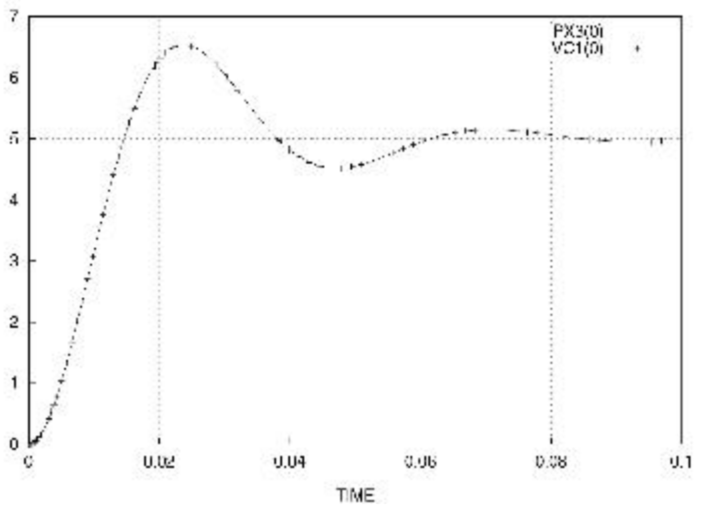


Figure 6. Output Voltage VC1 and Shaft Moment PX3

Conclusion

As can be seen from the examples, SCEPTRE is able to solve problems in a wide range of applications and is not limited to electrical engineering. It produces solutions without requiring the user to do any real programming or write equations. The Defined Parameter mode offers the greatest versatility. Avoiding any programming overhead, the problem to be solved is limited only by the ingenuity and experience of the user.

Resources

Wolf-Rainer Novender (Wolf-Rainer.Novender@e2.fh-friedberg.de) is a Professor in Electrical Engineering at the University (FH) of Giessen-Friedberg, Germany. He began working with computers in the late 1960s on an IBM 7040/1401 and has used UNIX for about ten years; Linux since 1992. He likes abandoned railroads (tracking ghost railroads), old theater pipe organs and dislikes electronic devices with too many features, multi-function buttons and unreadable and unintelligible instruction manuals.

Archive Index Issue Table of Contents

Advanced search

VMware Virtual Platform

Brian Walters

Issue #63, July 1999

According to the web site, VMware for Linux allows your PC to act as a host operating system for virtual machines which I call VMs.

- Manufacturer: VMware, Inc.
- E-mail: sales@vmware.com
- URL: http://www.VMware.com/
- Price: $299 US
- Reviewer: Brian Walters

Not too long ago, I looked across my desk and wondered, “How many PCs are too many for one consultant?” I regularly work with Windows, Linux and SCO operating systems and usually need to test interaction between different systems. I counted five PCs, including the company web server sitting on my desk. Coming up with the magic number four was fairly simple, since my keyboard switcher had enough slots for only four systems, and I didn't feel like springing for another box. I tossed the web server under the desk without a monitor or keyboard. This solution results in loading different operating systems when I'm working on different projects—certainly not very efficient. Finally, I found the solution—VMware Virtual Platform from VMware, Inc.

In mid-February, a colleague sent me an e-mail message describing a virtual machine for PCs. This sparked my interest—could it be real? With a virtual machine such as the one my friend described, the possibilities seemed endless. Virtual machines ran on old IBM mainframes; surely you would need special hardware and not just your average “run of the mill” PC sold today. A few clicks of the mouse took me to the VMware web site, where mention was made of an upcoming beta program. I signed up and waited patiently for the beta release date of March 15.

Figure 1. VMware Bios Screen

According to the web site, VMware for Linux allows your PC to act as a host operating system for virtual machines which I call VMs. The virtual machine presents a complete image of a standard PC to the guest operating system. The VM also has its own BIOS (see Figure 1) that can be adjusted just like your real computer. Many components are virtualized, such as the network card, sound card, hard drive and mouse. However, the processor is not; it is more like a serial multiplexer where instructions are multiplexed from the different VMs and host OS. This gives the user much better performance, since the instructions are executed without any translation.

Unfortunately, this fantastic performance does come at a cost—you can't step through a program and you can't test SMP (symmetrical multi-processing) applications on your VMs. The stepping problem should be fixed by the time this article is published, but the SMP issue will take longer. SMP systems are supported in the host OS, so you can actually have multiple VMs running on different processors. Virtualization of the hard drive also adds a wonderful feature for developers—rollbacks. A VM hard drive can be configured read-only to allow the guest OS to make changes in a log file. If something goes wrong, just restart the VM and everything is back to its original state. Imagine how easy this makes testing install scripts.

Finally, the day arrived when I was able to download the beta to test whether it was all it was cracked up to be. On the morning of the beta release, the web site received so many hits the company was forced to quickly put up mirror sites. I was pleased with the amount of documentation available, including ample information on the errors you might get and how to fix them.

I rebooted my primary machine from Windows to Linux and followed the installation instructions. VMware installed easily and seemed a perfect fit with Red Hat 5.2 and the 2.0.36 kernel. I then installed the license file received via e-mail. I started VMware and went through the configuration wizard, which made it easy to set up a ready-to-use configuration for Windows 98 (see Figure 2). Next, I popped in my Windows OEM preboot floppy and clicked the “power” button. In the window, a normal BIOS screen appeared with the VMware logo on the right. Fifteen minutes later, I was done. Just like a regular PC installation, the new virtual disk was partitioned and formatted following a virtual reboot and installation of the OS. There was absolutely no difference between my test installation and a normal Windows 9*x* OEM system load.

Figure 2. Windows 98 Startup

Next, I loaded Caldera, SuSE, Windows 95 and even Windows NT Terminal Server. All of these operating systems loaded without any problem, and I used the wonderful documentation to get the virtual sound and network cards working. Finally, I moved my installations over to a better-equipped PC, a dual Celeron 300a clocked at 450MHz with 256MB of RAM and a 13GB hard drive. Since the VMs appear the same on all hosts, I was able to copy my previous installs over to the bigger PC. With the addition of power and memory, I was able to run most of these operating systems simultaneously with ample performance in each (which was good, since Windows NT Terminal Server is a resource hog). Probably the best example of performance was when I started compiling the 2.2.5 Linux kernel on my host PC, started another kernel compile on a VM running SuSE, and viewed a RealVideo clip playing in Windows 95 on a VM. The sound and video never skipped a beat.

Figure 3. Windows 98 Running RealPlayer, SuSE 6.0 Running Star Office and Red Hat 5.2 Running the GIMP

Now those four PCs on my desk have become many more. Support for FreeBSD, Solaris and Windows 2000 Beta are currently underway and should be available by the time you read this. However, some operating systems are not on the immediate horizon: SCO OpenServer, UNIXware and NetBSD are not planned as supported platforms for the official 1.0 release. VMware allows me to test many versions of Windows and Linux workstations all at the same time. I am hoping to learn more about FreeBSD and Solaris once they are supported. Thanks to the virtualization of the network card, I can assign each VM its own IP address and actually test how multiple workstations will interact with each other from one PC. I have found it is even possible to have a VM respond to a DHCP server when your host OS does not have an address on the LAN.

How much does an incredible application like this cost? Considering the cost of a decent PC today is at least $1000 or more, I think $999 would be a fantastic price, since it will save you so much space. The folks at VMware think differently though—for just $299 US, you can give your computer a multiple PC disorder. A student discount of 67%, i.e., to $99, is available. While I wouldn't recommend this to all the gamers, if you're developing applications for Linux or Windows this is truly a tool you should not be without. VMware is also great for testing the new distributions of Linux with their many new features.

Brian Walters (Brian@TexasComputers.com) has worked with computers since the early 80's. In 1996, he formed R & B Consulting to specialize in providing unique solutions for small and medium businesses using both Linux and Windows. He enjoys hunting in his Jeep, but doesn't like to get too far from society, as he cannot live without the Internet.

Advanced search

Open Sources: Voices from the Open Source Revolution

Doc Searls

Issue #63, July 1999

Linux is the latest and greatest topic in a UNIX conversation which has been going on for nearly thirty years.

- Editors: Chris Dibona, Sam Ockman and Mark Stone
- Publisher: O'Reilly & Associates
- E-Mail: info@ora.com
- URL: http://www.ora.com/
- Price: $24.95 US
- ISBN: 1-56592-582-3
- Reviewer: Doc Searls

First, a bit of a digression that will ultimately lead to a discussion of the book.

> Don't look back. Something might be gaining on you.
> —Satchel Page

Comdex legitimately bills itself as “technology's main event”. This spring, it grew a second name: Windows World. So naturally, the first keynote address at Comdex Spring/Windows World in Chicago was by Bill Gates. Thousands packed the auditorium where a year earlier a Gates demo had famously crashed. Hundreds more herded into meeting rooms to see the leader of the computing world in a simulcast ironically projected by Macintoshes.

It seemed to go well. Windows and Office 2000 both looked terrific. Nothing went “boom”. Everything appeared to be on track for a pre-millennium release. When the speech ended, crowds dispersed onto the show floor which looked like a giant single-celled Microsoft animal, with the enormous Microsoft company pavilion as its nucleus. Clearly, the rest of the industry, which used to live in Microsoft's orbit, now must live within its cell wall, like mitochondria or viruses.

That's why the Linux pavilion looked like an infection. It was something odd, alien, alive, growing and *already inside* the cell wall. Huddled off in a corner amidst the connector-cable and small-bore peripheral vendors, the pavilion had a constant crowd that buzzed like a hive.

The entire population of that hive swarmed to the second keynote, given by Linus Torvalds. Perhaps hoping to ghettoize this group as far from the main floor as possible, the Comdex people scheduled the talk for a small basement-level meeting room. When the whole hive showed up, they hastily moved the venue upstairs to a room that seated 850 people. By the time the swarm settled, more than 1200 were spilling into the halls.

Of course, the press reported only Linus' *pro forma* digs at Microsoft. But the true story was more in the crowd than in the speech, which most attendees had heard before. “Linux started as this one-man project that exploded with the help of the Internet”, Linus began. You know the rest, but you might not know the whole context.

Linux is the latest and greatest topic in a UNIX conversation which has been going on for nearly thirty years. Linux also owes much to Berkeley UNIX and the unrealized dreams of the Open ethos, which at one point every UNIX vendor claimed but none practiced. A case can be made that Linux is the fastest-growing branch of the free software movement, which has been driven since the seventies by the highly principled zealotry of Richard Stallman.

At Linux Expo early this year, Eric Raymond, the Johnny Hackerseed of the open-source phenomenon, said “none of us would be here today if it weren't for Richard Stallman.” But Stallman doesn't like the Open Source label. “The rhetoric of Open Source focuses on the potential to make high-quality, powerful software, but shuns the ideas of freedom, community and principle,” he says; in the next sentence, he offers this magazine as an example. Our pages, he says, “are filled with advertisements for proprietary software that works with GNU/Linux. When the next Motif or Qt appears, will these magazines warn programmers to stay away from it or will they run ads for it?”

The answer is both. Contradiction and controversy are the stuff of stories, and stories sell magazines and books.

Now to the Book

Open Sources: Voices from the Open Source Revolution tells the Open Source story in the words of Stallman, Raymond and more than a dozen other very interesting protagonists. Published by O'Reilly and edited by Chris DiBona, Sam Ockman and Mark Stone, *Open Sources* is the perfect place to start understanding the deepest and most important change in the history of the software business. For all the disagreements about the particulars of Open Source, what holds the movement together is devotion to an ideal that both Linus Torvalds and Eric Raymond call “software that doesn't suck”. At a trivial level, this means “software that isn't Microsoft”, but at a deeper level it means a fundamental shift in the whole software-building trade.

This isn't an obvious matter. Eric Raymond is an unusually insightful observer, yet his understanding of open source and its virtues did not come quickly and is still highly speculative. “Encountering Linux was a shock. Even though I had been active in the hacker culture for many years, I still carried in my head the unexamined assumption that hacker amateurs, gifted though they might be, could not possibly muster the resources or skill necessary to produce a usable multitasking operating system,” he writes. “It took me three years of participation and another year to test it experimentally. Then I sat down and wrote *The Cathedral and the Bazaar (CatB)* to explain what I had seen.” Which was “a community which had evolved the most effective software-development method ever and did *not* even know it. That is, an effective practice had evolved as a set of customs, transmitted by imitation and example, without the theory or language to explain why the practice worked.”

CatB started a firestorm of talk about open-source development. Among other things, it helped convince Jim Barksdale and the crew at Netscape to release the source code to their browser. That story is also told in *Open Sources* by the Netscape folks.

What all these voices give us is not final truth, but code to hack. We have principles to discover here, not just to apply. “The point is that open-source software doesn't need to beat Microsoft at its own game,” Tim O'Reilly writes. “Instead, it is changing the nature of the game.”

This game is being played mostly at the server end of the client/server relationship. As Apache developer Brian Behlendorf puts it, “Open-source software has tended to be slanted toward the infrastructural/back-end side of the software spectrum.” After giving four good reasons for this, he adds, “This is why we see solid open-source offerings in the operating system and network services space, but very few offerings in the desktop applications space.”

Yet Robert Young of Red Hat describes his strategy as one intended to build a *consumer* market for Linux, which he pursues by purposefully advancing the Red Hat “brand” more than the virtues of its products. “The Linux OS gives consumers choice over the technology that comes with their computers at the operating system level,” he writes. In spite of the fact that the consumer market for Linux is approximately zero, Young asks, “Will (the consumer) go back to the old model of being forced to trust his binary-only OS supplier once he has experienced the choice and freedom of the new model? Not likely.”

I think this is delusional. The “choice and freedom” of this new model still floats all kinds of unfinished software that's catnip for hackers but poison to ordinary users. I lost an early draft of this review by attempting to write it in the “advanced” text editor that comes with KDE, before reading its extensive bug list. I'm finishing it off on trusty software from one of those awful binary-only suppliers.

Bob Young's branding-for-consumers strategy is plainly a brilliant one that has made Red Hat first among Linux distribution equals. It has also attracted investments by IBM, Intel, Compaq, Novell and a growing raft of old-school big boys. They want to be in on whatever changes the game Microsoft has been winning for the last twenty years.

If you want to understand this new software game, *Open Sources* gives you the best coaching you can get from the players who are already winning. Even in stadiums like Comdex/Windows World.

Doc Searls (info@linuxjournal.com) is a Senior Editor at *Linux Journal*. He also gives us marketing advice we take quite seriously. He telecomputes from Northern California—nice job if you can get it. While we have convinced him Linux is the wave of the future, we have not convinced him to learn vi.

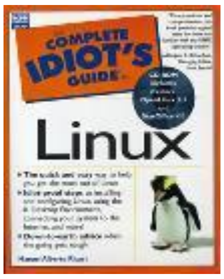
Advanced search

The Complete Idiot's Guide to Linux

Marjorie Richardson

Issue #63, July 1999

The book's format makes for easy reading, and needed information is easy to locate.



- Author: Manuel Alberto Ricart
- Publisher: Que, Macmillan Computer Publishing
- URL: http://www.mcp.com/
- Price: $19.99 US
- ISBN: 0-7897-1826-X
- Reviewer: Marjorie Richardson

The Complete Idiot's Guide to Linux is yet another book for the newcomer to Linux. The layout of the book is just like others in the *Idiot's* series, so if you have read one of the others, this one will seem familiar. I've personally never understood why a company would choose to insult their customers by calling them “idiots” or “dummies”, but it seems to work—this is a popular series. I also found the graphics a bit cutesy, but then again, whatever works. The book's format makes for easy reading, and needed information is easy to locate.

This book takes a unique approach by covering the use of Linux with a graphical desktop, namely KDE. This should make Linux seem a bit more palatable to users migrating from Windows or Macintosh environments. It is advertised as being for the beginning to intermediate user—I'd say just the beginning user. The information is good, but so basic the intermediate user will already know it. An example of one such simplistic bit of information is the statement “icons are buttons too”.

The distribution covered is Caldera OpenLinux v1.3, which is on the CD-ROM included with the book. Installation is relegated to Appendix A, so you read the Appendices first, then move on to Part 1, Chapter 1, which describes the KDE desktop. Installation, in fact everything in the book, is given a step-by-step approach that should be easy even for the complete computer beginner. Information is comprehensive and presented in a clear, concise manner.

The author points out that while KDE looks much like MS Windows and Macintosh desktops, quite a few differences exist. He then goes on to describe KDE, letting you discover the differences on your own. Using KDE certainly makes Linux more accessible to the GUI aficionado. KDE looks good and works well. It provides drop-down menus, buttons, tool bars, dialog boxes—all the things one expects of a good graphical user interface. Reading through Part 1, I kept asking myself, “where's the prompt?” I didn't find out until Chapter 11.

Basic file organization and management through the GUI is fully discussed. Changing file permissions and groups can be done by clicking on a button—no more time spent figuring out the right value to give the **chmod** command. Instructions are also given for reading and writing files to floppy disk.

Chapter 8 talks about accessing the network. Again, it is done with step-by-step instructions for everything from configuring the modem to setting up dial-up service to your ISP. All forms of communication, e.g., web, e-mail and FTP, are discussed at their most basic levels. After reading these chapters, you will know what these services are and how to use them—exactly what you, as a newbie, need to know.

In Part 2, I finally found my introduction to the shell. Here, the most basic commands, such as **cd** and **ls**, are discussed, along with those options considered useful by the author. When discussing file commands, the proper warning about running commands as root is given. The text editors covered are **vim** and **Xemacs**.

Part 3 deals with necessary system tasks, including user management and backups. Since the covered system is Caldera, Chapter 21 deals with LISA, the Linux Installation and System Administration utility. There are also chapters on installing applications using the Red Hat Package Manager (jointly developed with Caldera), customizing your kernel using LISA, and configuring Apache. By the time you've learned how to do all the tasks described in these final chapters, you can upgrade your standing from newbie to intermediate.

Mr. Ricart has written a very good book for the newcomer to Linux, giving the reader the basics without going deeply into technical details. He has been working with UNIX for many years and with Linux since 1996. He knows his stuff.

Marjorie Richardson is the Editor of both *Linux Journal* and *Linux Gazette*--she has no free time. If she did, she'd buy a motor home and tour the country visiting relatives and friends wherever they might live.

A Geek In Paradise

Jon “maddog” Hall

Issue #63, July 1999

A trip to see the particle accelerator at Fermilabs by a self-professed geek.

I had been to Fermilab only the year before, but when the invitation came from Dan Yocum to meet at Fermilab's facility outside Chicago, how could I refuse? I am a geek at heart.

Figure 1. Fermi Campus

Fermilab is short for “Fermi National Accelerator Laboratory”, located in Batavia, Illinois. It occupies a parcel of land about three miles on each side (see Figure 1), and houses several accelerator rings which generate (in a *very* concentrated space) amounts of power greater than those found in the sun or any other place in the galaxy, much less on the face of the earth. They use these fantastic amounts of power to collide various particles at extremely high speed in the search for the basic building blocks of the universe.

Figure 2. Dr. G.P. Yeh (third from the left) and Linux supporters: Ruediger Oertel from SuSE, Fermilab System Administrator, G.P. Yeh, Stefan Traby from Quant-X, Larry Augustin from VA Linux Systems, Norman Jacobowitz, Linus Torvalds, Dan Yocum, maddog and Matthew Cunningham

In ancient days, various philosophers stated that we would eventually find the "smallest particle", and for a while this was considered to be the atom. In the relatively recent days of discovering nuclear energy, it was recognized that the smallest particle was *not* the atom, but made up of various other parts such as protons, neutrons and electrons. (Students of physics, please have mercy on me as I try to explain this in words that most readers will understand.) During the last quarter of a century, more and more physicists began to believe there were even smaller particles making up the protons, called quarks and gluons. Quarks (having nothing to do with a resident of *Deep Space Nine*) are thought to have six different types, and in 1994 the last of these Quarks, the "top quark", was discovered at Fermilab. Unfortunately, the top quark exists for only a very short (10 -24 seconds) period of time, so it is very hard to collect data on it, particularly when it is seen only six times in a given year of running the accelerator. Therefore, Fermilab decided to increase the size and power of its accelerator, so it could see anywhere from 20 to 300 times the number of quarks. Unfortunately, this would take anywhere from 20 to 300 times the amount of power and generate 20 to 300 times the amount of raw data to be seen by the collectors, meaning 1,000,000MB of data would be generated every second. Yes, that is one million megabytes of data per second.

Of course, storing that much data would be very difficult, but fortunately Fermilab had determined they would be able to filter the information and store a smaller subset of it (only 18 to 100MB of data per second) for later analysis. To do this, they would have to increase the power of their computing systems significantly, and their former model of using expensive workstations in a workstation farm would not have been affordable. Enter Linux.

Figure 3. Fermi main building.

Last year, when people from Red Hat Software and I visited Fermilab while attending Spring Comdex, I was lucky enough to meet G. P. Yeh, a big fan of Linux and one of the physicists who discovered the top quark. He was kind enough to take us on a short tour of the Fermilab facilities and explain the role of Linux within Fermilab. He explained they investigated Linux and proved that inexpensive PCs running Linux could do the job more than adequately for a price they could afford. They estimated they would need about 2,000 CPUs working together.

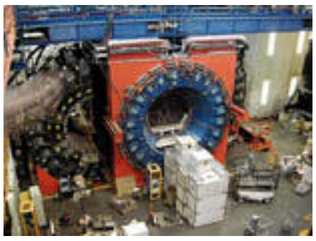


Figure 4. Collider Rings

This year, when Dan Yocum heard that Linus Torvalds was speaking at Spring Comdex, he enlisted my help in convincing Linus to make a separate trip to Fermilab to speak to the physicists and their families. This did not take much convincing, since Linus has an interest in math, physics and science.

Figure 5. Computer Room-stacks of Linux boxes

We met at the hotel where Linus was staying, and with a small group of Linux supporters (see Figure 2), drove to Fermilab. It is quite interesting to approach Fermilab, since the land around the accelerator is flat, with only the main building (see Figure 3) rising up from the ground to any height. It would definitely be a great scene for a science fiction movie. We parked the car, went inside and met Dr. G. P. Yeh (who everyone calls “G.P.”).

G.P. took us on an extended tour, beginning with the top floor of the main building, looking out over the collider rings. “As far as you can see in every direction is Fermilab”, G.P. said. It was an impressive sight. He then took us to see the collider detectors (see Figure 4)—“It weighs only 100 tons and cost about 100 million dollars.” Finally, we visited the computer room, where the Linux Farms were going to be placed (see Figures 5 and 6). Fermilab calls their systems “Farms” rather than Beowulf systems. They have master machines that delegate the work to many slave processors, connected by high-speed networking and switches. They are not planning on buying the 2000 CPUs until very close to the time they need them. After all, prices keep dropping and capabilities keep increasing, so why not wait until the last moment to get the best “bang for the buck”?

Figure 6. Linux Farms: Larry Augustin, VA Linux Systems, Dan Yocum, Fermilab

After the tour was over, we went to the main auditorium where Linus gave his talk. For those of you who have heard Linus give a speech, you know he does not like to talk with prepared slides, but instead gives a short prepared talk, then answers questions. This night was no different, other than the topic and complexity of the questions. It was obvious from the questions asked that the audience had more of a computer science bent than other, more general audiences. Questions regarding symmetric multi-processing and the reality of distributing interrupts over multiple CPUs entered the air.

After a significant amount of time answering questions and signing autographs, our little troupe went to the home of Jeff Gerhardt to enjoy pizza and “refreshments”. We were greeted by smoke rolling out of the front door, reminding everyone it is best to take the pizza out of the box before warming it in the oven. When the smoke died down, some interesting home brew made its way to the front, and everyone enjoyed the pizza and brew (see Figures 7 and 8).

Figure 7. Party Time: Linus on left by lamp, G.P. Yeh in far chair, Stefan Traby in far right

Figure 8. Jeff's Kitchen: Jeff Gerhardt's hospitality (and kitchen) were enjoyed by all.

Figure 9. G.P. Yeh shows map to Linus Torvalds and Stefan Traby

I love this type of computing where people push the envelope of what the human mind can conceive, and I thank the government of the United States for helping to fund such a quest.

Jon “maddog” Hall is Senior Leader of Digital UNIX Base Product Marketing, Digital Equipment Corporation. He is Executive Director of Linux International.

MP3 Linux Players

Craig Knudsen

Issue #63, July 1999

Is MP3 the wave of the future? Mr. Knudsen describes this new technology and what it will mean to the listener.

MP3 (Motion Picture Experts Group audio layer 3) is an audio file format capable of storing near CD-quality audio in relatively small files (as little as 1/10 the size of comparable WAV files). A four-minute audio track recorded at 44KHz requires about 4MB in MP3 format, while the same audio track recorded in WAV format can occupy over 40MB. MP3 has received a great deal of attention lately. According to searchterms.com, a site that ranks the most frequently used search terms, “mp3” is number one. That's right—it's even ahead of “sex”, which is number two.

MP3 as a Threat

The Recording Industry Association of America (RIAA) is concerned that MP3 will make it difficult to sell music on-line, as MP3 has no mechanism for preventing illegal copying. In December, RIAA announced its Secure Digital Music Initiative (SDMI) to encourage technology companies to create a new specification for on-line music. RIAA also sued Diamond Multimedia in October, attempting to prevent the sale of Diamond's RIO PMP300 portable MP3 player. Fortunately for Diamond and MP3 fans, RIAA lost the suit.

In February, Lycos and FAST launched a search site for MP3 files on the Internet. The site contains a database of over half a million links to MP3 files. While the site is a great resource for finding MP3 files, it immediately caused concern about illegal MP3 files. Search for one of your favorite artists, and you may find links to illegal MP3 files. In March, the International Federation of the Phonographic Industry (IFPI) began legal action against FAST for its role in the MP3 search site. IFPI members include BMG, EMI, Sony Music, Universal Music International and Warner Music. In a statement on their home page, IFPI Director of Operations Mike Edwards said:

> The Lycos/FAST search engine should be promoting opportunities for the many start-up businesses that are pioneering legitimate electronic delivery of music over the Internet. Instead, however, this search engine is doing the opposite: it is providing a service where virtually no authorized files can be found. This is a threat to the companies who want to build a flourishing legal electronic marketplace.

IBM has introduced its Madison project to allow cable modem users to purchase music on-line, download it and save it onto CD-ROM (provided they have a CD-R or CD-RW drive). The CD can then be played on any CD player. IBM claims the technology will prevent users from making illegal copies. The project has the backing of IFPI.

Sony has also proposed another way to distribute copyright-protected music. This system includes MusicGate for recording and playback and OpenMG for use on computers. OpenMG would require hardware installed on the computer's serial port to prevent illegal copying.

MP3 Meets Linux

Quite a few MP3 players are already available for Linux. The console-based **mpg123** is popular. You can use it from either the command line or one of the many applications that use mpg123 as the MP3 decoder, and have a nice GUI for song playlist management. GQmpeg is an excellent MP3 player with a customizable GUI using “skins”. You can also try one of the many MP3-capable GUI-based applications that don't use mpg123 as the MP3 decoding engine, such as FreeAmp or X11Amp. Some of the MP3 players also support streaming MP3 (called Shoutcast), similar to Real Audio.

MP3 players are not particularly useful without some MP3 files. MP3.com has a wide variety available for download. Many users are creating their own MP3 files from their personal CDs. This is legal for personal use only. You cannot redistribute audio tracks you copy from one of your CDs. The process of copying the audio data from a CD is typically called “ripping”; applications that do the work are “CD rippers”. A good tool for this on Linux is **cdparanoia**, which will read from both SCSI and ATAPI CD-ROMs. The CD rippers typically save data in WAV format, which must then be converted to MP3 with another tool such as **mp3encode** or **bladeenc**. An increasing number of console and GUI tools are available to automate this process by looking up the track names on the Internet with CDDB and then using external rippers and encoders to do the work. You put the CD in your CD-ROM, select the tracks to encode and wait.

Some issues must be considered before converting all your CDs to MP3. The process can be quite time-consuming for both CD ripping and encoding. The reading of an audio CD is handled much differently than a data CD, and the process can take hours. This is especially true if the CD has scratches on it. Encoding the WAV files into MP3 can also take hours. This process is best done when you're not otherwise using your machine. In fact, many Windows-based encoders have options to power down your machine when finished. Once your MP3 files are ready, there is still one last issue to consider. Playing MP3 files can use up a good deal of your system's CPU cycles. It is typical for MP3 players to use 20% or more of your CPU cycles on a 200MHz Pentium. That's not a big deal if you're browsing the Web or using a word processor, but it certainly is if you're recompiling the kernel.

MP3 on the Move

Diamond Multimedia was early to market with the Rio portable MP3 player. Weighing in at only 2.4 ounces and selling for around $200, Rio can hold 40 minutes of music and play for 12 hours on a single AA battery. Because the Rio has no moving parts, the player won't skip like a CD player when subjected to movement. The music is stored in 32MB of flash memory, and you can purchase an additional 16 or 32MB of flash memory. MP3 files are transferred from your computer via the parallel port at a rate of ten seconds per megabyte. The software provided with Rio is for use on Windows 95/98, but a few different Linux applications are under development which can be used to manage Rio's files (see Resources).

Linux and MP3 in Your Car: Empeg

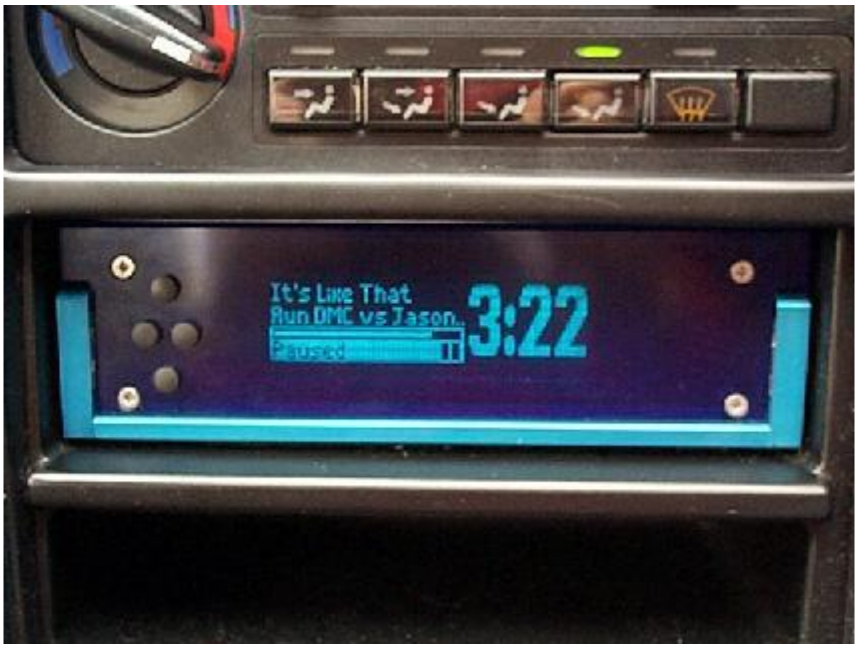


Figure 1. Installed Empeg Player

Empeg is at the cutting edge of MP3 technology with their car audio player. The unit uses a 220MHz Digital/Intel StrongARM processor with 8MB of memory and runs Linux 2.2. The player can handle all types of MP3 files and includes an

FM radio (but no AM due to interference problems). Additionally, you can connect any standard CD player, tape deck or radio to the player as long as it has line-level outputs.

Like Diamond's Rio player, you manage all the files on your Empeg player from your computer. The Empeg player slides out of the docking bay in your car so you can connect it to your computer (or to your home stereo via RCA jacks). The standard interface will be a Windows 98 application that helps manage your audio tracks via the USB port. For those without USB (or Windows 98), a version will be available for Windows 95 and NT that uses RS-232 rather than USB. Linux tools will also be included and available as source code. You can expect a group of enthusiasts to help further develop and improve the Linux tools. The user interface is written in Python and should allow developers to change the UI.

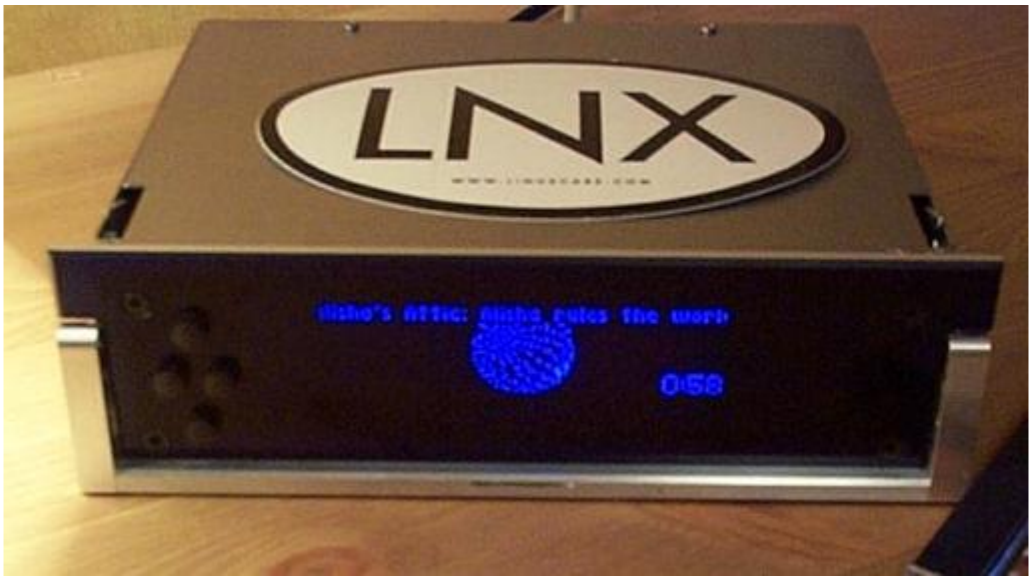


Figure 2. Empeg Player

The Empeg player is not something you can build at home. “It's a 100% custom hardware and software effort: there aren't any off-the-shelf parts in there (there simply isn't room),” said Empeg's Hugo Fiennes. Asked why Linux was chosen, Hugo replied,

> We needed a powerful, flexible OS to support our applications: the Empeg does much more than just play tunes—it has an integrated database, and uses glibc threads and IPC (interprocess communication) quite heavily. As the Empeg is hugely overpowered for its current task, we wanted an OS that would allow hackers to add their own code to the system.

Pricing starts at $999US with a 2.1GB disk capacity that can store approximately 37 hours of music. A 28GB version will also be available. The unit should be shipping any time now, and will initially be available only directly from Empeg (http://www.empeg.com/). Distribution sales will be considered after satisfying their backlog of 6000 interested parties.

Down the Road

Much of the interest in MP3 is not a result of the specific file format, but in what it allows one to do. Why else would people be excited about something that produces lower-quality sound than CDs 15 years after CDs were introduced? MP3 returns some of the power back to the music enthusiasts, who can now listen to a custom selection of their favorite singles. With the help of a computer, MP3 allows people to create their own personal commercial-free radio station.

It can also be said that MP3 is forcing the recording industry into the on-line music business. The industry giants seem content with their current business model. With full-length CDs accounting for 74.8% of music sales according to RIAA's 1998 Consumer Profile, they're making a lot of money. The future of on-line music sales is not in selling CDs from an on-line music store. It will involve selling digital audio and delivering it over the Internet. Users will be able to buy only the music they want, not forced to pay $10 or $15 to get the one or two songs they like. Even if something other than MP3 takes us there, it will still be a good thing.

Hugo Fiennes Interview

Resources

Craig Knudsen (cknudsen@radix.net) lives in Fairfax, VA and telecommutes full-time as a web engineer for ePresence, Inc. of Red Bank, NJ. Craig has been using Linux for both work and play for three years. When he's not working, he and his wife Kim relax with their two Yorkies, Buster and Baloo.

Archive Index Issue Table of Contents

Advanced search

Linux on IBM Thinkpad 750Cs

Daniel Graves

Issue #63, July 1999

One man's experience with installing, configuring and running Linux on a laptop computer.

Linux is known to be a stable operating system that does an excellent job of managing the PC's hardware. With that comes the challenge of getting Linux to work properly on some hardware. One particular example is getting Linux to work on laptops. Some brands can be tough to configure, while others are a piece of cake. A particular brand of laptop that works great with Linux is the IBM Thinkpad series. It can be hard to get working at first, especially the X Window System. I am a happy user of Linux on my IBM Thinkpad 750Cs, and this article is a description of what I did to get Linux functioning perfectly.

The Hardware

The IBM Thinkpad 750Cs has an Intel 80486 DX processor. The 750Cs has approximately 20MB of memory and 330MB of hard drive space. The hard drive and floppy drive are both made specifically for IBM. The floppy drive is a 2.88MB drive which shows up in a few Thinkpads, and the video card is a VGA card. These hardware specs are important to know for someone who will be configuring Linux.

Installation

I am using Red Hat Linux 5.1 with kernel version 2.0.35. Installing this version of Linux went smoothly. I downloaded the boot and root disk installation images and put them on two floppies. I used a Backpack, 4x speed external CD-ROM drive from Microsolutions Inc. for the installation. The installation program is capable of finding this drive, so that made the rest of the installation run without problems. My only difficulty was with X.

The Thinkpad Floppy Problem

The Thinkpad floppy drive has an inverted disk change sensor that Linux doesn't automatically support. Thus, in order to fix this, I had to pass **floppy=thinkpad** to the kernel at the LILO boot prompt. This must be done for the installation to complete properly.

X Window System

Getting X to work on my 750Cs was the toughest part. The problem lies in the 75xCs series of Thinkpads and their dual-scan monitors. X starts, but displays only a black screen with an occasional vertical line. The only way around this problem is to use a program, written by Michael Steiner, that disables the upper 512K of video memory. This program can be downloaded from http://www.zurich.ibm.com/~sti/tplinux.html. After downloading it, I ran **xf86config** and chose the smallest option for the monitor and a standard VGA card. Since the video card is a VGA card, the only server available is XF86_VGA16. Then Michael Steiner's program must be enabled once before starting X. Use the following command: **tpdualscan -e**. Note that when the problem first occurred, pressing **ctrl-alt-delete** wouldn't shut down the computer under that black screen. First, I had to end X by pressing **ctrl-alt-backspace**, then **ctrl-alt-delete** to shut down my computer, eliminating the black screen.

PCMCIA

Linux supports the Thinkpad's PCMCIA slots perfectly, and since my individual cards were supported, I had no problem here. I personally recommend 3Com's Etherlink III 3c589D card for Ethernet networking, because **cardmgr** found this card easily and it works fine. I didn't have to edit any of the PCMCIA configuration files.

Conclusion

The two main problems with the Thinkpad are the floppy drive and the XFree86 problem. Linux supports the rest of the computer perfectly. Problems with other Thinkpads shouldn't be too hard to fix, and Michael Steiner's web page is a good place to start for help.

Daniel Graves is a freshman software engineer at the Milwaukee School of Engineering. He enjoys programming in C/C++ and learning about game

development and the game industry. He can be reached through e-mail at gravesd@msoe.edu.

CORBA Program Development, Part 3

J. Mark Shacklette

Jeff Illian

Issue #63, July 1999

A look at CORBA implementations in Java to provide interoperability between platforms.

Over the past couple of months we have endeavored to present an overview of distributed application development on Linux using CORBA. In the first article we dealt with the question “what is CORBA?” and covered the basics of using an ORB with a simple client and server. The second article introduced two of the most common OMG (Object Management Group)-supported services, the Naming Service and the Event Service, and provided an example using both. In this, our third and last article, we will be digging a little deeper into the approaches used so far by providing an introduction to “tie”, which is a delegatory method of binding to a remote object. To date, all of our examples have been implemented in C++, but we must remember that CORBA is designed around the concept of both platform and language independence. In order to further demonstrate the concepts of platform and language independence, this time we will be presenting an example using two different operating system platforms, one of which is Linux, and our implementation will be in Java instead of C++.

While we used a combination of Linux and Windows when writing this article, our code should run on any combination of Java-enabled platforms. That is to say, you should expect this code to run equally well in a Linux+Solaris environment, or an HPUX+IRIX environment. For part 3, we have chosen to implement our example using one of the most popular Java-based ORBs, VisiBroker from Inprise Corporation. Inprise kindly and officially sanctioned all the research we conducted.

As we discussed in the second part, the OMG specification describes how an ORB bootstraps itself in order to find the location of the Common Object

Services (COS) such as the Naming Service, Trader Service or Implementation Repository. This bootstrapping process is hidden within each vendor's implementation of the **resolve_initial_references** method. The real trick in achieving interoperability between ORBs is to figure out how to bootstrap against another ORB. VisiBroker provides a simple proprietary naming service in the form of a binary executable called the **osagent**. The osagent provides this basic location service while at the same time supplying a measure of fault tolerance and load balancing of objects. In the case of VisiBroker, the implementation of resolve_initial_references has the ability to locate an osagent and then ask it for direct references to other COS services. Additionally, both VisiBroker's Naming Service and Event Service will try to find an osagent when started in order to register their IORs (interoperable object references). In our example, the osagent will supply our sample clients and servers with the references to VisiBroker's Naming and Event Services and any dependent objects.

Unfortunately, the osagent is a platform-specific binary application that has not yet been ported to Linux. The good news is only one osagent needs to be running on a local network for a single or group of VisiBroker-based applications to use. For our examples, we'll be running osagent on an Inprise-supported operating system. Quoting directly from VisiBroker's 3.4 release notes, “With the exception of the osagent, **osfind** and **locserv** executables, the VisiBroker for Java ORB is written entirely in Java and can run in any Java 1.1- or Java 2-compatible environment.” So we will run osagent on a supported platform and do all our other work on Linux. Since a simple ORB is also bundled with Java 2, be sure to read the release notes before trying to run VisiBroker using Java 2. The significant news is, according to James Ferguson, Senior Product Manager for VisiBroker, “Inprise has seen growing corporate demand for VisiBroker on Linux. This is just another indication of the rapid growth of Linux in corporations. To participate in this new growth market, Inprise will be making significant announcements regarding the availability of VisiBroker for Java and C++ on Linux.” If you'd like to register your support for that direction or find out more information, Inprise suggests you direct feedback to news://forums.inprise.com/inprise.public.visibroker.

The sample code in this article uses a popular OMG CORBA concept called “tie”. In part one, our server-based object was implemented by inheriting from the skeleton implementation **_sk_InterfaceName**, a class that was generated by OmniORB's IDL (Interface Definition Language) to C++ compiler. By inheriting from this base class, the developer can concentrate on implementing only the interfaces that have been defined in the IDL and not having to worry about all of the code that actually makes the CORBA communication possible. Multiple inheritance is sometimes used to allow for the inheritance of the skeleton provided by the IDL compiler, sometimes called the BOA (basic object adapter) implementation, as well as allowing the implementation to inherit from some other parent class. When the implementation is written in Java, which doesn't support multiple inheritance, this situation becomes problematic. Without multiple inheritance, it becomes impossible to inherit both a skeleton base class and another base class, such as an application framework.

To address this, the OMG specification defines a delegate class called the **tie** class. The IDL compiler for OmniORB generates an interface called **_tie_*`InterfaceName`*** to address this role, while VisiBroker's Java IDL compiler generates a Java interface called ***`InterfaceName`*Operations**. Rather than inheriting from the base implementation class, the developer instead implements a generated interface called an operations interface, which contains no methods, attributes or properties other than those defined in the IDL. The operations class is then passed in the constructor to a wrapper tie class which implements each method in the interface by delegating to the operations interface object. The tie class implements the orb functionality supplied by the generated base implementation. The tie object is then the component that is actually bound to the orb. Since the implementation of our interface is no longer inheriting from the base implementation, this frees up the developer to inherit from another base class such as an application framework.

In order to better understand how to implement a CORBA solution in Java, let's compare a Java implementation to a C++ implementation, which readers of parts one and two should be familiar with. There are several differences in the way a CORBA application is implemented in Java versus C++. Concentrating on the VisiBroker for Java implementation and the VisiBroker for C++ implementation, the difference can most obviously be seen in the number of files generated by the Java **idl2java** compiler and the C++ **idl2cpp** compiler. Basically, the idl2java creates about twice as many files as the idl2cpp compiler. The idl2java compiler even creates a new subdirectory to hold all the new files it generates. Certain flags can help control the number and types of files generated. When you run idl2cpp without any flags on, this very simple IDL file (example.idl) is generated:

```
{
   interface SimpleInterface
   {
      void SimpleOperation(in short x);
   };
};
```

Also, the following files are created:

- example_c.hh: contains the class definitions for SimpleInterface, along with supporting classes.

- example_c.cc: contains stub code to be compiled and linked with the client, which provides support functions (such as _ptr and _var definitions).
- example_s.hh: contains the definitions for the **_sk_Account** skeleton class for inheritance with the **bind** method, along with the tie classes for delegation in the **tie** method.
- example_s.cc: contains the internal skeleton's marshaling code, etc.

The same IDL file will generate the following files when compiled by the idl2java compiler (note these files would be contained in a subdirectory called Example):

- SimpleInterface.java: provides a simple public interface definition for the SimpleInterface declared in the IDL. This interface simply mimics, in Java, the interface defined in the IDL. The actual implementation of this Java interface is contained in the **_SimpleInterfaceImplBase** class (supplemented by the actual implementation you will write).
- SimpleInterfaceHelper.java: provides helper methods for SimpleInterface clients. Among these helper methods are the ever-important bind method overloads, as well as a **narrow** function (for use in the tie method).
- SimpleInterfaceHolder.java: provides a Java class that holds a public instance of a SimpleInterface object. It provides a wrapper class for a SimpleInterface object, which is necessary to allow the passing of SimpleInterface objects as **out** and **inout** parameters in function calls declared in an IDL interface.
- SimpleInterfaceOperations.java: provides classes that assist in the implementation of the tie method. (Not created if the **-no_tie** flag is given).
- _SimpleInterfaceImplBase.java: provides an abstract public Java class that implements the server-side skeleton for SimpleInterface. This base class itself extends org.omg.CORBA.protable.Skeleton, and implements Example.SimpleInterface. Your implementation inherits from this base when using the bind over the tie method.
- _example_SimpleInterface.java: provides simple code that you can fill in to implement the SimpleInterface object on the server side.
- _st_SimpleInterface.java: provides a Java class that implements the client-side stub, which proxies the SimpleInterface object on behalf of the client. The client makes calls on this proxy.
- _tie_SimpleInterface.java: provides the delegation class used to implement the tie method on the server side.

Exactly twice as many files are generated by default by the idl2java compiler as compared to the idl2cpp compiler. This is partially because of language differences between the two languages. Java has no user support for pointers (the language has support only for pass-by-reference for objects), so holder classes are needed to support **out** and **inout** IDL parameter types. Every ORB-defined, as well as every user-defined, type has an associated holder class defined to support the IDL **out** and **inout** semantics. You can think of holder classes as pure wrapper classes containing a value that is an instance of the actual fundamental class. Holder classes implement the **org.omg.CORBA.portable.Streamable** interface, which allows for the reading and writing of objects and streams. Holder classes are named by simply taking the base class name and appending “Holder” to it.

The idl2java compiler also generates a helper class for every interface. Helper classes offer a number of static methods which provide clients with vital functionality. These include the bind and narrow methods, which allow clients to connect to server-based objects. They also provide read and write methods to assist the Holder classes in translating between I/O streams and native object types. They also supply type code information that is useful when it comes to **Any** types and the Dynamic Invocation and Dynamic Skeleton interfaces. Type codes provide for runtime detection of type mismatches, along with metadata support for runtime type information. Since Java is primarily an interpreted language, it must be careful of added memory constraints. Helper classes help by off-loading several rarely used methods, such as bind and narrow, so that the actual object implementations can avoid loading these methods. You might call the **calculate** method a hundred times a second, but you'll usually call bind only once.

Beyond the generated helper classes, the Java and C++ implementations using CORBA look very similar. For example, the only true difference between finding a naming context under C++ and Java is the use of the helper class to do the narrow.

```
Mico C++:
CORBA::Object_var nsobj =
  orb->resolve_initial_references ("NameService");
CosNaming::NamingContext_var nc =
  CosNaming::NamingContext::_narrow (nsobj);
VisiBroker Java:
org.omg.CORBA.Object objRef =
  orb.resolve_initial_references("NameService");
org.omg.CosNaming.NamingContext rootContext =
  org.omg.CosNaming.NamingContextHelper.narrow
      (objRef);
```

Our example demonstrates a simple logging facility that makes use of the VisiBroker for Java Naming and Event Service, as well as demonstrates the use of the tie mechanism in Java. The example offers a Log Service and two clients: one supplies events (messages) to the Log Service, the other consumes (reads) those messages or events.

Our example for this article is a simple message delivery service in the form of a logger, implemented using classes that interact with the VisiBroker for Java

Event Service. A Supplier generates strings and then delivers them to a Log Service, a Java class that extends the Push Supplier interface. The Log Service publishes a function called **send** which allows one of its clients (a Supplier) to publish events (send messages) to the event queue. The send method forwards that event by pushing it onto the event queue. The PullConsumer, another client in the scene, binds to the event channel, then proceeds to pull the events issued by the Supplier from the queue. The example demonstrates both the use of the Naming and Event Services in VisiBroker for Java as well as the tie mechanism. (The LogService is implemented using the tie method.) The example has been kept simple in order to easily communicate the issues involved. Error handling, for example, has been kept to an absolute minimum so as to not obscure the foundational elements. Thus, the path a string travels through the system is as follows:

1. Supplier creates string.
2. Calls send on Log Service.
3. Log Service forwards string to Event Channel via **push**.
4. Event Channel buffers the string for Consumer.
5. Consumer polls the Event Channel for a new string.
6. Consumer retrieves the string from the Event Channel.

Listing 1. EventCannel.idl

The IDL for our logging facility is extremely simple (see Listing 1). It defines a module called logging and a single interface named LogService that implements a single function, called send, that accepts a single string parameter. This string is passed to the Log Service, which is then placed on the Event Channel, where it awaits being read by the Consumer. The Consumer polls the Event Channel periodically, checking to see if a new event (String message) has been delivered. If it has, it pulls that String from the Event Channel and prints it to STDOUT.

IDL modules are mapped, in CORBA's Java mapping, to Java packages. Therefore, the logging module in the IDL is mapped to a logging package that, by default in Java, is a subdirectory under the directory which contains the IDL file. It is the logging package (directory) that contains all the files generated by the idl2java compiler. When built, the *logging* directory contains eight Java files, generated by the idl2java compiler. The directory contains class definitions for the LogService interface, Helper and Holder classes which we mentioned above, and the tie and ImplBase classes for delegation and binding.

Listing 2. LJEventChannel.java

Listing 2 shows the **LJEventChannel.java** source, which defines two classes. LogServiceImpl and LJEventChannel. The LogServiceImpl class extends the

_PushSupplierImplBase base class, and implements LogServiceOperations. The LogServiceOperations class has the capabilities necessary for the tie mechanism, which we will use to connect to the **LogServiceImpl** object. The LogServiceImpl class provides the core functionality for binding to the proper VisiBroker Event Service channel. Since LogServiceImpl extends _PushSupplierImplBase, it is able to function in the role of a Push Supplier vis-à-vis the Event Service (for more information, see last month's article).

The meat of the LogServiceImpl class is in its constructor. When a new **LogServiceImpl** object is created, the constructor first binds to the ORB via an **org.omg.CORBA.ORB.init** call. Then, it connects to a particular event channel, “channel_server”, in order to pass strings via the Push Consumer proxy it creates. The actual process of connecting to the Event Channel was covered in detail last month; however, we will summarize the steps here briefly.

First, the bind method is called on the EventChannelHelper class that is part of VisiBroker's Event Service. The Naming service is not necessary to connect to the Event Service, because the osagent facilitates the binding by using its own simple naming service.

Once an EventChannel is bound, a Push Consumer proxy is obtained from the Event Channel, and then our LogServiceImpl object is connected to the proxy. This allows us to make calls on the proxy. From this point on, any supplier who calls the send method on our **LogServiceImpl** object will cause our implementation to call the push method on its Push Consumer proxy. This is done with the line **_pushConsumer.push(message)**. Notice that, as usual, we've packaged our string to be sent in an **Any** type, which the Event Service requires for transmission.

Class LJEventChannel consists of a single **main** method which, after binding to the ORB and initializing the Basic Object Adapter for our object, creates a new **LogServiceImpl** object described above. The tie method is used in the binding process, which means we will be using delegation instead of inheritance in our communication with the object's implementation. After we have tied to our new LogServiceImpl delegate named “new_service”, we then bind that service object to the Naming Service under the component path **Linux Journal:LJEventChannel**. This allows any client object on any machine in the visible network to connect to our new_service object via this naming convention, without having to know the name or IP address of the hosting machine.

Once the new_service delegate has been bound to the Naming Service, the BOA is advised that the object is ready and available and the implementation of the server is complete.

At this point, an implementation of the LogService interface defined in the IDL has been created and published and is now available for calls from clients wishing to use its send method to post messages to the Event Channel.

Listing 3. PushSupplier.java

Listing 3 shows PushSupplier.java, the supplier in our application. Class PushSupplier consists entirely of a single main method, which after initializing the orb with org.mg.CORBA.ORB.init, stores the name of the supplier which was optionally entered on the command line when the supplier was started. This arbitrary name allows you to name your suppliers Supplier1, Supplier2, etc., so that you will know, on the consumer side, which supplier's string was obtained. After initializing the ORB, a LogService reference named **logger** is created. Then we enter a try block, which seeks to bind to the **LogServiceImpl** object already created using the Naming Service. The supplier calls the resolve_initial_references method on the orb object, obtains a root context, creates the appropriate name components, and calls **resolve** on the root context using the created name component array. The generic object reference returned is then narrowed by a call to narrow using the **LogServiceHelper** object.

Assuming the **logger** object is not null, we then enter a loop that continually sends a string to the LogServiceImpl object via its send method. This continues until the user interrupts the supplier with **ctrl-C**.

Listing 4. PullConsumer.java

Listing 4 shows the PullConsumer class, which extends the _PullConsumerImplBase base class of the VisiBroker Event Service. After initializing the ORB and BOA, the PullConsumer object attempts to bind to the Event Channel by calling the bind method on the EventChannelHelper object. Then a new PullConsumer object is created, which implements the **disconnect_pull_consumer** method required by a PullConsumer. The reason a new PullConsumer must be created is because we need an object reference to pass to the BOA's **obj_is_ready** and the proxy Pull Supplier's **connect_pull_consumer** method. Since we are in the main method which is static, no “implicit this” reference is available to us. Therefore, we need to create a *new* object in order to obtain a reference to pass. Once a new PullConsumer object is created, the BOA is advised that the object is ready. After this, a Pull Supplier proxy is created via a call to the bound channel's **obtain_pull_supplier** method. Once the proxy is created, the PullConsumer object is connected to the proxy by calling connect_pull_consumer on the proxy, passing it the PullConsumer object.

At this point, a **while** loop is entered, and the consumer continually calls **try_pull** on the Pull Supplier proxy. If the proxy finds an event, then that event is returned, the PullConsumer object prints that message to standard output and the loop restarts.

You can try out this application by first running a single PushSupplier and PullConsumer locally on the same Linux box. (See the README.install and README.run instructions that accompany the code for details on the setup, building and launching of the applications.) Then you might want to launch another PushSupplier and notice that the PullConsumer automatically begins to process events from the new PushSupplier as well. (You might want to name the PushSuppliers as they are launched—see the README.run instructions on how to do this.) Then launch a new PullConsumer over on the Windows (or other OS) box, and watch how events from the two suppliers are conveyed to the two consumers, one of which is now running on the Windows machine. Finally, launch another PushSupplier on the Windows machine and watch how two consumers process strings created and delivered to the same event channel by three separate suppliers. Even though the code is simple, the project implemented here is quite capable and has some broad implications which you should explore.

Conclusion

In these three articles, we have attempted to introduce you to CORBA programming on Linux. Linux is a robust platform for developing CORBA applications, and the CORBA is quite versatile in terms of its capabilities, services, platform independence and language independence. We hope these articles have spurred your interest in both CORBA and Linux and wish you success in exploring these issues more fully on your own. For those who wish to learn more, visit the Free CORBA Page at adams.patriot.net/~tvalesky/freecorba.html. It has continued to grow in terms of subject matter as well as quality, including some information on the CORBA implementation in Java 2.

Resources

Mark Shacklette is a principal with Pierce, Leverett & McIntyre in Chicago, specializing in distributed object technologies. He holds a degree from Harvard University and is currently finishing a Ph.D. in the Committee on Social Thought at the University of Chicago. He is an adjunct professor teaching UNIX at Oakton Community College. He lives in Des Plaines, Illinois with his wife, two sons and one cat. He can be reached at jmshackl@plm-inc.com.

Jeff Illian is a principal with Energy Mark, Inc. in Chicago, specializing in electric utility deregulation and distributed trading technologies. He holds a degree from Carnegie-Mellon University in Operations Research (Applied Mathematics). He lives in Cary, Illinois with his wife, son and daughter. He can be reached at jeff.illian@energymark.com.

Archive Index Issue Table of Contents

Advanced search

Interview

David Phillips

Issue #63, July 1999

Mr. Phillips gets the low down on 4Front technologies and what's happening in the world of sound.

4Front Technologies is in the business of supplying UNIX sound card drivers for AIX, BSD, Solaris, SCO, HP-UX, Linux and other systems. The company has maintained a particularly amenable relationship with Linux for several years: 4Front's Hannu Savolainen has provided the kernel sound driver and the system sound applications programming interface. The company also sells an enhanced supported version of the kernel driver.

With warm greetings to Dev Mazumdar and Hannu Savolainen, we begin:

David: When, where, why and how did 4Front go into business?

Hannu: Actually, our story began in August 1992 after I got Linux working for the first time. I had started writing a Sound Blaster driver for Minix, but dropped the project for various reasons. I ported it to Linux and released the first hack after a couple of weeks. What happened after that is a long story.

Then in September or October 1994, Dev contacted me. He was selling an MCA-based sound card and asked if he could port my driver to AIX. After a few e-mail messages, we decided to join forces and port the driver to everything that moves.

Initially, our idea was to sell commercial Open Sound System (OSS) only for operating systems other than Linux, since there didn't seem to be any market for OSS/Linux. However, we decided to release OSS/Linux because the software was ready. We just had to check that there were no copyright issues.

Dev: I began writing the sound drivers for the Creative Labs Sound Blaster MCA on IBM's new RS/6000s that had MCA slots in 1992/3. Back then, the Internet was small and only about three or four of us were doing sound drivers for UNIX.

David: Who currently does what at 4Front?

Hannu: By definition, my responsibility is design and programming while Dev does support, marketing and also maintenance of our web site. However, these days Dev does almost as much programming as I do.

Dev: I also run the copier and take out the trash! ;-) In all seriousness, my main functions are mainly to run the day-to-day business operations of 4Front. I make customer calls and visit various companies that we partner with. I always defer to Hannu's technical expertise whenever I'm bug hunting.

David: How many sound systems are supported, i.e., sound cards, on-board and external?

Hannu: Actually, we implement support for sound chipsets rather than sound cards. The same chips are usually used by many sound card vendors, especially in the Far East. I haven't the faintest idea how many different sound cards we support.

Our configuration program currently knows 250 different “sound card” names. I think about 150 of them are unique in one way or another and have required at least minor modifications to OSS. The remaining 100 names in the list are some kind of alias names for some chipsets. We had to add them because so many customers keep asking why this or that sound card is not in our supported card list.

We support or are currently working on about 40 to 50 “major” sound card architectures or chipsets.

Dev: A better picture is to multiply the number of sound chips by the number of UNIX versions by the features for each sound device (audio, MIDI, mixing, synth, etc.) to get the size and complexity of OSS!

David: How do you decide to support a card or system?

Hannu: Mainly, decisions are based on customer feedback, but sometimes just because some cards look particularly interesting in some way. It's actually easy to figure out which chipsets need to be supported by looking at how many potential customers are asking if they are supported.

At this time, our focus is on the popular PCI-based chipsets made by big vendors such as Creative, ESS, Aureal and Yamaha.

Dev: As far as operating systems are concerned, system vendors are coming to us and requesting OSS, since they've seen it in action on Linux. Additionally, with the booming number of applications with OSS, we're finding many UNIX and real-time vendors are taking notice of the fact and are approaching us. A recent example is HP (Hewlett-Packard). We approached HP a while back about porting OSS to HP-UX and only now have we entered into a development agreement because of the attention Linux has been receiving.

David: What sort of work goes into creating, testing and eventually marketing a sound driver?

Hannu: The first thing is to get the programming specifications for the sound card or chipset. This usually takes a very long time, since hardware manufacturers simply don't have suitable documentation available immediately.

The programming can start after getting the specs. The tough part is getting the first sound from the device. With complex 3D PCI audio accelerators, this takes months, since the complexity of these chips is at least in the i286 class. Usually everything has to be done in complete darkness, since there is no way to see what the chip is doing internally. After the first sound, everything is much easier since we can at least hear what the chip is doing.

After the card seems to work, we test the driver for about a month in two or three different machines. After that time, it's released as a beta version so that everyone willing to try it can do so. Finally, the beta status is removed after several months, if no significant problems have been reported.

Actually implementing low-level drivers for different sound cards is just one part of the picture. Another big task is trying to figure out why certain applications are constantly causing problems. If something in OSS should be improved, we do it. However, we will not try to fix buggy applications by changing our code.

Another big task is making the installation easier. After all, a large number of our customers are those who don't want to spend their time getting things working. If they don't get sound working quickly, they just leave it alone. So we have to make it possible to download and install OSS in two minutes without reading any manuals. This is actually an endless battle, since Linux and the PC architecture is a rapidly moving target. All Linux distributions are slightly different, as are the different kernel versions. We currently support all 2.0.*x* and 2.2.*x* releases.

Finally, users have an infinite number of different CPU, chipset, SMP/UP, sound card and other hardware combinations (I almost forgot to mention overclocking). In addition, many users compile the kernel themselves and install all kinds of UFPs (unidentified flying patches) to their machines. It's a big challenge to identify the problems caused by these differences and find a workaround. We generally release a new OSS revision every two to four weeks (usually after a new kernel version is released) just to address these kinds of issues.

David: Please describe how your relationship with Linux has evolved over the years.

Dev: I have a UNIX background and worked on the IBM RT 4.2 BSD-based kernel, as well as the AIX 3*x* kernel for IBM (in a joint project with USC). One of my first drivers was the console driver for the IBM Megapel 5081 graphics card, and following that was the first color X Window System X11R1 server for the card.

I began to run Linux for the first time in 1994 when 1.2.13 came out (until then, I had a monstrous AIX/RS6000 530 at my desk). From my point of view, Linux has the best device driver programming environment, and it's developer-friendly compared to other UNIX environments. Of course, supporting customers on Linux is much more difficult than on RISC-based UNIX systems, due to the PC architecture's ISA bus.

Hannu: I have a minicomputer background. I started programming with an HP3000 Series III at Lappeenranta University of Technology (http://www.lut.fi/) in 1981. For this reason, I never liked DOS or anything else that ran on a PC. I had my first contact with UNIX (HP-UX) in December 1984, and it was the only OS I was interested in. So it was natural that I bought Minix in May/June 1991 soon after my first 386/25 PC. I installed the i386 extensions and GCC and had just finished porting a myriad of applications to it when I made my first contact with Linux. I was one of the lucky guys who saw the historic message posted to comp.os.minix by Linus (August 25, 1991).

At that time, I was already disappointed with Minix and I had started to plan my own OS called SKUNKX (for Smells liKe UNiX) but had not actually started writing it. My immediate first reaction about that no-name UNIX clone was slightly negative, because it was a competing project. I forgot the whole thing (both Linus and my project) for some time until I saw some information about the release of the first Linux version. I decided to skip the first ones, mainly because my machine had nonstandard disks (SCSI) not supported by Linux. After the first AHA1542 driver, it took a few months before I figured out how to get the card to work with Linux, which used a different IRQ than was available in my system. I don't remember what the solution was, but eventually I got

Linux installed in summer 1992. After that, I formatted my disk for Linux and have used it as my main operating system ever since. It was amazing how reliably it worked even in those days.

As I mentioned earlier, I soon started working on the Sound Blaster driver in my spare time. After the first release, it quickly got the attention of a few users who made suggestions and sent some fixes. Together with Craig Metz, we expanded the Sound Blaster driver to support the Pro Audio Spectrum PAS16. A few months later, I started adding support for the Gravis Ultrasound and received significant help from Greg Lee. Within a year, support for some other cards was added and finally the driver was included in the kernel.

In addition to developing OSS, I also use Linux for almost everything I do with computers, mainly all kinds of programming, e-mail, news, Net surfing, web server, Samba and reading documentation with Acrobat. All these features have worked very well since I started using Linux, or actually after I upgraded my machine to run X. Now, after getting StarOffice, there really aren't many reasons to use any other OS. I'm sure this will get even better in the near future.

As Dev said, Linux is a very good (if not the best) environment for device driver development work. Hardware interfacing is very easy, since Linux doesn't try to hide the iron behind a stack of abstraction layers. Also, Linux doesn't try to enforce any particular programming model, so you can do what you want in the way you deem most suitable.

David: How does 4Front regard the evolution of Linux itself?

Dev: As a commercial company, we would love to see Linux go to the desktop because that's where sound support makes a great deal of sense compared to the server arena that Linux has now begun to dominate.

Make no mistake—commercial applications will continue, and a lot of bright minds are tied up writing “non-open-source” applications. We expect the Open Source movement maturing and becoming all-inclusive of proprietary and GPL applications. We also expect to see a number of companies to use our hybrid open-source/proprietary model. Many companies will release part of their application under open source and still sell a proprietary version that has additional functionality, yet maintains API compatibility with the free version.

This allows for clean healthy competition between the open-source and proprietary development teams. We are in some ways competing with the OSS/Free kernel drivers that are now maintained by really talented folk like Alan

Cox. Some cards are supported only in OSS/Free, such as support for the Apple PowerMac audio and the Sun SPARC SS20 audio.

Hannu: I think our first and biggest mistake with OSS was the stupid idea that we could control all sound driver development for Linux. Actually, it's best for all (even for us) that independent groups continue developing the freeware drivers. This gives us some peace to concentrate on the features and sound cards most important to our current and future customers. Of course, this means there will be competing drivers for many sound cards, but I actually see this competition as a challenge rather than a problem.

The fact that we are making commercial, binary-only software doesn't mean we are against open-source ideology. We are here only because Linux and the original “VoxWare” sound driver was Open Source. Our original idea was to maintain one common source base for both OSS and OSS/Free. In this way, it would have been possible to easily incorporate most additions made to OSS to OSS/Free. This model was brain-dead, because it made it very difficult to adopt contributions to OSS/Free. The current model for how OSS/Free is maintained is much better. The internal architecture of OSS/Free is now significantly different than OSS, so our code is not very useful for the OSS/Free team. However, I'm sure we can find a way to make contributions to OSS/Free in the future.

Dev: You will see our concept used by many open-source developers—Sendmail, Ghostscript and others are examples of how developers have come up with a “proprietary/open-source” model to generate revenue. It takes complete commitment from the authors, and this means giving up day jobs and concentrating on the software. We have received kudos from many customers about OSS and this is what keeps us going.

4Front has been profitable, and we are not funded by venture capital or by sound-card or OS vendors. We may be in a position to hire some people next year if the pace of Linux's growth continues—it's all about being able to grow from a garage shop into a professional company by doing what you do best.

David: What can Linux users expect from future releases of 4Front drivers?

4Front: We are currently working on support for all those popular PCI sound cards. Support for some high-end audio cards should be released during the second half of this year. Features of the base OSS will remain almost unchanged this year, but we are implementing some features which will make OSS useful in simulators (3D sound effects) and radio (on-air) systems. Next year, the focus will most likely be on adding new features to the API, at least on the MIDI programming side, like support for DLS samples in the SoftOSS virtual synth and support for MIDI patchbays.

Of course, we will continue our work in making OSS even easier to install and use than it is today.

We're also looking at more OS support. For example, we'll be working on BeOS support later this year. We may look into Mac OSX when it comes out later this year, as it's rumored to have a UNIX-like core.

David: Is there anything you would like to add beyond these questions?

4Front: We would like to thank all the developers who have been writing audio applications for Linux. OSS couldn't have done it without your support. We hope your foundation work will bring more talented people from other operating systems into the Linux and OSS platform.

David: Please tell *LJ* readers something about what you guys do outside of your involvement with the company.

Hannu: My family takes up most of my time outside work. My hobbies are reading, listening to music, movies, bicycling, swimming, cooking and digital photography. I collect all kinds of electronic gadgets. I'm also trying to get some experience in the electronic engineering side, but don't seem to have enough time.

Dev: My hobbies include playing volleyball on the beach all year (one of the luxuries of living in Los Angeles). I also like skiing and golf (I'm a horrible hack at that!). I used to build electronic gadgets as a kid. I'm currently engaged to a wonderful woman and will be getting married in September.

4Front Technologies4035 Lafayette Place, Unit FCulver City, CA 90232Phone: 310-202-8530Fax: 310-202-0496Email: info@opensound.comURL: http://www.opensound.com/

David Phillips (dlphilp@bright.net) is a composer/performer living in Ohio. Recent computer-music activities include an ambient composition for the artist Phil Sugden, lecturing on computer-music programming languages at Bowling Green State University, and maintaining the “official” version of Csound for Linux. Dave also enjoys reading Latin poetry, practicing t'ai-chi-ch'uan, and any time spent with his lovely partner Ivy Maria.

Archive Index Issue Table of Contents

Advanced search

Building a Linux Certification Program

Dan York

Issue #63, July 1999

A report on a community-based initiative to develop a professional certification program for Linux.

Some time ago, the idea for a certification program for Linux existed only in the minds and discussions of individuals and small groups of people in different locations and all working separately. Over the past year, many of these separate people have come together in a community-based effort to define a Linux certification program. Who is this group? How did they come together? What have they accomplished to date?

We call ourselves the Linux Professional Institute (LPI). As stated on our web site (http://www.lpi.org/), our mission statement is:

> We believe in the need for a standardized, multi-national and respected program to certify levels of individual expertise in Linux. This program must be able to satisfy the requirements of Linux professionals, as well as organizations which would employ or contract them.
>
> Our goal is to design and deliver such a program from within the Linux community, using both volunteer and hired resources as necessary. We resolve to undertake a well-considered, open, disciplined development process, leading directly to the establishment of a recognized and widely-endorsed Linux certification body.

With these words, we put in writing our overall goal, a remarkable initiative that emerged from mailing list discussions over much of the past year. In this article, I will discuss how the initiative evolved, what our current plans are, and how you can become involved.

A Brief History

One part of our effort began with an article I wrote for the October 1998 issue of *Linux Gazette* (www.linuxgazette.com/issue33/york.html). In that article, I outlined the reasons I felt a certification program would help the growth of Linux, and encouraged people to contact me. The response was tremendous, and we immediately established a mailing list to help coordinate our discussions. Along the way, we found other individuals and groups who were also working on certification and tried to find ways to work together on this certification effort.

Meanwhile, a separate effort was underway, coordinated by Evan Leibovitch of the Canadian Linux User's Exchange (CLUE). Starting in April 1998, they established a mailing list focusing on certification and had gone quite far in discussing how a certification program might be implemented. The list grew rapidly and came to include people from around the world. At one point, their list included representatives of three distributions: Caldera, SuSE and Debian.

Last November, Jon "maddog" Hall of Linux International introduced me to Evan. We immediately saw the similarities between our two efforts and explored ways of combining the energy of our two groups After our groups united, we implemented an organizational structure to help work together and proceed along multiple paths to develop our program. As we proceeded, the initiative attracted a highly talented pool of volunteers, many of whom contributed (and continue to contribute) very long hours toward bringing our collective program to reality.

The Need for Certification

Many of us believe a certification program for Linux *will* occur. The question is whether we want that certification program to come from a particular vendor or have it evolve from within the Linux community.

The people who have come together behind our effort believe there are a number of reasons why certification is necessary. Briefly, we feel certification will do the following:

- Accelerate corporate adoption of Linux. As more and more people learn about Linux and pursue certification, they will speed up the adoption of Linux within corporate environments.
- Create industry recognition. Microsoft, Novell, Lotus and others have spent millions of dollars convincing the IT industry of the value of certification. A Linux certification program will allow those who value certification to see that Linux "has emerged as a viable option".

- Counter the “no-support” argument. As more candidates earn certification, it becomes a statistic that can be used to indicate how many Linux support professionals are available in the IT industry.
- Provide a learning path for new users. Often, people who want to learn about Linux do not know where to start. A certification program can provide a path for learning.
- Provide an organizational mechanism for training centers and publishers who want a path to help educate their clients (students or readers).
- Expand the marketing of Linux. Every training center and every book that focuses on the path to certification of a product creates more marketing of that product. Until now, marketing budgets have been used to promote other operating systems. We want to see a share of this money promote Linux and recruit new users.
- Turn students into advocates. If students learn about Linux and how to install, configure and use the operating system, they will become advocates for Linux as they move into the IT industry. People recommend products they know. We need them to know Linux.
- Provide other means of employment for Linux-skilled individuals. Each person who can be employed writing or teaching about Linux becomes yet another advocate, potentially full-time, for Linux.
- Recognize capabilities of Linux professionals. A well-done certification program provides a mechanism to recognize the accomplishments of individuals who use Linux.
- Assist in the hiring process. Most controversially, a certification program can assist a hiring manager in understanding what level of expertise someone has. It cannot be used as the sole criterion and is not a replacement for years of experience. However, many IT managers want to start using Linux and are seeking people knowledgeable in Linux. If the managers don't know Linux, how can they be certain of the type of background someone actually has? A certification program helps managers know that an individual has at least a basic level of Linux knowledge.

A longer description of some of these points can be found in my October 1998 article in *Linux Gazette*.

Developing a Consensus

After recognizing the need, the people who joined our mailing lists rapidly came to a consensus on a number of issues, including:

- The cost of attaining Linux certification should be as low as possible. Costs of exams should be targeted to cover delivery of the exam, with perhaps a slight portion to help offset development of future exams.
- The mechanism we develop for delivering Linux certification must be global in scale. People in any nation must be able to take exams toward certification.
- The Linux certification program must be distribution-neutral and vendor-neutral. It should not be seen as biased toward any one Linux distribution, nor toward any vendor of education or other services.

Through our program development, we decided we would handle distribution differences through a distribution-specific exam. We would create a separate exam for each distribution and cover the items unique to that distribution, such as installation, graphical administration tools and file locations.

Exam Structure

In order to be global and able to deploy on a massive scale, we also decided that for at least the first levels, we would need to use standardized computer-based testing systems such as those offered by VUE and Sylvan Prometric. We debated at length about utilizing web-based testing, but could not at this time determine any mechanism for preventing fraud. As long as someone could have a friend nearby providing answers (or taking the exam for them), the potential for abuse is too high. We agreed to continue monitoring technologies in the hope that someday a solution might be found.

In the meantime, we will be working with the computer-based testing vendors to make our exams available throughout the world. We are also considering other forms of proctored testing in locations where testing centers are readily available. For the highest level of our certification program, we are still debating whether to have some form of “hands-on” testing. That discussion and decision is still ongoing.

Throughout our discussions, we also realized we wanted our program to focus only on certifying individuals. We want multiple paths to certification. One person might download our exam objectives off the Web, work with their system at home, then go and take the exam, paying only the minimal cost of the exam. On the other hand, a candidate could also spend a great deal of money on instructor-led classes to prepare for certification. Someone else could also go to the bookstore and buy books that would prepare them for certification. Computer-based classes and web sites will certainly be options as well. We decided that our effort would be spent on *certification*, leaving the *education* of candidates to training centers, publishers and others interested in providing such services.

Our program is a community effort open to all who want to be part of the process. For that reason, we continue to use open, public mailing lists for the majority of our work.

Finally, a common refrain has been that we do not want our program to be as weak as many perceive other IT certification programs to be. We want to be sure this program is done right.

Our Program

While developing our program, Evan Leibovitch brought in much of the work developed through the CLUE mailing list. The mailing lists discussed and debated the program at length. Eventually, a Program Committee led by Tom Peters took over responsibility and fleshed out more of the details.

Through our program, certification will eventually be available at three levels, though the names of these levels have not yet been finalized at the time of writing (March 1999). The exams will be developed gradually, with the required exams for the first level being created before the second level and so on. In the short term, this will allow participants to complete lower-level certification while more-advanced exams are under development.

Content for the exams is presently under active development, although by the time this article is published, much of the first level of exams should be nearing completion. Although the exact content and objectives for each exam are still under development, the exam structure (we chose to label individual tests as "T#") is shown in the sidebar "Exam Structure".

Our intention is to create a separate T2 exam for most major distributions. The list of T5 exams is merely a set of examples of the kind of specialized exams which may be produced at this level; the exact list has not yet been determined.

As we have currently defined the program, candidates will be required to pass certain tests to reach the various levels of certification:

- To achieve Level 1, an applicant must complete T1 and one or more of the T2 exams.
- To achieve Level 2, an applicant must complete Level 1, as well as exams T3 and T4.

- To achieve Level 3, an applicant must complete Level 2, as well as any two of the T5 exams.

The choice of T2 and T5 exams completed will be indicated on the participant's certificate as endorsements.

Where We Are Today

Throughout early 1999, much of our work occurred in the individual committees focused on very specific tasks. As outlined below, we developed a job analysis survey, began our public relations, built an Advisory Council to provide additional feedback and formed an independent nonprofit corporation.

Among our pool of volunteers were several individuals with degrees in psychometrics, who spent considerable time working on methods to validate the results of our exams. One of these individuals, Scott Murray, chairs our Exam Development committee and has been working hard on ensuring our exams are developed in the best method possible.

In March and April 1999, Scott and Tom Peters developed a web-based system through which we conducted an extensive job analysis survey. The main purpose of this survey was to aid us in developing the objectives of our first level of exams. Hundreds of volunteers took time to complete our surveys and help us statistically validate the tasks Linux system administrators do on a daily basis. The results of this survey were used to help us derive the exam objectives we have posted on our web site.

During this time, we wanted to ensure our program met both the needs of the Linux community and the organizations which will employ the successful candidate; therefore, Evan, Chuck Mead and I along with other members of our Corporate Relations Committee built an Advisory Council. This council consists of individuals and organizations who can provide us with the feedback we require. Members of our Advisory Council are part of a private mailing list to which questions are occasionally posted and feedback solicited. Their assistance is sought to help guide the overall direction of the LPI program, as well as in helping solve questions that arise from time to time within the mailing lists where a wider industry perspective may be useful. As a consultative body, the Advisory Council provides input to the LPI Board when it makes decisions related to LPI. In the process of building our Advisory Council, we had very successful meetings, both at trade shows such as LinuxWorld and CeBIT, and also separately with individual people and companies. We announced a large council including representatives from several distributions, Linux International, *Linux Journal*, UniForum, publishers, information technology companies and others who believe in the need for Linux certification. We appreciate their support and assistance in making our program a reality. Visit our web site (see Resources) for the full listing of our Advisory Council.

Meanwhile, Evan and the Public Relations Committee were collecting names and addresses of reporters and web sites. Evan coordinated our work to regularly distribute news releases publicizing our efforts. His work resulted in a great increase in the number of visits and added participation in our plans.

We also began to work with the System Administrator's Guild (SAGE), a special technical group within USENIX, whose members are working on developing a certification program for UNIX. We shared information about efforts and designated a few individuals to act as liaisons between the programs.

Finally, our Steering Committee began the process of becoming a formal Board of Directors and incorporating as a nonprofit corporation. Our board also began the process of submitting funding proposals to finance the exam development already underway.

How You Can Help

By the time you read this, several of our exams should be nearing completion. Yet even as those exams are nearly done, we have many more still to develop. Over 200 people are now on our various mailing lists, and there is no shortage of tasks to complete. Please visit our web site, read about how you can become involved, and join in our efforts to make a strong certification program for Linux.

Conclusion

It has been a wild ride since we began our discussions last fall. We have had vigorous debates and put in some very long hours. Above all, though, our effort has shown the power of many people working together to accomplish a common goal. We've been able to take on large tasks and accomplish them primarily because we could divide the effort between many people. It has truly been a community project which we believe will result in the finest certification program in the information technology industry. We invite you to visit our web site and join with us.

Resources

Dan York (dyork@linuxcare.com) is a member of the Board of Directors for the Linux Professional Institute. He has been a technical instructor and training manager within the corporate training industry for nine years and has been working with the Internet and UNIX systems for 13 years. He is also a member of the Certification committee of the Systems Administrators Guild (SAGE—a division of USENIX). He is employed by Linuxcare, Inc., (http://www.linuxcare.com/) to work full-time on helping develop the LPI certification program.

Archive Index Issue Table of Contents

Advanced search

Focus on Software

David A. Bandel

Issue #63, July 1999

netsaint, UdmSearch, phpgen and more.

Last month, I had just installed Caldera's OpenLinux Beta 2.2 and was having problems with **ssh**. Following the suggestions of Stephan Seyboth (sseyboth@linuxland.de) and Erik Ratcliffe (erik@caldera.com), I finally got a working ssh. Stephan suggested I add **-D_GNU_SOURCE** to the top-level Makefile, and Erik suggested I comment out any offending **HAVE_SYMBOLS** in the config.h file. The define Stephan suggested is not a silver bullet, but it did make a difference in some cases.

I also ended up commenting out **HAVE_UTMPX** in the sshconfig.h, and ssh compiled fine. I am now running **egcs**, which is a bit different from **gcc**. Apparently, OpenLinux is much happier being told it is working with GNU source. OpenLinux uses glibc-2.1, and **utmpx** exists in glibc-2.1. However, somewhere between glibc-2.1, egcs and ssh, this symbol just isn't recognized (or not handled properly) during the build.

This month's packages were all compiled and installed on Caldera 2.2. I haven't seen Red Hat 6.0 yet, but I will guess it will have egcs, glibc-2.1 and a 2.2.x kernel. So, Red Hat users will likely see the same things I have. Since these applications were built on my system, they will most likely build on Red Hat, the new Slackware system when it is released, SuSE and Debian. I no longer offer any assurances about packages compiling on glibc-2.0.7 systems, since I no longer test them on that system.

netsaint:http://netsaint.linuxbox.com/

netsaint is a network monitoring tool that can monitor network services and notify administrators of problems via e-mail or page. Unlike the “Big Brother” package (http://maclawran.ca/bb-dnld/), netsaint doesn't require any client-side installations. Initial installation and configuration can be difficult and would be enhanced by a web configuration tool. Once installed, you can review (but not change) the configuration from a web browser. The current status is also monitored from the web browser. It requires glibc, Apache (or another web server) and a (preferably graphical) frames-capable web browser.

UdmSearch:http://search.udm.net/about/

UdmSearch is a web indexing tool that uses MySQL to store words and references found on web pages. It is extremely easy to configure and use. Searches are quite fast, and common words to exclude can easily be added to the package's list. It is also simple to embed the search function into your own pages. The search function can use either **php3** or a CGI program (both included). Some preliminary results from large sites suggest that, once indexed, this search tool offers a faster search engine for a large web site than most native web sites have. It requires MySQL, Apache with php3 compiled with MySQL support, glibc, libm, libnsl, mysqlclient and a web browser.

phpgen:http://www.byggsoft.se/~mikaelu/

phpgen looks like a good start toward automating the creation of **php** web pages. Since much of this type of code is repetitive and subject to error, phpgen could be a welcome addition to your toolkit. It could use a few more instructions on the web page where creation takes place, and maybe a few more Themefiles, but the code is usable. It requires MySQL, Apache with php3 compiled with MySQL support and a web browser.

stamos:privat.schlund.de/K/Knut_Grahlmann/english/stamos.html

stamos (some things about my operating system) will display certain statistics about your system, including OS version, RAM, load average, bogomips, uptime (requires **uptimed**) and hard drive usage. When run, stamos creates a web page with this information nicely formatted. This Perl script reads uptimed and some /proc files. It could easily be extended to provide more information, and optimally, a way for root to change some of the proc files via the web interface. It requires uptimed and Perl 5.

quicklist:http://www.quicklist.org/

quicklist is a working, finished-looking start on a list maker. You can concoct address books or phone lists and quickly be entering data. The lists are saved as ASCII text files. Standard quicklist format is backslash-separated data. You can also specify tab-separated, comma-separated or HTML format for the file. The only annoying thing I noticed was that the file extension used the abbreviated “.htm” rather than the standard “.html”. Quicklist also had trouble reading an .html file it wrote, although other file types could be read just fine. A number of options remain to be completed, but this package looks very promising. It requires gtk+-1.2 (including gdk and glib), dl, Xext, X11, libm and glibc.

stunnel:http://mike.daewoo.com.pl/computer/stunnel/

In the beginning was ssh, and ssh took care of secure TELNET and FTP sessions. Now we have **stunnel**, which is meant to complement ssh. The stunnel program does not do TELNET or FTP, but it does permit secure communications with SMTP servers, POP servers and others. Note that the home page and FTP site are located in Poland, so there is no hassle with US ITAR laws. When you build stunnel, you can build it with either the ssleay library or the openssl library. If you want stunnel to read your hosts.allow file, you can also build it with libwrap, the tcp wrappers library. It requires glibc, nsl, pthread, ssleay or openssl, and optionally, libwrap.

David A. Bandel (dbandel@ix.netcom.com) is a Computer Network Consultant specializing in Linux. When he's not working, he can be found hacking his own system or enjoying the view of Seattle from an airplane.

Linux and E-Commerce

Yermo Lamers

Issue #63, July 1999

The experience of one company selling Windows software using Linux to build a reliable e-commerce solution.

Through the use of Linux, open tools, and a few commercial components, my company built a reliable e-commerce solution that through its flexibility has enabled us to make our business more effective and explore opportunities that would otherwise have been unreachable.

The capabilities required to implement such a system used to be solely in the domain of larger organizations that could afford large cash outlays and dedicated personnel. Linux has changed this by making a top-quality, open platform available at virtually no cost.

Background

In the beginning, we were selling our Windows shareware stock-tracking software, Personal Stock Monitor, through a third party. It soon became clear this service didn't offer the flexibility we needed. When looking at other options, we realized no third party would be able to do what we wanted. For starters, we wanted to be able to do the following:

- Make the purchase process as easy and quick as possible.
- Get to know our customers better and take the opportunity to get more feedback from them.
- Experiment with various business ideas, change our pricing, provide upgrade incentives, provide discounts, etc.
- Track sales through multiple distribution channels and get a much better understanding of how and why people were buying our software and, more importantly, why people weren't.

Clearly, we needed a better solution that could be changed quickly as the need arose without a great deal of effort or expense. Being a small company, we were constrained by limited resources. However, we were convinced that doing e-commerce ourselves was a business necessity.

The Business Case for Linux

After looking at a number of options, we chose to build our own solution and base it on Linux for reasons both technical and business-related, including:

- We could use inexpensive hardware and share it between a number of tasks. This meant we didn't need to buy an extra box and didn't need to upgrade the one we had.
- We could administer the server remotely as easily as from the local keyboard. For a small company like ours, this was one of the many reasons we didn't even entertain using Windows NT.
- Linux is rock-solid reliable. As eight months of operational experience would show, Linux never seems to crash. It just runs, and as a result, we have more time to focus on business.
- All development tools and applications we needed are available for Linux and most of them are either Open Source or carry the GPL. All are of high quality.
- The “openness” of Linux provides a significant business benefit. We can always get the answers we need when we need them at no cost. We've never been slowed down due to lack of information when using Linux.
- All critical applications we needed were available for Linux.

We could have chosen other options that would have worked equally well, but they would have cost significantly more money and required much more expensive hardware. For our particular needs, we couldn't find anything we believed could do the job better at any price.

The Pieces and Parts of an E-Commerce Solution

Once we decided on Linux, the rest of the system fell into place.

Transaction Service

The first thing we needed was a way to authorize transactions. We looked at a number of e-commerce tool providers with varying levels of sophistication. It turns out these e-commerce companies provide the equivalent of the credit card machine you see at convenience stores. This means all order tracking, accounting features, reconciling, demographic reporting, feedback gathering and interactions with the customer along with most of the administrative features you need are left for you to provide.

For technical reasons, we ended up choosing the Cybercash service. This provides a library of C routines and Perl modules supported under Linux. Cybercash calls this software development kit (SDK) their Merchant Connection Kit. It's essentially a credit card transaction SDK and makes no assumptions about the rest of your business. It provides the kind of flexibility we need.

The Cybercash account didn't cost any money upfront, but it did have a transaction fee. More information is available on their web site at http://www.cybercash.com/.

We especially liked the fact that it wasn't tied to a web interface. It's just an SDK with which you can build your own e-commerce desktop applications, CGI scripts or server modules. It's a very flexible toolkit and was exactly what we were looking for.

Merchant Account

The second component we needed was a Merchant Account that supported the transaction service we chose. In order to process credit card transactions, you need to have a merchant account that acts as an intermediary between your bank account and your customer's credit card company. Getting a merchant account involves a large amount of paperwork, a credit check and a setup fee.

Other than finding a merchant account that supported Cybercash, we didn't see much difference in the offerings aside from cost. There's typically a setup fee and fixed transaction fees. Then the credit card companies take their cut. However, when all transaction fees are totaled, you're still usually under 4%. Compared to the fees typically charged by third-party e-commerce companies, the difference can add up to non-trivial sums. In our case, we earned back the money we spent setting up our Linux based e-commerce solution in a couple of months based on this percentage difference. Typically, on-line software stores for shareware will charge between 15% and 40%.

SSL Server

The next component we needed was an SSL (secure sockets layer) server that encrypts traffic to and from the web server. It increases your customer's confidence and improves the security of on-line transactions. We were comfortable with the Apache web server, so we wanted to find an SSL server based on Apache. We looked at a couple of vendors and ended up making our decision based on price. We chose the Raven SSL web server, and it has worked well for us. Their tech support has been very helpful. Today, a number of other options are available.

Certificate Authority

In order to set up an SSL server, you need a “certificate” from a third party known as a Certificate Authority. The SSL vendor will give you a temporary invalid key to use for testing purposes.

The certificate is designed to verify the identity of you and your company. It provides assurance to the customer that they are actually dealing with your company. Unfortunately, getting a certificate can be paperwork-intensive, as you must verify your identity to the Certificate Authority. This usually means giving them your incorporation paperwork. In our case, it took slightly over two weeks to go through the certificate process. The Certificate Authority then issues you a certificate key via mail. It's just an encrypted block of text that you cut and paste into your SSL server setup.

We ended up choosing Thawte for our certificates because they were less expensive. The only problem we've had has been with older browsers that no longer recognize the certificate authority. This generates some spurious errors. However, since fewer people are using the 3.*x* versions of Netscape and MS Internet Explorer each day, we don't see this as a major problem.

Database

We wanted to be able to do more with our e-commerce solution than just process fixed transactions. We wanted to have a system that could easily be extended as the need arose. Additionally, we wanted to keep track of all kinds of variables so we could answer a number of questions, such as:

- Is the purchase process easy enough?
- Where do customers hear about us?
- What versions sell better?
- Is our pricing effective?
- Are there any trends in our sales that might shed some light on our customers?

As a result, we needed a robust and flexible database back end to store and organize all of this data. We needed to balance the speed and scalability of the database back end with reliability and ease of programming. Additionally, we needed easy access to the data and the ability to alter the structure on the fly. Beyond that, we didn't want to spend much money.

We decided on MySQL (http://www.mysql.com/). It's extremely fast, multi-threaded, flexible and supports a large subset of the SQL standard. It's a very popular database for web applications, and a good Perl interface is available for it. In addition, the licensing is flexible, and in many cases you are allowed to use it at no cost. An active mailing list and a tremendous amount of information is available on their web site.

Unlike the SSL server or the merchant account, our choice of database ended up being a critical one, as it was one of the components that made a difference when it came time to go after new business ideas.

Perl

We talked about implementing a “commercial grade” e-commerce solution in C or C++. This implied a major development effort and a lot of work if we decided to modify it later.

Being afraid to lock ourselves into a solution we couldn't change easily, we opted to develop in Perl, which saved us a great deal of time at the expense of some runtime speed. We figured since we were running under Linux that the overhead added by using Perl would be negligible, and by the time it became an issue we would be making so much money we could buy a faster machine.

Another key advantage of developing in Perl is that the code is quickly and easily changed. On-line business changes so fast it's hard to keep up. Anything we could do to make it easier on ourselves was in our business interest.

Putting It All Together

Once we had all the pieces identified, it was fairly easy to put it all together. Just two pieces of code were left to write. The first managed the experience customers have when they come to our web site to place an order. The second was an administration interface that enabled us to do some maintenance tasks. The first version of our solution worked as follows:

- Display a form on the secure server that allows the user to enter their credit card information. It's linked to a CGI script that does the bulk of the work.
- In the Perl CGI script, verify the FORM fields and make sure all required fields are filled with values that make sense. If there is an error, let the user know, log the error in the database and send an e-mail notice to the administrator. We wanted to know how well users were doing with our order form. As a result of this tracking, we ended up improving the order process significantly.

- If no errors are found in the form, format a Cybercash transaction package with all the relevant fields and pass it to the Cybercash Perl module for submission to the Cybercash servers.
- Take a look at the response codes from Cybercash. Quite a few fields are returned, but the key ones are the transaction status and the AVS (address verification service) code.
- If the transaction was not authorized, log an error in the database, let the customer know and fire off an e-mail message to the administrator.
- If the transaction was authorized, check the code, which is a simple test designed to reduce fraudulent transactions. It compares the numerals in the address and zip code provided by the customer to the address and zip code present in the records at the credit card company. The reasoning is that if someone steals your credit card, it is unlikely they have your address as well. AVS codes include “Y” for everything matched correctly, “A” for only the address matched, “Z” for only the zip matched and “N” for nothing matched. Unfortunately, the AVS system is available for US addresses only.
- Send the customer a thank-you message by e-mail, log the transaction in the database for later perusal and send the administrator e-mail indicating that a transaction was authorized.

The administration interface allows us to enter orders by hand for phone orders, mail-in orders and fax-in orders. It also allows us to edit individual records and review all transactions and errors. We use this interface on a daily basis to manually examine each transaction for signs of fraud. (A complete discussion of fraud detection is outside the scope of this article.) We've been seeing a fraud rate of around 1-2%, which we are told is about industry average.

The way our system is set up, transactions get “authorized” but do not automatically get “settled” until we've had a chance to review them. It is possible to automate the entire process so that no human is involved, but we prefer to look over all transactions prior to letting them go through.

Operational Experience/Lessons Learned

Having our own e-commerce solution built on a reliable platform with flexible tools has meant we can change our system to experiment with various methods for improving our business.

First, we noticed through all of the error tracking we built into the system that quite a few customers would come to the order form, try to enter their information once, make some simple error such as not filling in a zip code and not return. We made some changes to our data entry form and reduced the number of fields down to the bare minimum. This seems to have improved things dramatically—there was a noticeable increase in sales.

One afternoon, we decided to add a “comments” box to the order form to obtain more feedback from our customers. Strangely enough, we've found that customers are much more likely to provide us with detailed feedback while they're ordering our product than they are at any other time. Because we chose MySQL and Perl to build the system, adding this option was literally 30 minutes of work and gave us a whole new opportunity to know our customers. It has proved surprisingly valuable.

Another lesson we've learned through experimentation is that making our product easy to buy makes an incredible difference in the quantity we sell. You can have a great product, but if it's not easy to license, few will buy it. Again, the flexibility of the tools we chose allowed us to try a number of different approaches until we found one that worked for us.

On October 20, 1998, we came out with the 3.0 version of our stock-tracking application. Based on quite a few different parameters, not the least of which were comments from customers, we decided to change our pricing and bring out two editions of the product: a standard and a gold. We also realized we could offer “upgrade” incentives on-line. We ended up taking our single product e-commerce solution and modifying it quickly to sell essentially four products: two editions and two upgrades, each at a different price point.

Another interesting side effect of running our own e-commerce solution and being able to modify it quickly for new situations was “affiliate transactions”. We were presented with a few opportunities to have other companies promote our software from their web sites and on their CDs. Obviously, the question arose of how to account for the sales. Again, the flexibility of the tools we had chosen allowed to us to respond to these new opportunities. Had we not chosen Linux and this set of tools, we would not have been in a position to go after this new business.

Conclusion

Choosing to build a Linux-based e-commerce solution has provided a critical benefit for our business and allowed us to effectively respond to constant change.

Yermo Lamers (yml@dtlink.com/) is co-founder of DTLink Software (http://www.dtlink.com/), a company that develops and distributes Personal Stock Monitor for MS Windows, an Internet-based stock tracking application. He has built and licensed quite a few revenue-generating server systems on top of

Linux since the days of SLS .99pl14 and looks forward to the day when Wine can be used to run Personal Stock Monitor on Linux.

Archive Index Issue Table of Contents

Advanced search

Personalizing “New” Labels

Reuven M. Lerner

Issue #63, July 1999

How to let the site visitor know which documents he hasn't seen.

Like many people, I spend a great deal of time on the Web. Some of that time is spent working—writing and debugging programs for various clients. I also spend quite a bit of time reading on the Web, keeping track of the latest news from the real world and the computer industry, and even exploring new sites that friends and colleagues have suggested I visit.

A common feature on web sites, one which never fails to annoy me, is the proliferation of graphics indicating which items are new. I don't mind the fact that the site's author is letting me know the most recently changed or added items. Rather, it bothers me to know that these tags indicate whether the document is new, rather than whether the document is new for *me*.

When I visit a site for the first time, all of the documents should have a “new” indication, since all are new to me. When I return to the site, only those added since my previous visit should have the “new” graphic, perhaps including those modified since my last visit. In other words, the site should keep track of my usage patterns, rather than force me to remember whether I have read a particular file.

This month, we will take a look at this problem. Not only will we see how to create a web site that fails to annoy me in this particular way, but we will also look at some of the trade-offs that often occur when trying to handle site maintenance, service to the end user and program maintainability.

The Simple Way

Now that I have disparaged the practice of putting “new” labels on a web site's links, let me demonstrate it, so we can have a clear starting point. Here is a simple page of HTML with two links on it, one with a “new” graphic next to it:

```
<HTML>
<Head><Title>Welcome to My Site</Title></Head>
<Body>
<H1>Welcome to My Site</H1>
<P>Read <a href="resume.html">my
resume.</a></P>
<P>Read <a href="deathvalley.html"><img src="new.gif">
about my recent trip to Death Valley!</a></P>
</Body>
</HTML>
```

When the page's author decides enough time has passed, the “new” logo will go down. These labels are updated by modifying the HTML file, inserting or erasing the graphics as necessary.

This technique has a number of advantages, the main one being that the site requires less horsepower to run. Downloading text and graphics does not require as much of the server's processor as a CGI program, which requires additional memory as well as processing time.

However, this technique also has many disadvantages. First of all, the labels change only when the webmaster decides to modify the HTML file, rather than on an automatic basis. Secondly, the labels fail to take users' individual histories into account, meaning first-time users will see the same “new” labels as daily visitors.

Auto-expiring the Labels

How can we approach this problem? Let's begin with a simple solution that does not use personalization, but does provide more accurate labels than the above approach. We can auto-expire the labels, printing “new” during the first week a file is made available and “modified” the second week. Files more than two weeks old will not have a label.

The easiest way to do this is via server-side includes. SSIs execute as if they were CGI programs, but their output is inserted inside an HTML file. SSIs are useful when you want dynamic or otherwise programmable text inside an HTML file, but don't have enough dynamic output to justify burying the HTML inside a CGI program.

In this particular case, we can take advantage of Apache's advanced server-side include functionality, which allows us to execute a CGI program and insert its output into an HTML file. For example, we can slightly modify our file like this:

```
<HTML>
<Head><Title>Welcome to My Site</Title></Head>
<Body>
<H1>Welcome to My Site</H1>
<P>Read <a href="resume.html">my resume.</a></P>
<P>Read <a href="deathvalley.html">
<!-#include
virtual="/cgi-bin/print-label.pl?deathvalley.html"
->
```

```
about my recent trip to Death Valley!</a></P>
</Body>
</HTML>
```

As you can see, the second link includes an SSI. One nice thing about SSIs is they look like HTML comments, so if you accidentally install an SSI-enabled file on a server that does not know how to parse them, the entire SSI will be ignored.

SSIs work thanks to a bit of magic: before the document is returned to the user's browser, it is interpreted by the server (hence the term "*server-side* includes"). Apache replaces all of the SSI commands with the result of their execution. This could mean printing something as simple as a file's modification date, but might be as complicated as inserting the results of a large database-access client invoked via CGI.

Listing 1. print-label.pl

In the above example, we run the CGI program print-label.pl, the code for which is in Listing 1. While this program is run via SSI rather than a pure CGI call, it works just like a CGI program. We use CGI.pm, the standard Perl module for writing CGI programs, to retrieve the *keywords* parameter, which is another way of describing a parameter passed via the GET method following the question mark.

Once we have checked to make sure the file exists, we use the **-M** operator to ask Perl to tell us the number of days which have passed since the file was last modified . If **$ctime** is equal to less than 7, the file was modified within the last seven days, meaning the file should be considered "new" for our purposes. We use a **font** tag to tell the user that the file is new.

If we use SSI with each link on our site, the "New!" message will appear for all links less than one week old.

I considered several ways of handling errors within print-label.pl, including using Perl's **die** function to exit prematurely and print an error message on the user's screen. In the end, I decided the program should exit silently if the file does not exist, or if no file name is specified at all. You may wish to send a message to the error log, which can be accomplished from within a CGI program by printing to STDERR as follows:

```
print STDERR "No such file \"$filename\"\n";
```

A major problem with this arrangement is that CGI programs are inherently resource hogs. If we have ten links on a page, using this technique involves running ten CGI programs—which means launching ten new Perl processes each time we view this page. For now, we will ignore the performance implications and focus on how to get things working. I will discuss performance toward the end of this article and in greater depth next month.

Bringing the User Into the Picture

The above technique is a good start, but it still ignores the user's perspective. That is, the links are expired on an absolute time scale. But a user who visits the site less than once per week will see too few “new” labels, while someone who visits it more often than once per week will see too many “new” labels.

How can we take care of that situation? One way is to keep track of when the user last visited our site, and make the comparison to that time stamp rather than to the file's creation or modification date. How can we know when the user last visited our site? Since HTTP, the protocol used to transport most Web documents, is “stateless”, each transaction takes place in a vacuum. When a web browser makes a request to a server, the request is not connected to any previous or following request. No information about previous requests is passed along, and nothing we do in our request is saved for later ones.

The best and easiest way is to use HTTP cookies, which nearly every browser supports. Cookies are variables set by the server and stored on the client's computer. Cookies allow us to track state across transactions by storing information on the user's computer. When the server next encounters the user, it can compare the time stamp on the cookie with the time stamp on the file.

Thus, we can rewrite the above program so that it auto-expires the labels based on when the user last visited the site. Each time the user visits our site, we set a cookie. The cookie's expiration date is set to be one week in the future, meaning that if the cookie exists, this user visited our site within the last week. Our labeling program (Listing 2, print-label.pl) then has a simple way to determine whether it should print “new” next to a link—the label should be printed only if the cookie does not exist.

Listing 2. print-label.pl with Cookie Check

Because we are using CGI.pm, which includes all necessary functionality for writing CGI programs, we can check whether the cookie exists in this way:

```
my $visited_recently = $query->cookie('RecentVisitor');
```

We can then print the label with the following code:

```
if (!$visited_recently)
{
        print "<font color=\"red\">(New!)</font>\n";
    }
```

Setting the Cookie

That about does it for reading the cookie. But how do we write the cookie? This is a stickier problem, one which has a number of potential solutions. The cookie specification requires that an expiration date be written with a full UNIX-style time and date stamp, as in

```
Thu Apr  8 02:25:30 IDT 1999
```

We cannot simply create and send a cookie with an expiration of “one week in the future”. We also have to figure out a way to set the cookie from within our HTML file—unless we want to use a CGI program to send the text, which would defeat the purpose of using SSIs to begin with.

One solution, although admittedly not the most elegant or efficient one, is to take advantage of the **META** tag supported by standard HTML. **META** tags have a number of uses, among them the ability to send data that would otherwise be sent in an HTTP header.

Since HTTP cookies are sent as part of the header in the browser's HTTP request, it's possible to set the “RecentVisitor” with the following HTML at the top of our page, within the **<Head>** section:

```
<META HTTP-EQUIV="Set-Cookie"
CONTENT="RecentVisitor=1;expires=Thu Apr
15 02:19:17 1999; path=/">
```

This tells the browser it should pretend a **Set-Cookie** HTTP header was sent from the server, and the **content** attribute should be handled as if it were the header's value. That is, the above **META** tag sets the **RecentVisitor** cookie to **1** and allows the cookie to be anywhere in my domain. The cookie is set to expire on April 15, 1999.

Creating this **META** tag is a bit difficult, since the date depends on when the user loads the page. If the user loads the page on April 8, the cookie should be set to expire on April 15. If the user loads the page on April 10, the cookie should expire on April 17. We need to modify the output according to when the user visits.

Listing 3. send-cookie.pl

The fact that the cookie's expiration date must change with time means we need to insert a program somewhere. The easiest way to do this is with another program invoked via SSI, which will create the **META** tag for us. Such a program, send-cookie.pl, is shown in Listing 3. With that installed and in place, we can say

```
<!-#include virtual="/cgi-bin/send-cookie.pl" ->
```

Our program, send-cookie.pl, sets the cookie's value by creating a **META** tag based on when the user accesses it. With this in place, each visit to our site will produce a cookie that disappears (or “crumbles”, if you prefer) within one week. Our SSI checks to see whether that cookie was sent, and if it was, prints an appropriate “new” label.

Problems with this Approach

The above approach has two major problems, one having to do with the user interface and the second with performance.

Let's address the user interface issue first. In short, what happens if the user reloads the page? The first time he viewed the page, the cookie was set with the **META** tag, regardless of whether the cookie had been set before. The next time the user loads the page, even if it is just a few seconds later, the “new” labels no longer appear, because the cookie has been set, indicating the user visited the site within the last week. We need a finer-grained method for keeping track of these labels.

The second is a more serious problem—the performance hit. In order to implement this solution, we need to invoke at least two CGI programs for each document on our system. Given how resource-hungry even the most innocent CGI programs can be, particularly when written in Perl, this adds a tremendous load to the web server. Add to this the time it takes to start up a Perl process and execute such an external program, and our users will suffer as well, unless we make a significant investment in hardware.

We can solve the user interface problem with the **Text::Template** module, written by Mark-Jason Dominus and recently re-released as version 1.20. This module, as is the case for most modules, is available from CPAN (see Resources) or by using the CPAN module that comes with modern installations of Perl.

Text::Template allows us to mix Perl and HTML within a file. Everything within curly braces, {}, is considered to be a Perl program. The results of the block's evaluation are inserted into the document in place of its code block. Thus, if we say

```
<P>This is a first paragraph.</P>
<P>{ 2 + 5; }</P>
<P>This is a second paragraph.</P>
```

the end user will see

```
This is a first paragraph.
7
This is a second paragraph.
```

on his or her screen.

Remember, the result of evaluating a block is not the output from that block, but rather the return result from the final line in the block. So if we say:

```
<P>This is a first paragraph.</P>
<P>{ print 2 + 5;}</P>
<P>This is a second paragraph.</P>
```

we will see

```
7 This is a first paragraph.
1
This is a second paragraph.
```

The “7” comes from evaluating “print”, while the “1” is the returned value from the final line of the embedded Perl block.

Listing 4. template.pl

In order to use Text::Template, we will need to write a small CGI program that invokes the module and parses the indicated file. The program template.pl, shown in Listing 4, does the trick simply and easily. If we install it in our CGI directory, we can then go to /cgi-bin/template.pl?file.tmpl, and the template file.tmpl will be interpreted by template.pl, then returned to the user's browser.

In order to deal with potential security problems from people specifying unusual file names, we remove any occurrences of the string “../” and ensure all file names start in the directory /usr/local/apache/share/templates/. You may want to define a different templates directory on your system.

Now that we have our templating system in place, we can rewrite our template cookie, in which contents and “new” labels are printed only when necessary. The final result is shown in Listing 5.

Listing 5. travel.html

We create the dynamic **META** tag with the following code:

```
<META HTTP-EQUIV="Set-Cookie"
CONTENT="RecentVisitor=1;
expires={scalar localtime(time + 604800}; path=/">
```

As you can see, this **META** tag contains a small Perl block that returns an appropriate expiration date. The date is set to be 604,800 seconds in the future, better known as “one week from today”.

We retrieve the cookie later in the template, just before deciding whether to print a “new” tag:

```
use CGI;
my $query = new CGI;
my $visited_recently =
$query->cookie('RecentVisitor');
$outputstring .= "<font color=\"red\">(New!)</font>\n"
unless $visited_recently;
$outputstring;
```

Notice how we can import the CGI module within a block of the template. We can then create an instance of CGI and use it to retrieve one or more cookies. We don't use CGI.pm to print output to the user's browser, since that will be done by the templating system.

Next Month

It would seem that my obsession with “new” labels has led us in all sorts of new and interesting directions. This month, we looked at cookies, server-side includes, CGI programs and HTML/Perl templates. While templates did reduce the load on the server somewhat, they still require the invocation of a CGI program, which is inherently more costly than serving a straight HTML file.

One solution to this problem is to make the labeling an inherent part of our web server. If the server could keep track of the cookies and the labeling, things would work fine. Most people don't want to mess around with their web server software to that degree; Apache might be free software that allows you to mess around with the source, but few of us are that daring.

However, as we have seen in previous installments of *ATF*, we can easily modify parts of the server with Perl, rather than with C, by installing the **mod_perl** module. While such a system still requires some code for each retrieved document, the overhead for running a Perl subroutine to Apache via mod_perl is much lower than that required for an external CGI program.

Next month, we will examine a mod_perl module that goes through the links on a page and adds a “new” label for each item new to the user accessing the site. When we're done, we will have made the web a bit better and easier for pedants like me and for users who should not have to remember when they last visited a site.

Resources

All listings referred to in this article are available by anonymous download in the file ftp.linuxjournal.com/pub/lj/listings/issue63/3473.tgz.

Reuven M. Lerner (reuven@lerner.co.il) 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. The ATF home page, including archives and discussion forums, is at http://www.lerner.co.il/atf/

Guest Editorial

Eric Hughes

Issue #63, July 1999

The average Joe wants something for nothing, and Mr. Hughes wants to give it to him.

I paid for my university education, in that I paid an institution for access to its faculty and for use of its facilities. The knowledge was free; all I had to do was take advantage of the fact that it was there for the learning. I've told many people about things I learned in school. I didn't need to pay a license fee to tell other people about Hamiltonian mechanics or Gödel incompleteness. The academics who created this body of knowledge published papers and gave away their knowledge. As far as I'm concerned, I got something for nothing from them.

Michael Faraday discovered electromagnetic induction and thereby paved the way for the electricity and electronics industry. The value he captured from this development was nowhere near all the value he created for everyone else. Such was his station in life. He made the choice to freely give his time to the advancement of science. Many others have followed his example, but nowhere near a majority. Yet this tiny fraction of people has exerted a significant, disproportional change on the world. Such is the fate of certain knowledge workers. Get used to it.

On first blush, software appears to follow this academic pathway. Yet software has users, who must find the software useful to keep it. Researchers do not have to make their output useful, just accurate, novel and potentially useful. To wit, researchers don't have to do market research. Clearly, software developers don't think they have to understand their users. Developers need to understand their users only when they expect to have users.

The distribution of software clearly follows the model of academic knowledge—create it once for free, then make the user pay for its distribution. Yet the creation of software does not mirror the creation of knowledge quite so accurately. The principle of “academic freedom” is about as antithetical to the principle of “attaining user benefit” as possible. The two just don't fit and can't. The university never had a monopoly on creating knowledge—or on developing free software—and it never will. The university, however, does embody the understanding that some development requires nurturing that cannot be easily obtained elsewhere. Software development has no such analogous institutions, and it needs one that is not the university.

This new institution should be a non-profit, tax-exempt corporation especially created to pay for the design, construction and delivery of public software. Let's be obvious about this: giving away software for free is about as close as possible to the center of the charitable purpose requirement of a tax-exempt company. Like the university, the staff—no, let us call them the talent—the talent working at this institution are not motivated by dreams of entrepreneurial riches, but by the advancement of the craft. The talent needs to come from a wide variety of disciplines, essentially all that are present at a for-profit software company. This new institution will always be in competition with commercial interests for talent, so it will have to pay for talent accordingly. Talent gets paid, the institution creates some well-needed feeling of solidity, and regular folks get free software.

“Public software” is any software that people can obtain and use without any entanglement with intellectual property issues. Public software also denotes the ubiquity of the software and the corresponding expectation that people encounter it frequently. Some open-source software is public software, some is not. Public software is a concept rooted in the nature of its end use, not in the means of creating it. All the wrangling over license terms for derivative works has obscured the obvious point that the license is in service of some goal; if you don't name your goal, you can't possibly attain it. I know what my goal is—it is free beer.

I still can't figure out how the claim that the GNU Public License encourages free speech is not utterly disingenuous. The GPL is the opposite of free speech; it's a highly detailed copyright agreement with the purpose of restricting the expression of derivative works. If I can't keep an expression to myself, I am restricted. All license agreements begin from the starting point of complete restriction, that is, total prohibition against use, and then work forward from that point. The summit of free speech is public domain expression—if you want to speak it again, go ahead, and for whatever purpose you care to seek. As much as I am an advocate of free speech and all other civil rights, my purpose with public software is not free speech—it's free beer.

The crucial reason the GPL has achieved such limited success in scope is that its purpose is to benefit programmers who want access to code, not to benefit outside customers. Whatever benefit an outside customer gains is ancillary to the benefit to programmers. This observation also explains why most GPL code is in development tools and environments. To be blunt, the GPL is a selfish contract for selfish purposes which cannot possibly be generalized.

The open-source model, by contrast, is a technique in search of goals. Open Source encompasses the goals of the GPL, and indeed found some of its initial inspiration there. Yet Open Source also encompasses other disparate goals, such as those for Mozilla. I want to promote Open Source, but as part of my desire to promote public software. My goal is compatible with Open Source, but I seek Open Source not to promulgate its own merits, but as an enabler of public software. Indeed, I fear sometimes that the Open Source movement may fall to the hazard that is the core selfishness of the GPL. In an attempt to seek openness, its promoters may forget that, whatever benefit they derive from access to code, they must place benefits to users first. By avoiding that hazard, I am confident good results will come from the hubbub of activity surrounding Open Source. I wish them well, but their effort is not identical to mine. My effort is toward public software.

The natural venue for the release of public software is the new institution. The value of any software derives in large part from the solidity and gravity of the organization that creates it. Ferreting out those expectations about the future which affect the net present use value of software is another essay. Let me observe, without justification, that people overwhelmingly prefer code that is stable, architecture that is well-designed, products that can be repaired and upgraded, and companies that will endure. Most people lack the means to evaluate technical merit in software. Even more people prefer to do things other than evaluate software. For almost everybody, one's expectations about the institution stand in as a proxy for all that thinking. Since I want public software to become ubiquitous, I want the new institution to ship product.

I have not yet mentioned sustainability of the new institution, because concerns about sustainability are concerns about means, not about goals. A coherent and organizing focus on what needs to be accomplished has been lacking; instead, we have had endless fretting about how to do it—whatever this “it” is. For all of its faults, at least Richard Stallman elucidated a clear goal for the GPL: “Make all source code everywhere usable by me personally.” A selfish goal, but at least it roused the mutually self-interested to action. I cannot name another general goal that has inspired more coherent effort in this area. There has been much agreement about techniques, but I have seen no successor—in goals—to the free software manifesto and the GPL.

Nor do I want a single successor. I want to see many inspirations for creative technical work. Some may overlap; some may conflict—this is of no matter. I want to see manifestos that exult in the clarity of their vision. I want to see new approaches conceived in the understanding of the obstacles to victory. Here, for example, is one such possible goal for Linux:

> I want Linux to be the only conceivable choice for every commercial and personal use of operating systems. I want universal device support, instant installation, zero administration and a completely correct implementation.

We will know we have succeeded when Microsoft's market capitalization suddenly drops to exactly its cash balance.

No institution can survive on a single goal, for when that goal is accomplished, the purpose of the institution fades. The life spring of an institution, that whence all material support arises, is the stream of specific purposes that passes through it. Universities achieve this with tenure and academic freedom. The new institution, for its existence and its continuation, requires this same kind of stream of purposes. I want software for nothing; thus I require cooperation with others.

Given my goal of just-use-it-ware, I propose the means of a new institution. As a means to a means, we now have to examine the grungy underbelly of institutional sustainability. Or, to wit, who pays? If you grant the desirability of the goal (no-fee software) and the usefulness of the means (the new institution), then we need not become spooked when we discover sustainability is hard. Once we know our goal, we may persevere in seeking it and not be distracted by less attractive or undesirable goals.

The first, almost too-obvious place to look for resources is the existing network of public and government money that funds such non-profit endeavors as public broadcasting and particle accelerators. No one can do this alone. Who would give a few million dollars to a hastily organized bunch of technical guys who look suspiciously like the “Hacker Development Environment League”? If the point of the institution is software that benefits the public directly as a whole (and not indirectly, with compilers), then we need representatives on the board who are believable and legitimate to the various constituencies who might provide funding. That means people outside the computer industry.

As a rule of thumb, the university takes one-third of each grant to support the institution. This is simply the necessary overhead to have an institution and not just a group of researchers. Rather than worrying about the shibboleth of “efficiency” of the grant allocation, accept that the very existence of the institution is of separate efficacy. The institution creates value—value in solidity, value in stability, value in longevity—that individuals and informal groups can never provide. The total value of software is far more than the actual running code. Software of uncertain provenance and indeterminate future is mostly worthless. If you disbelieve this, go look at market share estimates. Put an institution behind that same code and it suddenly becomes valuable. To be generous, maybe one-quarter of the total value of software comes from the product. We can get the other three-quarters value from an institution and pay for the institution with only half the money we spend directly on talent. Still worried about efficiency? Three times the value for half the money, and institutional maintenance is looking six times more productive than the talent.

It's not that the developers are unproductive. The mythos of the hacker community rests in the power of a small number of programmers to change the world. When put this way, the effort is discrete and the change is instantaneous. In other words, the formulation is one of magic, not of economics. Extensive change comes only from sustained effort by numerous people with aligned goals. Unless there are people who nurture the project without interruption, these efforts at change wither and become of no consequence. Those who have set their lives around the mythos of the magical coder urgently need assistance in completing their work. I believe in this mythos. I do not identify it as magical in order to kill it, but rather to feed it. The new institution I advocate herein is a completion of the creative spark at the heart of all good software.

The activity here is not merely inventing such mechanisms and analyzing them, but also mounting experiments. The institution needs fiscal solidity in order to have the confidence to attempt new forms of support.

A first, concrete funding goal would be to build an endowment fund. Rather than endowing a chair in which a single researcher sits, I want to endow a table around which a release committee shall meet. Here's a specific beginning goal for such a new institution: a $25 million endowment for the Linux Release Table. The residual investment income would be adequate for five members of the table, two paid interns and two administrative staff members, all full-time positions. This table would not do any technical work, but would coordinate planning, architecture, development and release. Now this table could not possibly encompass the breadth of activity generated from a ubiquitous, dominant Linux; clearly, more people and more money and more structures would be necessary. Yet this first endowment could be the seed of a full set of institutional structures surrounding Linux.

The focus on endowed tables for release is one of the lessons learned from the open-source world and from Java. Namely, *who* matters more than *how*--the party who releases new versions of a product matters more than how the license terms read. The new institution needs to focus strategically on branding and compatibility as keystones to generating value for users. No matter what the licenses for intellectual property are, the association of the institution with the product is the critical element in delivering the value of the institution. Branding is the manifestation of this association and is the initial point of its delivery. Compatibility prevents pollution of the brand and thereby ensures its longevity. Trademark licensing enables control of the brand and subsequent control over compatibility. Sun has masterfully demonstrated with Java how to pull off this trick. How much better if one of the new institutions had pulled it off instead!

I have suggested a new institution; I also suggest a new idea for an institution. A mature field of public software creation could not subsist on a single organization of this new type. No initial efforts in creating these new institutions should be taken as an excuse to defer one's own effort away from building a “competing” institution. The principle of the new institution is public and cooperative, not singular and nepotistic. I should hope for jockeying for position between institutions, only as a convivial process of mutual betterment.

The average Joe wants something for nothing. With knowledge and information, we can come as close as possible to this ideal. Let the scarcity economists haggle over flesh. We won't appreciably change GDP figures. The new institution is an exercise in abundance economics. Free knowledge and information add untold real wealth to the world. Let our revenge upon scarcity be that its limitation upon wealth become miniscule.

Eric Hughes (eric@sac.net) was one of the founders of cypherpunks, and has been worried about software infrastructure ever since. He is Chief Technology Officer of Signet Assurance Company, LLC, a development company soon to announce its first products and services.

Letters to the Editor

Various

Issue #63, July 1999

Readers sound off.

Red Hat Paranoia

I have seen several letters regarding fear of Red Hat taking over the Linux market. I feel compelled to point out a few important points regarding Red Hat:

One, Red Hat Linux is GPL, which means they cannot squeeze out *any* distribution of Linux by introducing proprietary system components. An excellent example is the RPM system; anyone can incorporate a Red Hat package manager into their system, and this is encouraged by Red Hat. Intel's investment in Red Hat will be beneficial to all Linux users and distributions.

Two, Red Hat is not out to crush competition. Red Hat software has been used as the basis for new distributions such as Linux-PRO. Anyone can freely download, use, modify, sell and incorporate their work.

Three, Red Hat embraces new technologies, such as glibc, before other commercial distributions. This may make certain programs which use these technologies incompatible with other distributions, but this isn't the fault of Red Hat. Other distributions are just slower to adopt these non-proprietary technologies.

Four, Red Hat isn't significantly harder to configure than other Linux distributions. Everything is hard until you know how to do it.

Five, no one forces anyone to use Red Hat. No one even tries to do so.

I have had experience with Red Hat, SuSE, Caldera and Slackware; I see strengths and weaknesses in each. Red Hat is not my favorite distribution. However, it bothers me to see people criticizing Red Hat and comparing them to Microsoft when to me most of these criticisms are unwarranted.

—Michael O'Brien mobrien@unm.edu

gawk 3.1

I read Juergen Kahrs' excellent article in the April 1999 issue, “Network Administration with AWK”, on using **gawk** to access live net connections as files. Now I am watching for the announcement of gawk 3.1, which has the network features.

This article advertises gawk's net functions as a Swiss Army knife for network administration, and one can easily see how it would help with network monitoring, housekeeping, security and optimization. Also hinted at in the article and dealt with in more detail at home.t-online.de/home/Juergen.Kahrs are many other selling points for gawk's net functions, such as learning and prototyping artificial intelligence, agents and web server functions.

It would be ideal if gawk was using callbacks internally rather than looping to wait for activity. Is this what is meant by the term “non-blocking read”? I would think wrappers of other languages or mere shell functions might work for gawk 3.1's missing features of broadcasting, non-blocking read, timeout and forking server processes. Thanks.

—Bob recbo@erols.com

DOSEMU and Wine

Although I am a new user of Linux, I have noticed a tendency to either make applications that communicate well with Windows or use emulators to run DOS-native programs. This seems to contradict the philosophy behind Linux, to create a free operating system. With the superior file system of Linux, why would people want to run DOS or Windows programs? Some program managers have gone so far as to use a Windows 9*x*-style toolbar. I think programmers should be more original in program design.

—Matthew Stapleton bifflinux@juno.com

At present, this is a Windows world, so good communication between Linux and Windows is a must. Many Windows and DOS applications are not available for Linux, so fans of those programs want emulators such as DOSEMU and Wine. Those same people want their desktop to look familiar, so developers tend to provide managers with which the user will be comfortable rather than developing a completely new design —Editor

Music and Linux

Great magazine! I'm a newcomer to Linux and it was quite amazing to browse the magazine stand and find a non-Microsoft/Apple-based magazine. I really enjoyed the interview with John Ousterhout and the reviews. If I might make one request, though: in all the documentation I've found, there's little to no mention of the sound manipulation capabilities of Linux systems. I'm a musician and have used quite a few good programs on the Windows platform, but I can't seem to find anything similar on Linux. If you could print an article on this subject, it would be greatly appreciated by at least one person.

—Matt Nelson mnelson@nelcomail.com

A very good article on Linux and sound can be found in our February 1999 issue: “Csound for Linux” by David Phillips. Also, back in September 1998, we had a review of a music publisher called MUP by Bob van der Poel —Editor

Correction to grep Article

In the syntax section of the April article “grep: Searching for Words”, Jan Rooijackers mistakes shell expansion with regular expressions. The grep command **grep flip *** does indeed use the regular expression “flip”, but it never sees the *, as the shell has expanded that asterisk to be each file in the directory. Mr. Rooijackers incorrectly explains that grep is using the asterisk as a regular expression meaning all the files in the directory.

—Michael Jones mjone4@amfam.com

Minor Correction

Linux Journal is amazing, and issue #60 was no exception. However, I'd like to point out a minor typing mistake in the article “Linux 2.2 and the Frame-Buffer Console”; the text says the frame buffer device files are /dev/fd*. Shouldn't that be /dev/fb*?

—Celso Kopp Webber webber@sj.univali.rct-sc.br

Yes, you found a typo we missed. Thanks for pointing it out —Editor

URL in May 99

I received my free issue of *Linux Journal* today and began reading the article “Creat: An Embedded Systems Project” by Nick Bailey. It contains the URL http://ryeham.ee.ryerson.ca/uCinux/ for the Linux/Microcontroller home page. I tried to access this site, with no luck. Can you tell me the correct URL?

—Rich keske@lexmark.com

Pesky typos! The correct URL is http://ryeham.ee.ryerson.ca/uClinux/ —Editor

More Letters to the Editor

awk Article

Thanks for the recent articles on **awk** (“The awk Utility”, Louis Iacona, June 1999) and **sed** (“Good Ol' sed”, Hans de Vreught, April 1999). I think these useful tools have been unfairly eclipsed by Perl, and deserve more attention than they have received recently.

awk's powerful library on string-editing functions makes it ideal for reformatting the output of one program to be read as input by another. It can do things that are difficult with sed—not least because of sed's prohibition of comments, which reinforces its reputation as a write-only language. Still, sed is much faster than awk, so I prefer to use sed scripts for quick editing (e.g., converting TeX to HTML).

Though the awk article emphasized its use as a file processor, there's no need to process any input at all. I often write awk scripts with just a BEGIN section, ending with an exit statement. These take the place of all those one-page, throwaway programs I used to write in FORTRAN. It's very handy for tabulating a function or checking an algorithm.

—A.T.Young, aty@mintaka.sdsu.edu

Attention all Linux users

In reading a recent article on slashdot.org, something struck me—a strategy for increasing commercial support for Linux. If every Linux user made a list of commercial applications he or she wished was available for Linux, then mailed the companies behind those products, promising to buy their software if a Linux version was released, I believe companies would take notice.

How many of us must switch to some form of Windows 9*x*/NT to use a program not available for Linux? I do it almost everyday. For my parents, it's Quicken; for me, it's StarCraft. I'm fairly sure I'm not the only person who would be willing to buy a piece of software a second time to be able to run it under Linux.

What I believe keeps companies from developing for Linux is that we (Linux users) sometimes get stereotyped as anarchists who want to destroy corporate America and who believe anything not released under the GPL is bad. Please, I truly feel if we all let companies know we are willing to pay for what we want, they will be more than willing to sell it to us. Write letters and e-mail messages, sign petitions—whatever you can to let people know. Make your wish list of applications to be ported to Linux, then help make it a reality. Well, I'm off to start writing letters.

—Loren Weber, cobweber@jps.net

> *SSC maintains an on-line Wish List at http://www.linuxresources.com/wish/. Other sites do too.<\br>—Editor*

Hello and Thanks

I just spoke with Matthew Cunningham at the *LJ* booth on the Comdex floor here in Chicago. *LJ* has sponsored a very nice exhibit. Anyway, I'm a long-time subscriber to *LJ*, (I don't quite go back to Issue 1, but almost.)

I very much enjoyed the article “Virtual Network Computing” by Brian Harvey in issue 58 (February 199). In fact, I was just about to purchase 400 seats of PC Anywhere for my company when I read the article. Instead, I installed VNC and saved the company a great deal of money. Mostly we use Windows 95 and NT at work, but we do use Linux for for firewalls and servers.

Keep up the great work, I read my issue faithfully cover to cover every month!

—Dave Truckenmiller, dtrucken@zsassociates.com

(article) Grep: Searching for words

> From page 10, lj issue 60
> `grep flip article.txt`
> This will search for the word “flip” in the file article.txt and will display all the lines containing the word “flip”

That's wrong. grep does not know what a word is. Or you instruct it to really search for a word, or you are searching for lines containing the 4 consecutive/juxtapose letters “f” “l” “i” “p”.

In fact, the command above would match “flippant” “flipflip” etc.

—Nagib Hobeica, nag@beryte.com

---

linux logo for computers

I like the *Linux Journal* very much. Recently I was fiddling to get the Linux penguin with “Linux inside” beneath it into a nice sticker that fits in the square on the front of regular PC enclosures. 'Linux' in green with an italic red 'i'. It came to my mind that it would be nice if you would print such a sticker in *each* issue of the *LJ* magazine were readers can cut it out and paste it on their machines. I believe that in a very short period we will see that to happen.

What do you think of it?
Kind regards

—Kees Schoenmakers

Code Error

First of all I would like to congratulate you and your team on the good job you do. I've been an avid reader of *LJ* for quite some time.

I would like to point out that the code for the multithreaded in issue 61 has a bug: the pointer tids isn't malloced correctly/at all—this may very easily lead to a segmentation fault. Anyway, this article clearly addressed people with good programming knowledge, so everybody should be able to spot the bug and correct it.

Regards,

—Andreas Zielke, azielke@TechFak.Uni-Bielefeld.DE

Thanks and e-address change

Thank you for publishing my opinions in Letters to the Editor. Unfortunately I have changed my e-mail address, so won't receive any feedback. Pacific Bell didn't support Linux and I surely didn't like to have to bring down my system to boot Microsoft, so changed to ISPchannel and am happy now to have reasonable speed.

Regards,

—David Baker, debaker@ispchannel.com

Re: resource location

> From: “Berg, Rick (GEAE)”, rick.berg@ae.ge.com
> In the article A Toolbox for the X User, there is a resource reference for a FTP site called ftp.ienet.ie/pub/X-contrib/applications, I can not see it and nslookup cannot find ftp.ienet.ie or ienet.ie. Is the reference correct? If so is there other locations where I could find the source for xtar and mgdiff?

Thanks for your interest. Since I have written the article over half a year ago, the mentioned site seems to have vanished.
Both sourcecodes are available from ftp.x.org:

ftp.x.org: contrib/utilities/xtar-1.4.tar.gz ftp.x.org: contrib/applications/mgdiff.tar.gz

—Christoph Dalitz, dalitz@infotech.de

Red Hat

I have been using Linux for about 5 years now and after attempting to install Red Hat 5.0, 5.1 and now 5.2 I am forced to say that this product is going to be the death of linux. It is difficult to install, even more difficult to configure and the email help from this organization is nonexistant. They rarely answer questions and when they do it is usually wrong. Slackware has a 32 page installation guide that I can fit in my shirt pocket and with it I can install and configure linux completely. O'Reilly's 'Running Linux' is a great book for the new user of linux, unfortunatly a great deal of the information on how to configure your desktop and X does not apply to Redhats cryptic configuration files. I have been trying to get my boss to try Linux for over a year now, he is no neophyte to computers, he has been a software engineer for over 20 years. He could not get ppp to dial, I told him to check to see if it was installed with 'pppd --version', when nothing happened he decided to ask Redhat why it wasn't installed. Redhats answer was as follows 'Tried to hack into the internet lastnite, almost made it'. Any company that wants to compete with a larger company like Microsoft will have to put up with a lot of apparently stupid questions, and answer them, quickly and intelligently.. for the good of the OS, even if they are just canned answers. sign me .. never again Red Hat.

Bob, daunais@metro2000.net

Signal to Noise

Thanks for what has been a fantastic magazine. Until very recently I've been very impressed with the quality of the articles in *LJ* and even the Q&A section. This month's Q&A section is exactly what I feared would begin to occur with the groundswell Linux is experiencing.

There are alot of questions being posted that people could have solved on their own had they JUST RTFM'd, it's all there the man pages explain everything in detail! I'm actually REALLY scared that out favorite OS is being dumbed down by the influx of NT Admins who like the pretty pictures and are not willing to learn anything, they want everything spooned into their gaping mouths.

NT has a quality: (ease of install) which allows these people to get in way too deep, it requires no knowledge of networks, security or otherwise, and I'm scared, REALLY scared for my favorite OS and the corporate pressures that it is beginning to suffer.

Anyway, please please at least tell these people (when answering their absurd questions) that they could've gotten to the info by typing “man foo”.

Let's keep signal to noise high...PLEASE??

—Eric Sobalvarro, eric@razorjack.com

BTS May1999 FTP Stuck

In your May 1999 issue of the *Linux Journal* Steve Mitchell wrote to ask for help with an FTP problem. He was getting the message “connection is closed by the remote host” when he attempted to initiate an ftp session with a machine running Redhat. While Chad Robinson listed two possible explanation/solutions to the problem, he missed what might be the most likely one.

Ftp and other services on Redhat are wrapped by tcpd to enable fine grained control of access to these services (via /etc/{hosts.allow,hosts.conf}) and detailed logging of requests for these services (in /var/log/secure). While Chad mentioned checking to see if the line pertaining to ftpd in /etc/inetd.conf was commented out (recommending that Steve uncomment it and send inetd a SIGHUP) he did not advise Steve to check hosts.allow and hosts.deny to verify that tcpd would let him connect to that service on the Redhat machine.

A faithful *Linux Journal* reader,

—Darryl Allen, dallen@value.net

---

i18n and charsets

Date: Tue, 11 May 1999 16:13:05 +0400
To: turnbull@sk.tsukuba.ac.jp
I would to express my appreciation for your article in *Linux Journal*, March 1999 : http://www.ssc.com/lj/issue59/3286.html

About two years I have to studying the same subject. If you can read Russian, please visit my page “Locale AS IT IS” (about POSIX locales) : http://www.sensi.org/~alec/locale I hope also, my collection of links could be useful for you : http://www.sensi.org/~alec/locale/locale_i.html

As you correctly state in your article, there are several encoding of russian (and other cyrillic/slavic) characters. You can also visit the page of Roman Czyborra http://www.czyborra.com (english) to find detailed information about it. See “Cyrillic Charset Soup” : http://czyborra.com/charsets/cyrillic.html

The same author explains the meaning of terms CES (Character Encoding System) and CCS (Coded Character Set): http://czyborra.com/utf/ More formal explanation there is on UNICODE site : http://www.unicode.org/unicode/reports/tr17/

Unfortunatelly, in “pure” POSIX enviroenmant we **completlly** loose CES, both for “flat” file and for serial terminal (or for /dev/vty emulatior). Also we have no way to get charset name for current locale. We should use XPG's nl_langinfo(CODESET) function, not included in POSIX standart.

Now, about mounting... The new Linux kernels (>2.0.35) support NLS which allows you to have two different encoding : one for “storing” filenames on disk, and another for “transferring” to kernel level. For example, for VFAT filesystem with russian filenames one can use :

```
  $ mount -t vfat -o noexec,codepage=866,iocharset=koi8-r /dev/hdb1 /mnt
```

Since Codepage 932 means ShiftJIS, this function could be useful for you. And now, about charsets. Regret, at present moment the standart for “Charset Name” does not exist. We can use :

```
  ISO8859-1
  ISO_8859-1
  ISO-8859-1
```

Many year ago IBM and Microsoft introduced the term “Codepage” which now practically means “Charset”. We can also use several names for there codepages :

```
  CP866
  IBM866
  e.t.c
```

For Windows we can use CP1251 which in Internet Explorer and MIME is called by different name “Windows-1251”...
Some time ago I had a discussion with Ulrich Drepper. In glibc2 library he uses charset name “mangling” method. He changes all to lowercase and removes all “-”, “_”, “.” e.t.c. This works for ISO8859-1 and ISO-8859-1, but complitly does not work for CP1251 and Windows-1251.

The incorrect setting of charset name (by LANG={}) breaks translation Keyboard Event --> Keysym --> Character string for XKB and Xlib XLookupString() and XmbLookupString() function.

Unfortunatelly, Ulrich in glibc2 and X-Free authors does not support charset name aliasing method, introduced by IANA : http://www.isi.edu/in-notes/iana/assignments/character-sets as “Alias:” field.

I thank you once again for your very interesting and informative article !

P.S. And finally I would like to ask you to change the addresses indicated in you article: http://turnbull.sk.tsukuba.ac.jp/Tools/I18N/LJ-I18N.html into links.

with best regards
—Alexander Voropay, a.voropay@globalone.ru

“Linux Threatens More Than Microsoft” - reply

Mark Matthews writes (letters May 99, “Linux Threatens More Than Microsoft”) that “what everyone seems to be overlooking is the threat it [Linux] poses to other UNIX systems ... Sun and SCO should watch out”. What, I would like to ask, is so good about that? Yes, I value free software, and yes, I value Linux, but what I really value, when I get into work every morning, is:

- having a proper make utility to run my compilations with, rather than an infexible IDE;
- having a powerful and portable scripting language, in the form of the Bourne shell, rather than the cmd.exe or command.com batch language;
- having a powerful interactive shell and associated utilities such as find and grep with which to interact with with my computer, rather than NT Explorer and its inconvenient and inflexible find utility;
- having the Unix file system, with everything under / and a standard set of directories such as /usr, /home, /opt, /dev, and so on, rather than “drive letters”, any number of non-standard directories under C:\, “COM ports”, shortcuts, etc;
- being able to telnet across the lab, or the world, rather than ... um?

and many other advantages too numerous to mention.
Clearly I am preaching to the converted in *Linux Journal*. My point, however, and what I feel that Mark Matthews misses, is that these are the benfits of all Unix systems, be they Sun, HP, SCO or Linux. In-fighting within the Unix community has only ever helped its opponents, and certainly will not make it any easier for us to enjoy more Unix in our working lives.

—Alexander Thorp, athorp@lucent.com

'Bug'

I saw a letter in your March 1999 issue, followed up in your May issue describing the genesis of the term 'bug', describing a flaw in some piece of equipment. The Oxford English Dictionary 2nd ed. usually provides a definitive origin; in this case they state:

bug.
b. defect or fault in a machine, plan or the like, orig U.S. 1889 Pall Mall Gaz. 11 Mar. 1/1 Mr Edison, I was informed, had been up the two previous nights discovering 'a bug' in his phonograph - an expression for solving a difficulty, and implying that some imaginary insect has secreted itself inside and is causing all the trouble.

The OED listing is the first known print use of the term 'bug', so it is quite possible although by no means certain that the term actually originated in the experimental laboratories of America's greatest inventor, Thomas Edison. Certainly it predates the start of the 20th century. Attributions to Admiral Hopper are clearly well short of the mark.

Sincerely,

—Eric H. Larson

At The Forge Factual Error - Hidden Fields - Don't trust them

Reuven,

On page 14 of the June 99 *LJ* issue, you state that a hidden field in a form can be useful for hardcoding a value that the user cannot change. This is not true. For example, I could save you form to disk, edit the value, and POST it to your handler. Your code needs to validate such fields on the back end or else bad things can and will happen.

I use the Digest::MD5 module to verify that my hidden fields sourced user data is not tampered with (this technique is also recommended for Netscape cookies processing - a clever user can fiddle with cookies in ~./ netscape/cookies).

Also, I'd suggest that you look into the MIME::Lite module for sending mail. It has the nice ability to handle attachments, which I often find the need to do from my mod_perl based applications.

Keep up the good columns, and if you can, how about some more mod_perl related stuff ?

—C. Jon Larsen, jlarsen@ajtech.com

“grep: Searching for Words” rife with errors

I'm a little bit late getting to read my April issue of *Linux Journal*, so no doubt somebody has already pointed this out, but “grep: Searching for Words” was such a poorly-written article, I just had to write in. It is certainly the worst article to appear in *Linux Journal* during the time I've been reading the publication (a few years now).

My complaints:

1. The worst inaccuracy in the article is where Jan says that because grep “accepts regular expressions”, you can use “grep flip *” to search for “flip” in “all files in the directory”. Yes, grep accepts regular expressions in its patterns, but this has nothing to do with the fact that most shells will expand '*' to be all the (non-dot) files in the current directory! This is a dangerous misconception to feed to Linux newbies, especially if they're coming from a DOS background, where it really _is_ the command-line programs' responsibility to interpret file-matching wildcards.

2. The weird self-referencing example output of the grep command (“article.txt: grep flip article.txt”) was confusing enough to make me have to re-read it a couple of times—I hate to think how it would have been for a newbie. I know space was limited, but the whole article would have been a lot clearer for beginners if it had started out with a very short example text file that all the grep examples would have been run against, rather than running them against the article itself (where each successive insert of grep results could invalidate the results of previous grep runs).

3. In any case, the example output is only correct if there are multiple files in the current directory containing the word “flip”. Other- wise, the string “article.txt: ” would not be prepended to the results. I'm sure many beginners were confused when they tried out the command and didn't get the type of output Jan said they would.

4. Next the -e option is introduced, but for no reason! “I put the -e (i.e., do pattern search) option in this example just for demon- stration purposes.” Just what is being demonstrated? In any case, -e doesn't tell grep to “do pattern search”—that's all grep does! It _specifies_ the pattern to search for. If the -e option is going to be introduced, why not use it for the only thing it's good for—searching for expressions that begin with a '-'? That's a non- obvious usage requirement that all grep users bang up against eventually, and would have been nice to mention here.

5. It's confusing to say that the output of grep -n “will look like that shown above, with the file name replaced by the line number before the colon”. What is the grep command that Jan is proposing here? 'grep -n “is the” filename' or 'grep -n flip *'? Since he references the output of the latter command, that seems to be the one he's talking about, and in that case, the line number won't _replace_ the file name before the colon—it'll be _appended_ to it.

6. Next egrep is introduced along with fgrep as a way to make grep faster. What?!? egrep, if anything, is slower than grep, not faster. The only example I can think of where egrep would be faster would be as an alternative to pipeline of normal grep commands. Jan could have made this point as he includes an example of such usage in the next section, but drops the ball. That would have been a good point to make as it's not necessarily desirable to be showing pipelines with multiple grep calls (without intervening output _transformation_) as the “right” way to do things.

7. And finally, the afore-mentioned “grep ... | sort ... | grep ... | cut ... > result” example is next-to-worthless for a text filter newbie, the kind of user being targeted here. If the article had contained a short example text file, as I suggest, an actual useful pipeline example (without '...'s) could have been given, and would have been much more illuminating.

In conclusion, I must stress again that the limited space afforded to this article is not a valid excuse for all the inaccuracies and misleading information. I could have easily written a much more accurate and informative article (or edited Jan's into one) in the same amount of space. This article was far, far below the writing and editing quality I have come to expect from *Linux Journal*.

—Dan Harkless, dan@wave.eng.uci.edu

> Sorry, my editors and I take the responsibilty for this one. English is not the first language of the

> author, and we obviously did not spend enough
> time on this one.
> —Editor

---

awk article in LJ

I like articles like yours in the most recent issue of *Linux Journal* about awk. They point readers to the fun of unix.

I too like awk and use it a lot for small, preferably command line, applications. However, I think it would not be useful to tell readers at the same time about the much greater capacities of Perl, thus preventing them to write larger applications in awk.

Your listing 4 can be written in Perl as follows:

```
#!/usr/bin/perl -w
open(IN,"/etc/passwd");
while (<IN>) {
  chomp;          # remove \n
  s/.*://;                    # keep last field only
  $shell{$_||"no shell"}++;   # increment shell's counter
                              # calling it "no shell" if empty
}
while(($k,$v)=each %shell) {  # print all counters
  printf("%3d users %3d%% %s\n",
    $v,int($v/$.*100+.5),$k); # $. number of lines read sofar
}
```

which, even though your awk script can be written a lot shorter, is much more compact, and yet easier to understand, Also, it counts and reports the usage of other shells beyond the three you selected (but this aspect could have been arranged in awk as well, using its associative array capacity.)

```
A (simpler and less verbose) one-liner would be:
perl -ne 'chomp; s/.*://; $s{$_||"no shell"}++; \
          if (eof) { while(($k,$v)=each %s)     \
          { print "$v\t$k\n" }}' /etc/passwd
```

Hartelijke groet,
—Wybo H. Dekke, wybo@servalys.hobby.nl

Stop the Presses

Matthew Cunningham

Issue #63, July 1999

Having just returned from back-to-back trade shows, COMDEX Spring and ISPCON, I can confirm what you already know—the IT community has not only heard of Linux, but is also embracing it.

Having just returned from back-to-back trade shows, COMDEX Spring and ISPCON, I can confirm what you already know—the IT community has not only heard of Linux, but is also embracing it. A major reason for this is the growing prominence of Linux at trade shows. In Chicago, there were about 15 Linux-related exhibitors, a respectable number for a non-Linux-specific show. Most ironic was the location of VA Research (now VA Linux Systems) directly across from Linux Hardware Solutions. The following week, the former acquired the latter for an undisclosed price. Any cautions given at the show by Kit Cosper, President of LHS, to put off purchasing decisions were well-heeded.

Crowd at Linux Pavilion; Mark Bolzern is on the left

Linux Journal and Linux International sponsored an impressive pavilion and theater, with informative presentations from companies such as Caldera, Cygnus and the always entertaining “BRU Guys”. Also presenting was Jon “maddog” Hall, who wasted no time in challenging the FUD (fear, uncertainty and doubt) that was leaching from the Microsoft area of the show. All of these talks were broadcast live on “The Linux Show”, an Internet-based radio show about Linux (http://www.ttalk.com/). Linus Torvalds drew an enthusiastic crowd of at least 300 while being interviewed by Jeff Gerhardt, host of “The Linux Show”. Kudos to Jeff for asking Linus to autograph his *Linux Journal* hat.

Linus with Sonny Saslaw of Ziff Davis and your intrepid reporter

Obviously afraid of offending their largest exhibitor, the folks at COMDEX were on a non-committal tightrope walk that proved discouraging, and at times, offensive. As Doc Searls mentions in his *Open Sources* review in this issue, the drawing power of Linus Torvalds' keynote speech was obviously underestimated. A standing-room-only crowd of 1200 included not only those who wanted to learn more about Linux, but an impressive number of media people who were there to cover the event as well. The concept of the Gates keynote being pitted against the Torvalds keynote was more a blatant effort on the part of COMDEX to draw attention to the show than actual rivalry, although the idea of showing up in a penguin costume during the Gates keynote crossed my mind. With any luck, the lessons COMDEX learned from this show will improve their treatment of the Linux crowd this fall in Las Vegas.

Compile the kernel or the Penguin gets it!

Norman Jacobowitz shows Linus and maddog his WANpipe technique

ISPCON was just as exciting as COMDEX, but with a more Linux-friendly and knowledgeable crowd. *Linux Journal* sponsored a “Linux Luncheon”, with Phil Hughes challenging the crowd of 500 to answer Linux-related trivia questions.

Phil also moderated a discussion panel with representatives from VA Linux Systems, Cobalt and Rebel.com's (name subject to change) Netwinder division. It was good to see hardware vendors getting along so nicely.

Linus with his beautiful family—Tove, Patricia and Daniela.

Birthday cake!

The untold story behind all these trade shows is the folks lucky enough to work them. Sure, they are grueling and expensive, but at the same time, they give the feeling that you are a part of something larger than yourself, an ambassador or spokesperson to the uninitiated, unwashed masses. The Linux “trade show circuit” has engendered family of its own, and always, for some unknown reason, results in daily hangovers, not entirely caused by “virtual” beer. What do we talk about after discussing Linux all day? Linux, of course, and the nice part is that we are getting paid for it—truly a dream job. Chicago also allowed me the opportunity to have dinner with Linus Torvalds not once, but three nights in a row. I will always remember, although a bit hazily, sitting in Trader Vic's with Linus and maddog, drinking some obnoxious concoction from a two-foot-long straw. What I can draw from this experience is: don't challenge maddog to a drinking contest—he will always win; $8 drinks should be bought only on a company expense account; and the earliest we can expect to hear something from Transmeta is this fall.

maddog speaks at the *Linux Journal*-Linux International Presentation Theater

Dean Taylor demonstrates the Caldera 2.2 install in the *Linux Journal*-Linux International Presentation Theater

My fear for the future is that the growth of so many Linux-related companies will result in less communication among the people who drive each of them. The presidents and CEOs of these companies are being replaced by self-proclaimed “booth bimbos”. There is something to be said for an evening I spent at a restaurant in Chicago. Joining me were folks from Red Hat, SuSE and Caldera, all of us sharing a pitcher of beer. Trade shows are, in a sense, the only chance we get to have a Linux “company” meeting. It is my hope that with the growth of these companies, the importance of the trade show in maintaining and promoting a feeling of community will not be lost.

Matthew Cunningham does marketing and trade shows for *Linux Journal*. He can be found in the *LJ* booth passing out Penguin peppermints, or contacted at info@linuxjournal.com.

New Products

Ellen Dahl

Issue #63, July 1999

LML33, Aestiva HTML/OS, I-Gear 3.0 and more.

LML33

Linux Media Labs announced its video capture/compression/playback MJPEG PCI card, LML33, for Red Hat Linux 5.2. The board is closely based on Zoran's reference design. Changes to it are kept open and the software drivers are developed under the GPL. Parameters include compression rates of 3.5 to 30; video-stream DMA transfer into video-board memory or RAM, which allows for video in a window; composite and S-video analog input/output. LML33 can be ordered for $410 US.

Contact: Linux Media Labs LLC, Phone: 719-231-3173, Fax: 719-593-9452, E-mail: vleo@linuxmedialabs.com, URL: http://linuxmedialabs.com/.

Aestiva HTML/OS

Aestiva, LLC announced the availability of Aestiva HTML/OS, a powerful web-based software product for developing and hosting advanced, dynamic web sites. This version targets users wishing to build commerce sites, web-based applications, or high-access dynamic web sites. Its tagging language allows HTML documents to merge with other server information on the fly, and performs high-speed word processing, spreadsheet, database and programming operations. It eliminates the need for CGI programming. Aestiva HTML/OS is available for Mac OS 7 to 8.5, Mac OS X Server, Linux (Intel and DEC Alpha) and other platforms. It retails for $799 US. Upgrades are $150 US.

Contact: Aestiva, LLC, 400 Crenshaw Blvd., Suite 109, Torrance, CA 90503, Phone: 310-328-8122, Fax: 310-328-8403, E-mail: sales@aestiva.com, URL: http://www.aestiva.com/.

I-Gear 3.0

URLabs, Inc. is shipping I-Gear version 3.0 for Red Hat Linux 5.2. I-Gear is the company's Internet content management product that couples high-performance caching with filtering features such as Content Category Lists and Dynamic Document Review for regulating web access or enforcing acceptable-use policies. The Linux version includes all enhancements released as a part of I-Gear 3.0, including features specifically requested by security-conscious companies and family-oriented ISPs. Visit the web site for pricing and download information.

Contact: Unified Research Laboratories, Inc., 303 Butler Farm Road, Suite 106, Hampton, VA 23666-1568, Phone: 757-865-0810, Fax: 757-865-4528, URL: http://www.urlabs.com/.

Leeloo v1.4

JET Software, Inc. announced the release of Leeloo v1.4. Leeloo is a simple-to-use authoring tool for producing Internet or intranet demos, support and training materials. Created sequences can be viewed without plug-ins from a web server with any recent browser on Linux and other platforms. A free version of the authoring tool may be downloaded from http://www.jetsoftware.com/. Sequences created with the free version will carry banners. A licensed copy of Leeloo removes banners. Price is $999 US for one author, lower for more authors. Sequences with a banner are free; without banners, $400 US per licensed copy.

Contact: JET Software, 2055 Gateway Place, Ste. 400, San Jose, CA 95110, Phone: 408-451-3947, Fax: 408-971-3578, E-mail: sales@jetsoftware.com, URL: http://www.jetsoftware.com/.

8000 8-way SMP Server for Linux

Penguin Computing, Inc. announced that it is now offering an 8-way SMP server for Linux. Penguin has built the fastest possible Intel-based computer by combining an L3 cache design adopted from mainframe technology with the industry-standard Intel Xeon processors. Penguin Computing offers its customers the option to have Internet applications, such as the Apache web server and Oracle applications and databases, pre-installed on all servers. Pricing for the base system will vary depending on options such as CPU, memory, number of hard drives, type of monitor and tape backup.

Contact: Penguin Computing, 965 Mission Street, Suite 630, San Francisco, CA 94103, Phone: 888-PENGUIN, 415-243-8100, Fax: 415-243-8181, E-mail: sales@penguincomputing.com, URL: http://www.penguincomputing.com/.

Proven CHOICE

Proven Software, Inc. announced the release of Proven CHOICE, their off-the-shelf general business accounting package written specifically for Linux. Proven CHOICE for Linux includes Sales Invoicing/Accounts Receivable, CheckWriter/ Accounts Payable, General Ledger and Financial Report Generator bundled for $499 US. Users may work with state-of-the-art equipment or use inexpensive character-based terminals and terminal emulators for multi-user entry. This flexibility creates the most efficient and economical installation possible for any end user.

Contact: Proven Software, Inc., P.O. Box 476, Manlius, NY 13104-0476, Phone: 800-487-6532, Fax: 315-682-1142, E-mail: info@provenacct.com, URL: http://provenacct.com/.

PostShop, ScanShop, OCR Shop, FaxShop version 4.0

Vividata, Inc. announced the release of version 4.0 of its product suite, which includes PostShop, ScanShop, OCR Shop and FaxShop imaging software for UNIX. With version 4.0, Vividata has a simplified installation process for all products. Updated with the latest commercially licensed Ghostscript, PostShop now provides additional drivers, many PostScript Level 3 enhancements, and expanded support for new printers and scanners from companies such as

Canon, Kodak, Hewlett Packard, Lexmark and Epson. PostShop and ScanShop are available starting at $99 US for Linux X86 personal edition, and OCR Shop for Linux at $299 US.

Contact: Vividata, Inc., 1250 Addison St., Ste. 213A, Berkeley, CA 94702, Phone: 510-841-6400, Fax: 510-841-9661, E-mail: sales@vividata.com, URL: http://www.vividata.com/.

WipeOut 1.5

SMB announced the newest release of the WipeOut 1.5 IDE for Linux, Solaris and FreeBSD. Available for free evaluation download, the evaluation copy is a fully functioning version for single users. The package includes team-based project management, revision control, class browser, editor, debugger, make tool and integrated on-line help. Pricing for WipeOut 1.5 begins at $85 US for educational usage. The single-user price for WipeOut Pro is $295 US, $445 US for the WipeOut ProTeam edition.

Contact: SMB, Neustaedter Markt 4, 04315 Leipzig, Germany, Phone: +49 (0) 341-699-4604, Fax: +49 (0) 341-699-4704, E-mail: wipeout@softwarebuero.de, URL: www.softwarebuero.de/wipeout-eng.html.

ODBC SDK for Linux

Dharma Systems, Inc. unveiled the industry's first Linux-based ODBC software development kit (SDK). Dharma's announcement makes it the first vendor in the industry to offer an ODBC data access solution for application vendors who have or want to have their data or applications run on Linux. The Dharma ODBC SDK provides easy-to-implement, high-performance access from any ODBC client tool to data on Linux servers. Dharma ODBC SDK Lite v6.2 for Linux is available as a no-charge, royalty-free download from Dharma's web site or as a fully supported version (Dharma ODBC Professional for Linux) directly from the company.

Contact: Dharma Systems, Inc., 436 Amherst Street, Nashua, NH 03063, Phone: 603-886-1400, Fax: 603-883-6904, E-mail: info@dharma.com, URL: http://www.dharma.com/.

IPSec VPN

The most popular enhancement option for the Paktronix Systems Firewall is IPSec VPN. The Paktronix Systems VPN connects any two or more networks across the Internet using full IPSec encryption technology. Advanced routing structures within the Linux 2.2 kernel enable multiple redundant routes with failover and priority routing services. The CPU of the firewall system is completely free to deal with firewall and advanced security functions. The IPSec hardware is not directly connected to any physical network, thus providing maximal protection and management of the VPN data streams. The Linux 2.2 kernel provides full RFC compliance for all network routing functions. Please contact Paktronix or its sister site, Midwest Linux, for ordering information.

Contact: Paktronix Systems, LLC, 1506 North 59th Street, Omaha, NE 68104, Phone: 402-932-7250, E-mail: sales@paktronix.com, sales@midwestlinux.com, URLs: http://www.paktronix.com/, www.midwestlinux.com/products/pakprods.html.

Best of Technical Support

Various

Issue #63, July 1999

Our experts answer your technical questions.

Linux General Question

I am new to the Linux world and have what I think is a fairly simple question. I would like to know how to get information from a CD using the X Window System. Can you help me with this? —Anthony, DrexelDG@aol.com

First, mount your CD-ROM using the command **mount /cdrom** if /etc/fstab is properly configured. Otherwise, as root use **mount /dev/hdb /cdrom.** Your distribution probably offers some graphic tools to help with the task; remember **man mount** is your friend. —Alessandro Rubini, rubini@prosa.it

First, in an xterm, log in as root (**su -root** and type in the root password when prompted). Then, create a directory where the CD will appear to be:

```
mkdir /mnt/cdrom
```

Next, insert the CD and mount it on the directory you created:

```
mount -t iso9660 -o ro /dev/hdd /mnt/cdrom
```

Then, you should be able to cd to /mnt/cdrom and use the files on the CD. When you're done, the command

```
umount /mnt/cdrom
```

will tell Linux to “let go” of the CD so you can eject it and work on another. —Scott Maxwell, s-max@pacbell.net

Error when Booting

When LILO starts to boot Linux, I get an error message saying, “BIOS32 extension does not exist. Sorry, this release still depends on it!” I am unable to boot Linux because of this. What should I do? —Mark, msudnick@ionsys.com

LILO must be installed on your MBR (Master Boot Record), on a floppy disk or the first sector of your Linux partition. If you have an IDE drive, it must be located before the 540MB limit; if you have a SCSI drive, it must be lower than cylinder 1024. Is your LILO installed in the proper location? —Daniel Lafraia, lafraia@iron.com.br

This message is printed by a SCSI driver which is compiled in your default kernel and is benign as you don't have that SCSI controller. Most drivers don't print any message when they fail detecting the hardware; in my opinion, this is good. It looks as if your kernel is locking when probing for some other hardware, but I can't tell which one. You should try to remove any strange ISA device installed in your system, then compile your own kernel, leaving out everything of no interest to you.

Note that most modern distributions use modules for each device driver in order to avoid probing for uninstalled peripherals. —Alessandro Rubini, rubini@prosa.it

Modem Speed

At what speed is my modem connecting? I have **REPORT CONNECT** in my **chat** script, but it always indicates I am connected at 57600. How do I query the modem or **pppd** to find out the speed at which I am actually connected? —James M. Pothering, jmpothering@netscape.net

You need to make a change in your modem settings. Consult your modem's manual to learn which string to send, then use a terminal program such as **minicom** to instruct your modem to return the modem-to-remote (DCE) speed, not the computer-to-modem (DTE) speed. For example, for my modem I would type **ATW2&W**. This turns on “Report connection rate only as CONNECT XXXX” (W2) and saves the settings for the next use (&W). —Chad Robinson, chadr@brt.com

Applications Stopped Running

I've upgraded to glibc 2.1, but certain older applications (StarOffice in particular) don't want to run now. How do I either convince StarOffice to like the new glibc, or else keep the old 2.0.7 installed just for StarOffice's use? —Tim Pepper, tpepper@calpoly.edu

Here's a shot in the dark... Instead of executing “soffice” by itself to start StarOffice, execute the following:

```
LD_LIBRARY_PATH=/glibc_libraries_path soffice
```

*Be sure to have the 2.0.7 libraries in a directory separate from all the rest of your system libraries (they typically reside in /lib). The command above should make StarOffice look for shared libraries in the **LD_LIBRARY_PATH** first before moving on to look elsewhere. Note that you do not want this particular **LD_LIBRARY_PATH** variable to be permanent; you just want to execute it when you start StarOffice. (Of course, replace **/glibc_libraries_path** with the actual path to your glibc 2.0.7 libraries.) —Erik Ratcliffe, erik@calderasystems.com*

You can influence how any application searches for shared libraries by setting the **LD_LIBRARY_PATH** environment variable (don't forget to use **export LD_LIBRARY_PATH=...**, so the environment variable's value will be visible to applications you run). The value of this variable should be a colon-separated list of directories to be searched before the usual locations.

Writing a simple shell script will make life simpler. Do something like this:

```
#!/bin/bash
export LD_LIBRARY_PATH=/path_old_glibc/
exec soffice
```

If necessary, you can also force the issue with LD_PRELOAD:

```
#!/bin/bash
export LD_PRELOAD=/path_old_libc.so
exec soffice
```

—Scott Maxwell, s-max@pacbell.net

Installation Question

I would like to know if it is possible to install a Red Hat 5.1 or 5.2 system from a parallel port Zip drive. —Chris Bensch, chrisbensch@iname.com

Yes, it should work. The distribution won't autodetect your Zip drive (I'm fairly certain), but you can tell it you have a SCSI controller and select the **ppa** controller, which really is SCSI, over your parallel port. You should then be able to do your install. However, Red Hat will not fit on a Zip drive. You will have to get rid of many of the packages in the /RPMS directory, and the installer will complain about not finding those packages, but should proceed with the install anyway. If you carefully choose which packages to install on the Zip drive, you may end up with a working system. In other words, it's much easier to use a Jaz drive or some other device that can hold at least half a gigabyte. —Marc Merlin, marc@merlins.org

Creating Mount Points

I am trying to install the Oracle8 database and don't know how to create mount points. On the installation guide, it says I need to make one software mount point /u01 and three DB mount points /u02, /u03 and /u04. Please tell me how to create them and what the difference is between these mount points. —Tim Wu, tim@vbisd.org

A mount point is just a directory. On a UNIX platform, file systems are not referred to with drive letters. They are mounted, at which time they become part of the root file system mounted on /. It is at this location that you create /u01, /u02, /u03 and /u04. You may need to check the permissions required for those directories. Also, Oracle may require you to create partitions to actually mount, but that is another issue. —Chad Robinson, chadr@brt.com

Resources

Archive Index Issue Table of Contents

Advanced search

The Linux Position

Doc Searls

Issue #63, July 1999

Most companies—especially those in the technology world—are not interested in simple answers to simple questions.

For most of my adult life I have been two things: a journalist and a marketing expert. Frankly, I did the latter mostly to sublimate the former. This made me a very different kind of marketing expert—one who brought a writer's skepticism to the marketer's job. At every meeting, I always seemed to be asking the same two questions: what is this and what's the story?

Most companies—especially those in the technology world—are not interested in simple answers to simple questions. They make “solutions” rather than products and “management inference engines” rather than spreadsheets. They also want their stories to consist entirely of happy beginnings.

“Reporters write stories,” I would say, “and stories don't start with *happily ever after*. They start with a character with a problem. The solution comes at the end, and you never want to get there or the writer will find some other story to tell.”

I didn't get very far with this approach. What got me somewhere was working on the character issue. I called this “positioning” and I wasn't alone.

Hotbot finds 233,780 pages on the Web with the word “positioning” in them. When I weed out GPS and other non-marketing meanings, I get 26,940 pages, almost entirely by marketing consultancies selling “strategic distribution analysis”, “rollout plan reviews”, “campaign launch programs”, “performance impact studies”, “collateral market options”, “market penetration analyses”, “outsourced staff deployments” and other such nonsense, all served up in euphemistically delusional language.

No wonder Linux is a hit. It is a character with a story none of those 27,000 agencies—including mine—could have thought up. Who would seriously talk about “world domination” and “software that doesn't suck” in the face of Microsoft, whose software runs on every computer you see and whose market value exceeds the GNP of the Southern Hemisphere? *“You see, there was this Finnish guy, and something about a penguin...”* I don't think so.

Thus, what we have with Linux is more than the world's first big-time open-source operating system. We have the world's first marketing success that owes nothing to marketing. This warrants further study.

I just did for Linux what I used to do for my clients. I took a look at how the customers for Linux's message—the analysts, reporters and editors of the world—are describing its character and telling its story. I got on AltaVista (http://www.altavista.com/) and looked up every page with the phrase “Linux is...” and found over 27,000. Searching through links to the first fifty, I came up with these answers, which I sort into four ways of depicting Linux's character:

Descriptive (you've got to start somewhere)

- an open-source UNIX clone
- a freely available UNIX clone
- a UNIX clone
- a complete, copylefted UN*X clone
- a freely distributable, independent UNIX-like operating system
- a UNIX-type 32-bit operating system
- an operating system developed under the GNU General Public License
- an open-source operating system anyone can download from the Internet and compile
- a freeware version of UNIX
- a true 32-bit, multi-tasking, multi-threading operating system
- a full-featured UNIX-type operating system
- a powerful, flexible, 32-bit OS
- a 32-bit multi-user, multi-tasking clone of UNIX
- a full-fledged UNIX-like operating system
- a UNIX-type 32-bit operating system
- an embedded operating system
- the open-source operating system

Superlative (stuff to like)

- free
- stable
- awesome
- a full, rich, dependable workhorse
- the best Windows file server
- a bedroom hacker's dream
- the OS to run
- the only real OS
- a bandwagon
- the future's universal operating system
- a significant OS in corporate IS departments
- the #1 OS in Germany
- the #1 UNIX on x86
- the heir to UNIX
- the largest collaborative programming effort ever
- the new king of the hill
- the most dynamic, interesting and exciting development on the operating system scene today
- not just for geeks anymore
- a lesson in hard work and well-earned rewards
- the first major evolution in operating systems since MS-DOS

Competitive (necessary for the war and sports stories that write themselves)

- the fastest-growing OS outside of NT
- a worthy contender
- a juggernaut
- a movement
- the definitive answer to the small business needs for Internet connectivity, a web server and e-mail services

Flawed (always a good character trait)

- complicated
- a good teacher of science-oriented college students, but a rotten teacher for just about everyone else
- overkill for most home applications
- simply too hard to use for the average user

- entrenched in a grass-roots development model

Believe me, you can't buy PR this good. Especially since the default story is about a fight between this Finnish dude and a Schwarzenegger character and it isn't an act. Here's a look at six different “versus” constructions:

- Linux vs. Microsoft: 57
- Microsoft vs. Linux: 110
- Linux vs. NT: 512
- Linux vs. (Windows, Cisco, BeOS...) 1968
- Microsoft vs. (DOJ, Justice, Netware...) 2721

A big part of positioning is unconscious, yet revealed by who you list first. Is it Yankees vs. Dodgers or Dodgers vs. Yankees? We tend to list favorites first. So if I had to call a play-by-play on the games here, I'd say Microsoft appears to be the favorite in the company game, while Linux is the favorite in the OS game (which, fortunately, is the one that truly matters).

Even when editors don't use the “versus” construction, they still apply a rule I obtained recently from a *Wall Street Journal* reporter: “These days you can't write about Microsoft without bringing up Linux.” It's *pro forma*. Even if they know nothing about Linux, they still cast its character.

And what about Microsoft, really? Does this corporate Terminator truly consider Linux to be a competitor? “This much is clear to them, right to the top of the company.” One cross-platform developer told me this morning, “They can't win the server war. There's no way they can lock it down like they did with the desktop. Linux is now the server of choice. They know they have to cope with that, and they're really not sure how to do it.”

Let's hope they start figuring out how. Microsoft is a problem we don't want to lose.

Parallel Algorithms for Calculating Underground Water Quality

Tran Van Lang

Issue #63, July 1999

The PVM system helps us in designing parallel processing programs for multi-computer systems. The implementation of the parallel algorithm enables the solving of large mechanic problems requiring large amounts of computer time and memory.

In scientific calculations, we often process a large amount of data, requiring much computer time. Thus, the studying of parallel and distributed algorithms is a serious requirement in the developmental computer age. In this paper, I discuss the implementation of algorithms solving engineering mechanics problems in the form of parabolic differential equations, in particular the problem for transfer and diffusion of pollution in underground water. The problem's algorithms are implemented in a heterogeneous network of computers running Linux. Thus, we will obtain a distributed network including many processors in the form of a multi-computer. This multi-computer has formed a *Parallel Virtual Machine* (PVM). Developing applications for the PVM system follows the traditional paradigm for programming distributed-memory multiprocessors such as the nCUBE or the Intel family of multiprocessors. This paradigm can be subdivided into two categories:

- The master-slave (host-node) model in which a separate *control* program termed the master is responsible for process spawning, initialization, collection and display of results, and timing of functions. The slave programs perform the actual computations; they are either allocated their workloads by the master (statically or dynamically) or they perform the allocations themselves.
- The node-only model where multiple instances of a single program execute, with one process (typically the one initiated manually) taking over

the non-computational responsibilities in addition to contributing to the computation.

We used the master-slave model to implement the parallel algorithms for our water pollution problem. We also proposed and proved some mathematical questions; for example, the establishment of an approximated solution and an estimation of the error contained in the approximated solution.

Parallel Algorithms

In this paper, we apply the parallel algorithm to calculate the diffusion S(t,x,y) of polluted matter resource of underground water by using the following equation. This is the evolution problem in the form of a quasilinear equation:

$$s\frac{\partial S}{\partial t} = \frac{\nabla.\left[(z - z_d)LK|\nabla z|\nabla S\right]}{z - z_d} + K\nabla z.\nabla S + Q$$

Figure 1

where the variables are defined as follows:

- **- s**: spongy coefficient of environment
- **- K** : coefficient of penetration
- **- L**: dispersion
- **- z = z(t,x,y)**: free surface of underground water
- **- zd = zd(x,y)**: director surface
- **- Q = Q(t,x,y)**: resource of polluted matter

We discrete the problem by using the difference method. Therefore, we will obtain two tridiagonal systems of linear equations. Using these equations, we can calculate to find the concertrantion of pollution Sn+0.25, Sn+0.50 at the point in time equal to (n+0.25)Dt and (n+0.50)Dt. The flow-chart of the parallel algorithm is shown in Figure 1.

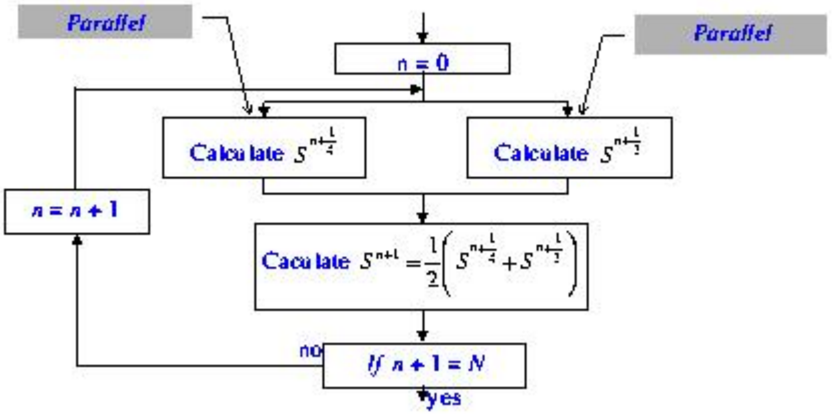

Parallel
Parallel
n = 0
Calculate $S^{n+\frac{1}{4}}$
Calculate $S^{n+\frac{1}{2}}$
n = n + 1
Calculate $S^{n+1}=\frac{1}{2}\left(S^{n+\frac{1}{4}}+S^{n+\frac{1}{2}}\right)$
no
If n + 1 = N
yes

Figure 1. Flow-chart

The values Sn+0.25 and Sn+0.50 are calculated concurrently on two processors of our virtual machine that form a shared or local-memory multiprocessor.

Parallel Virtual Machine

PVM is a portable message-passing programming system designed to link separate host machines to form a “virtual machine” that is a single, manageable computing resource. The virtual machine can be composed of hosts of varying type in physically remote locations. PVM applications can be composed of any number of separate processes, or components, written in a mixture of C/C++ and FORTRAN. The system is portable to a wide variety of architectures, including workstations, multiprocessors, supercomputers and PCs. The most common PVM platform is UNIX, and it currently runs on more than 30 different platforms.

In the master-slave model, the master program spawns and directs a number of slave programs which perform computations. PVM is not restricted to this model. For example, any PVM task can initiate processes on the machines; however, a master-slave model is a useful programming paradigm and simple to illustrate. The master calls **pvm_mytid**, which, as the first PVM call, enrolls this task in the PVM system. It then calls **pvm_spawn** to execute a given number of slave programs on other machines in PVM. The master program contains an example of broadcasting messages in PVM. The master broadcasts to the slave the number of slaves started and a list of all the slave tids (task IDs). Each slave program calls **pvm-tid** to determine its task ID in the virtual machine, then uses the data broadcast from the master to create a unique ordering from 0 to nproc minus 1 (nproc-1), where nproc is the number of slave processes. Subsequently, **pvm_send** and **pvm_recv** are used to pass messages between processes. When finished, all PVM programs call **pvm_exit** to allow PVM to disconnect any sockets to the process, flush I/O buffers and keep track of which processes are currently running.

The SPMD (single-program, multiple-data) model contains only a single program, and no master program directs the computation. Such programs are sometimes called hostless programs. However, all processes still need to get started initially. The user starts the first copy of the program; by checking **pvm_parent**, this first copy can determine it was not spawned by PVM and can thereby know itself to be the first copy. It then spawns multiple copies of itself and passes them the array of tids. At this point, each copy is equal and can work on its partition of the data in collaboration with the other processes. Using pvm_parent precludes starting the SPMD program from the PVM console because pvm_parent will return the tid of the console. This type of SPMD program must be started from a UNIX prompt.

For our program, we used a master-slave programming model. The master program spawns a copy of the slave program to the other processor to calculate Sn+0.25, and calculates Sn+0.50 itself. When both processors have completed the calculation, the master program will calculate the sum Sn+1 = 0.5(Sn+0.25 + Sn+0.50). The program will finish when the final step time is completed. The progress of parallel computing in calculating Sn+0.25 and Sn+0.25 is shown in Figure 2.

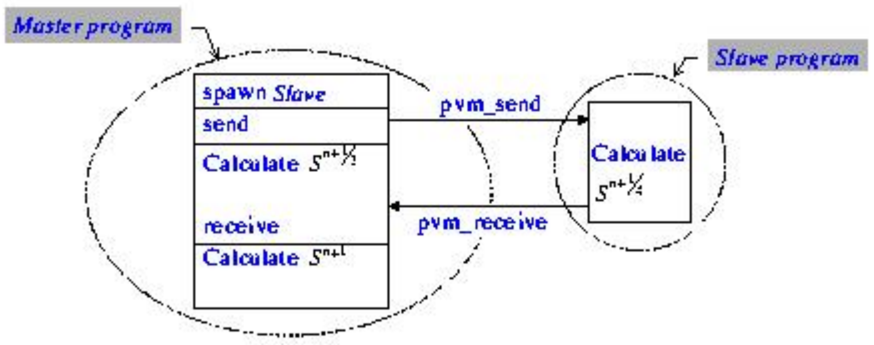


Figure 2. Progress of Parallel Computing

Linear system equations are also in the master and slave programs, so we also use the master-slave programming model to solve the tridiagonal systems by Linear Recurrence of order two. Horner's algorithm can be implemented to evaluate a polynomial of degree **n** at a single point to run in O(log2n). If we use sequential algorithms of the LDU factorization of a tridiagonal matrix, we will run it in O(n).

Results

Figure 3. Xndd program

We designed and built the X Window System program **Xndd** under Linux. This program can be used for simulating the surface of underground water and the transfer and diffusion of pollution. The interface of the program and some calculation results are shown in Figure 3.

Resources

Tran Van Lang can be reached via e-mail at lang@netnam2.org.vn. He is a research scientist for the Institute of Information Technology branch in Ho Chi Minh City, doing system analysis and design. He uses Linux as a development platform and builds tools for parallel and distributed computations. He enjoys spending his off-work time with his wife and two daughters.

Archive Index Issue Table of Contents

Advanced search

MuPAD

Alasdair McAndrew

Issue #63, July 1999

MuPAD deserves the full support of the Linux community, and if you use mathematics in any way, then MuPAD should find a home on your system.

A Computer Algebra System (CAS) is a software package designed for the symbolic manipulation of mathematical expressions. A CAS should be able to:

- perform numerical arithmetic with precision bounded only by the computer's hardware
- perform basic calculus: partial and complete differentiation; symbolic and numerical integration
- solve differential equations (partial and total)
- manipulate power series
- simplify algebraic expressions
- understand standard functions: exponential, trigonometric, hyperbolic; in particular their derivatives and anti-derivatives, and their power series expansions
- have a knowledge of some other functions: hypergeometric, Bessel
- manipulate matrices and vectors; basic arithmetic, eigensystems, determinants, matrix decomposition
- create two- and three-dimensional graphs
- produce output in various forms: TeX/LaTeX, Fortran, C, HTML, PostScript
- interact with other software; the CAS should be able to incorporate programs written in other languages
- be extended: the CAS should have some programming language built in so that a user can add further functionality

As well as these basic requirements, many current CASs can do much more. Further capabilities may include:

- number theoretical functions: primality testing, modular arithmetic, primitive root finding
- orthogonal functions: Legendre, Chebyshev, Laguerre
- animation of sequences of graphs
- recurrence relation solving

There are a number of very mature and full-featured CASs available now. Maple and Mathematica are probably the most popular all-round packages; others include Mathcad, Matlab, Axiom, Derive, Macsyma, Reduce, and of course MuPAD.

MuPAD

MuPAD is a CAS developed at the University of Paderborn by a team headed by Benno Fuchssteiner. Although it may not have quite the range and pizazz of its better known rivals Maple and Mathematica, it is equal to them in depth, and in some ways even surpasses them. Its name is short for “MultiProcessing Algebra Data Tool”, and as we shall see, is a fairly good descriptor of it.

In contrast to the major commercial CASs, MuPAD is designed to be an open system—anybody can extend and add to it. The current version is 1.4, and was released in March 1998.

First Impressions

Let us suppose that you have installed MuPAD on your system. (I will discuss installation later.) What now?

You can start MuPAD in two ways: in a console, using the command

```
mupad
```

or, if you are running X, the command

```
xmupad
```

will provide a graphical interface. If you have installed MuPAD correctly, both of these commands are, in fact, shell scripts which define certain environment variables and then call the actual executables.

Let us suppose you have chosen the latter. You will obtain a window something like that shown in Figure 1.

Figure 1. The Initial MuPAD Window

Notice the OpenWindows look and feel. As yet the MuPAD team has not released a Linux binary using Athena or Motif widgets, although one is in preparation. MuPAD was designed to work with the OpenWindows widget set, and as such behaves best if you use an OpenLook window manager. Many of its subsidiary windows do not have their own close or exit button, but rely on the window manager for this. For all the figures and screenshots in this article I have used olvwm: the Open Look Virtual Window Manager.

You will also notice that the menus are rather sparse; there is, in fact, not a great deal more functionality offered by the graphical interface than in an intelligent console. Don't be put off—there's more here than meets the eye!

MuPAD in Action

To save displaying too many screenshots, we shall go back to console mode. All MuPAD expressions are terminated with a colon or semi-colon (the colon suppresses display of the output), and the syntax is similar to that of Maple.

When MuPAD is started, a kernel (written in C++) is loaded; this defines a number of basic commands, functions, and constants. Other commands are available in specialized libraries, written in MuPAD's own language. These commands have to be loaded explicitly: either individually, or with the entire library.

Numerics

We will start our investigation of MuPAD with a mathematical classic:

```
>> 2+2;
                                        4
```

Passed with flying colours! Now let's try some problems beyond the reach of your average hand-held calculator.

```
>> 3^(4^5);
373391848741020043532959754184866588225409776783734007750636931722079040617
26525122999368893880397722046876506543147515810872705459216085858135133698
280918731419174859426258093880701995195640428557181804104668128879740292551
766801234061729839657473161915238672304623512593489605859058828465479354
0505936202376547807442730582144527058988756251452817793413352141920744623
275187291854328623757370639854853194764169262638199728870069070138992565242
971985276987492741962768110607023337103564 81
```

As you see, MuPAD has no fear about dealing with very large integers.

```
>> DIGITS:=1000:float(PI);
3.14159265358979323846264338327950288419716939937510582097494459230781640
6286208998628034825342117067982148086513282306647093844609550582231725359408
128481117450284102701938521105559644622948954930381964428810975665933446128
4756482337867831652712019091456485669234603486104543266482133936072602491
```

```
141273724587006606315588174881520920962829254091715364367892590360011330530
548820466521384146951941511609433057270365759591953092186117381932611793105
118548074462379962749567351885752724891227938183011949129833673362440656643
086021394946395224737190702179860943702770539217176293176752384674818467669
405132000568127145263560827785771342757789609173637178721468440901224953430
146549585371050792279689258923542019956112129021960864034418159813629774771
309960518707211349999998372978049951059731732816096318595024459455346908302
642522308253344685035261931188171010003137838752886587533208381420617177669
147303598253490428755468731159562863882353787593751957781857780532171226806
6130019278766111959092164201995
```

The value **DIGITS** gives the number of significant digits when dealing with floating point values. Its default is 10, and it can be set to any value between 1 and 2^31 - 1.

```
>> DIGITS:=100:float(1/997);
0.001003009027081243731193580742226680040120361083249749247743229689067201604814443329989969909729187562
>> 43^(1/5);
2.121747460841897852639905031079416833442447899377300135845506404964677379294415637755003497680157377
```

Solution of Equations

The general command **solve** can be used to solve equations of all types: algebraic, differential, recurrence.

```
>> solve(x^2-4*x+3=0,x);
                                  {1, 3}
>> sols:=solve(a*x^3+b*x^2+c*x+d=0,x):
```

We will suppress the output as it is rather long, but let's see what we can do with it:

```
>> op(sols,1);
/                3     /   2              3      3       2  2  \1/2 \
| b c     d     b      |  d     b c d    c      b  d    b  c   |    |
| ---- - --- - ----- + | ---- - ----- + ----- + ----- - ------ |    |^(1/3)
|    2   2 a       3   |    2      3        3       4        4 |    |
\ 6 a          27 a    \ 4 a    6 a     27 a    27 a    108 a  /    /
           /                3     /   2              3      3       2  2  \
      b    | b c     d     b      |  d     b c d    c      b  d    b  c   |
   - --- + | ---- - --- - ----- - | ---- - ----- + ----- + ----- - ------ |
     3 a   |    2   2 a       3   |    2      3        3       4        4 |
           \ 6 a          27 a    \ 4 a    6 a     27 a    27 a    108 a  /
   1/2 \
       |
       |^(1/3)
       |
       /
```

The **op** command picks out subexpressions; in this case, as the result is a three-element set, we have chosen its first element.

```
>> generate::TeX(%);
"- \\frac{b}{3 a} + \\sqrt[3]{- \\frac{d}{2 a} + \\frac{b c}{6 a^2} - \\fr\
ac{b^3}{27 a^3} + \\sqrt{- \\frac{b c d}{6 a^3} + \\frac{d^2}{4 a^2} + \\f\
rac{c^3}{27 a^3} + \\frac{b^3 d}{27 a^4} - \\frac{b^2 c^2}{108 a^4}}} + \\\
sqrt[3]{- \\frac{d}{2 a} + \\frac{b c}{6 a^2} - \\frac{b^3}{27 a^3} - \\sq\
rt{- \\frac{b c d}{6 a^3} + \\frac{d^2}{4 a^2} + \\frac{c^3}{27 a^3} + \\f\
rac{b^3 d}{27 a^4} - \\frac{b^2 c^2}{108 a^4}}}"
```

The **TeX** command is one not automatically loaded when MuPAD is launched. To access it, we have to give its full address within MuPAD's libraries.

Here **%** refers to the output of the previous command. This result can now be saved to a file:

```
>> fprint("solution.tex",%);
```

MuPAD can also solve systems of algebraic equations.

```
>> solve({x+2*y+a*z=-1,a*x-3*y+z=3,2*x-a*y-5*z=2},{x,y,z});
   { {              2             2                           2      } }
   { {         a - a           3 a  - 5 a - 19      11 a - 2 a  + 19 } }
   { { z = --------------, x = ---------------, y = ---------------- } }
   { {      3                   3                     3              } }
   { {     a  - 17 a - 19      a  - 17 a - 19        a  - 17 a - 19  } }
```

The above system, being linear, could have been solved equally well by using the **linsolve** command.

Number Theory

There are a few basic number theory functions in the kernel; others are contained in the **numlib** library.

```
>> isprime(997);
                                   TRUE
>> Factor(2^67-1);
                          193707721 761838257287
>> nextprime(1000000);
                                  1000003
>> powermod(9382471,322973,1298377);
                                  880825
>> phi(nextprime(2^20)-1);
                                  498400
```

Here **phi** is Euler's totient function returning the number of integers less than and relatively prime to its argument. These functions allow us to perform simple RSA encryption and decryption. Suppose we choose two primes and compute their product:

```
>> p:=nextprime(5678);
                                   5683
>> q:=nextprime(6789);
                                   6791
>> N:=p*q;
                                 38593253
```

Now we have to choose an integer e relatively prime to (p-1)*(q-1); a smaller prime will do; say e:=17.

```
>> e:=17:
```

The values e and N are our “public key”. Now we find the d, the inverse of e modulo (p-1)*(q-1). This is very easily done using the convenient overloading of the reciprocal function:

```
>> d:=1/e mod (p-1)*(q-1);
                                  6808373
```

Suppose someone wishes to send us a message M < N; say

```
>> M:=24367139;
```

They can encrypt it using our public key values:

```
>> M1:=powermod(M,e,N);
                                 18476508
```

We can now decrypt this using the value d (and N):

```
>> powermod(M1,d,N);
                                 24367139
```

This is indeed the value of the original message.

Symbolic Manipulation

We have seen a glimpse of MuPAD's symbolic abilities in the equation solving above. But MuPAD can do much more than this: all manner of algebraic simplification; rewriting in a different form; partial fractions; and so on.

```
>> expand((x+2*y-3*z)^4);
 4       4       4         3      3            3       3            3
x  + 16 y  + 81 z  + 32 x y  + 8 x  y - 108 x z  - 12 x  z - 216 y z  -
       3              2          2         2           2  2        2  2
   96 y  z + 216 x y z  - 144 x y  z - 72 x  y z + 24 x  y  + 54 x  z  +
        2  2
   216 y  z
>> Factor(%);
                                            4
                             (x + 2 y - 3 z)
>> sum(1/(k*(k+2)*(k+4)),k=1..n);
                                  2        3       4
                     310 n + 337 n  + 110 n  + 11 n
                 ----------------------------------------
                                2        3       4
                 4800 n + 3360 n  + 960 n  + 96 n  + 2304
>> partfrac(%);
               1           1           1           1
           --------- - --------- - --------- + --------- + 11/96
           8 (n + 3)   8 (n + 1)   8 (n + 2)   8 (n + 4)
>> normal(%);
                                  2        3       4
                     310 n + 337 n  + 110 n  + 11 n
                 ----------------------------------------
                                2        3       4
                 4800 n + 3360 n  + 960 n  + 96 n  + 2304
>> Factor(%);
                                            2
                      n (n + 5) (55 n + 11 n  + 62)
                    ----------------------------------
                    96 (n + 1) (n + 2) (n + 3) (n + 4)
>> sum(sin(k*x),k=1..n);
(I exp(-I x) - I exp(I x) + I exp(-I n x) - I exp(I n x) -
   I exp(- I x - I n x) + I exp(I x + I n x)) /
   4 - 2 exp(-I x) - 2 exp(I x)
>> rewrite(%,sincos);
(2 sin(x) + 2 sin(n x) + I cos(x + n x) - 2 sin(x + n x) - I cos(- x - n x)
   ) / 4 - 4 cos(x)
```

Calculus

MuPAD's calculus skills are excellent. It implements very powerful algorithms for differentiation, integration, and limits.

```
>> diff(x^3,x);
                                      2
                                   3 x
>> diff(exp(exp(x)),x$4);
                             2                       3
exp(x) exp(exp(x)) + 7 exp(x)  exp(exp(x)) + 6 exp(x)  exp(exp(x)) +
         4
   exp(x)  exp(exp(x))
```

The dollar operator, **$**, is MuPAD's sequencing operator. As with most operators, it is overloaded; in the context of a derivative it is interpreted as a multiple derivative. We can of course also perform partial differentiation.

```
>> int(sec(x),x);
                    ln(2 sin(x) + 2)   ln(2 - 2 sin(x))
                    ---------------- - ----------------
                           2                  2
>> int(cos(x)^3, x=-PI/4..PI/3);
                                 1/2      1/2
                              5 2      3 3
                              ------ + ------
                                12       8
>> int(E^(-x^2),x=0..0.5);
                           /    1                 \
                        int| --------, x = 0..0.5 |
                           |        2             |
                           |       x              |
                           \ exp(1)               /
>> float(%);
0.461281006412792448755702936740453103083759088964291146680472565934983884\
29529385671266224869994247 45
```

If we require only a numeric result, then we don't want to force MuPAD to attempt a symbolic or exact solution first. In such a case we may use the **hold** command, which returns the input unevaluated, but “holds” onto it for the purposes of later evaluation. Thus we may enter:

```
>> hold(int(exp(-x^2),x=0..0.5));
```

followed by the **float** command.

Series and Polynomials

The default order of a power series is 6. This can be changed either by changing the value of **ORDER** (analogous to **DIGITS** above), or by including the order in the **series** command. This last inclusion is optional.

```
>> tx:=series(tan(x),x,12);
                  3      5       7       9         11
                 x    2 x    17 x    62 x    1382 x        12
             x + -- + ---- + ----- + ----- + -------- + O(x  )
                 3     15     315    2835     155925
>> cx:=series(cos(x),x,12);
                    2    4    6      8        10
                   x    x    x      x        x          12
               1 - -- + -- - --- + ----- - ------- + O(x  )
                   2    24   720   40320   3628800
>> tx*cx;
                  3    5      7       9        11
                 x    x      x       x        x           12
             x - -- + --- - ---- + ------ - -------- + O(x  )
                 6    120   5040   362880   39916800
```

This certainly *looks* like the series for sin(x), but let's see if MuPAD recognizes it as such.

```
>> sx:=series(sin(x),x,12):
>> is(tx*cx=sx);
                                   TRUE

>> type(tx*cx);
                                  Puiseux
```

This means that the result of the series product is recognized by MuPAD as an object of type “Puiseux”; that is, a series possibly containing fractional powers.

MuPAD can also deal with polynomials.

```
>> p1:=x^8-3*x^5+11*x^4-x^2+17;
                            4    2      5    8
                        11 x  - x  - 3 x  + x  + 17
>> p2:=x^3-23*x^2-4*x+11;
                            3             2
                           x  - 4 x - 23 x  + 11
>> divide(p1,p2);
                  2        3       4    5
285641 x + 12337 x  + 533 x  + 23 x  + x  + 6613228,
                           2
   23310861 x + 153111100 x  - 72745491
```

The result of the last command consists of two terms: the quotient, and the remainder. MuPAD also has commands for extracting coefficients from polynomials, evaluating polynomials using Horner's algorithm, and lots more.

Linear Algebra

We shall first create a matrix domain:

```
>> M:=Dom::Matrix();
               Dom::Matrix(Dom::ExpressionField(id, iszero))
```

The result returned here indicates that MuPAD expects that the elements of matrices will be members of a field for which no normalization is performed (the function **id** just returns elements as given), and for which 0 is recognized as the zero value.

Now we shall make all the commands in the library **linalg** available to us:

```
>> export(linalg);
```

Now we shall create a few matrices and play with them. First, a matrix with given elements.

```
>> A:=M([[1,2,3],[-1,3,-2],[4,-5,2]]);
                             +-            -+
                             |   1,  2,  3  |
                             |              |
                             |  -1,  3, -2  |
                             |              |
                                 |   4, -5,  2  |
                                 +-            -+
```

We have entered the matrix elements as a list of lists (a list, in MuPAD, is delimited by square brackets).

Next, a matrix with elements randomly chosen to be between -9 and 9. We do this by applying the function returned by **random** to each of the elements.

```
>> B:=M(3,3,func(random(-9..9)(),i,j));
                                  +-            -+
                                  |  -7, -5, 8  |
                                  |             |
                                  |   3, -1, 7  |
                                  |             |
                                  |   3, -5, 6  |
                                  +-            -+
```

Clearly this approach can be used to generate any matrix whose elements are functions of their row and column values. There is a **randomMatrix** command in the **linalg** library, but it requires the elements to be members of a coefficient ring. For our purposes, it is as easy to roll our own.

```
>> A*B;
                                +-               -+
                                |   8,  -22, 40  |
                                |                |
                                |   10,  12,  1  |
                                |                |
                                |  -37, -25,  9  |
                                +-               -+
>> det(A);
                                        -37
>> 1/A;
                             +-                       -+
                             |  4/37,  19/37, 13/37  |
                             |                       |
                             |  6/37,  10/37,  1/37  |
                             |                       |
                             |  7/37, -13/37, -5/37  |
                             +-                       -+
```

As we have seen above, MuPAD supports operator overloading, which means that since **A** is a matrix, **1/A** is interpreted as the inverse of **A**.

```
>> A^10;
                      +-                                      -+
                      |   19897010, -20429930,  22281963  |
                      |                                   |
                      |  -42711893,  43857348, -47862790  |
                      |                                   |
                      |   64993856, -66730117,  72852811  |
                      +-                                      -+
>> b:=M(3,1,[7,9,-21]);
                                   +-      -+
                                   |   7   |
                                   |       |
                                   |   9   |
                                   |       |
                                   |  -21  |
                                   +-      -+
```

Here the first two (optional) values give the number of rows and columns of the matrix, the matrix elements are then given in a single list. If the list isn't long enough, the remaining values will default to zero.

```
>> linearSolve(A,b);
                                   +-    -+
                                   |  -2  |
                                   |      |
                                   |   3  |
                                   |      |
                                   |   1  |
                                   +-    -+
>> AM:=A.b;
                             +-                   -+
                             |   1,  2,  3,  7    |
                             |                    |
                             |  -1,  3, -2,  9    |
                             |                    |
                             |   4, -5,  2, -21   |
                             +-                   -+
```

The . operator is concatenation. Again, this is an overloaded operator, as it will work for other data types as well.

```
>> gaussJordan(AM);
                               +-              -+
                               |  1, 0, 0, -2  |
                               |               |
                               |  0, 1, 0,  3  |
                               |               |
                               |  0, 0, 1,  1  |
                               +-              -+
```

The **linalg** library is very full-featured, and contains plenty of commands for operating on matrices and vectors: row and column operations; matrix factorization and decomposition; commands for dealing with matrix polynomials and eigensystems; and so on.

Differential and Recurrence Relations

We first define a differential equation by defining it to be an equation in the **ode** domain.

```
>> de:=ode(y'(x)+4*y(x)=exp(3*x),y(x));
               ode(4 y(x) + diff(y(x), x) = exp(3 x), y(x))
```

Note that MuPAD supports the dash notation for the derivative.

```
>> solve(de);
                          {        3                }
                          { exp(x)                  }
                          { ------- + C1 exp(-4 x) }
                          {    7                    }
```

We can, of course, solve equations with initial conditions:

```
>> de2:=ode({y''(x)-2*y'(x)+3*y(x)=0,y(0)=1,y'(0)=2},y(x));
ode({D(y)(0) = 2, y(0) = 1, 3 y(x) - 2 diff(y(x), x) + diff(y(x), x, x) = 0
   }, y(x))
>> solve(de2);
             {                          1/2          1/2   }
             {                1/2    exp(x) 2    sin(x 2    ) }
```

```
                { exp(x) cos(x 2   ) + ------------------------ }
                {                                  2             }
```

MuPAD will solve linear differential equations with little fuss, and also a large class of non-linear differential equations.

Recurrence relations are solved similarly; by defining an equation to be of type **rec**, and applying the command **solve**.

```
   >> rr:=rec(a(n)=a(n-1)+3*a(n-2),a(n));
                     rec(a(n) = a(n - 1) + 3 a(n - 2), a(n))
   >> solve(rr);
                  {    /   1/2       \n      /         1/2 \n }
                  {    | 13          |       |       13    |  }
                  { a1 | ----- + 1/2 |  + a2 | 1/2 - ----- |  }
                  {    \   2         /       \         2   /  }
```

If we include initial conditions, MuPAD will return the exact solution.

```
   >> rr2:=rec(a(n)=a(n-1)+3*a(n-2),a(n),{a(0)=1,a(1)=3});
          rec(a(n) = a(n - 1) + 3 a(n - 2), a(n), {a(0) = 1, a(1) = 3})
   >> solve(rr2);
   { /           1/2 \ /         1/2 \n   /     1/2       \ /   1/2       \n }
   { |       5 13    | |       13    |    | 5 13          | | 13          |  }
   { | 1/2 - ------- | | 1/2 - ----- |  + | ------- + 1/2 | | ----- + 1/2 |  }
   { \         26    / \         2   /    \   26          / \   2         /  }
```

In the case of a non-homogeneous equation, MuPAD returns the particular solution.

```
   >> rr3:=rec(a(n) = a(n - 1) + 3*a(n - 2)+n^2, a(n));
                                                      2
                  rec(a(n) = a(n - 1) + 3 a(n - 2) + n , a(n))
   >> solve(rr3);
                             {           2         }
                             {   14 n   n          }
                             { - ---- - -- - 59/27 }
                             {    9     3          }
```

Graphics

Graphical display in MuPAD is produced by means of a separate program called **VCam**, which can be used entirely independent of MuPAD. But let's suppose we are in the middle of a MuPAD session (started with xmupad), and we wish to plot a few functions. The command **plotfunc** will allow us to plot functions of the form y=f(x), or in three dimensions, of the form z=f(x,y).

```
   >> plotfunc(1/(1+x^2),x=-4..4);
   >> plotfunc(x^3-3*x*y^2,x=-2..2,y=-2..2);
```

Such commands will open up a VCam window with the function displayed in it, and also another window in which it is possible to change various plot options. The output of this second command is shown in Figure 2, along with various of VCams's windows.

Figure 2. A Screenshot of VCam in Action

More general plot commands are **plot2d** and **plot3d**, by which the above plots can be generated as follows:

```
>> plot2d(Mode = Curve, [u, 1/(1+u^2)], u = [-4, 4]]);
>> plot3d([Mode = Surface, [u, v, exp(-sqrt(u^2+v^2))], u = [-4, 4],
v = [-4, 4]]);
```

We see that these commands use a parametric representation of the function. This allows for the easy plotting of a large number of different functions. There are a large number of plot options, including colours of plots and backgrounds, style of curves or surfaces, position and style of axes, labels and titles. All these can be changed interactively in VCam, or given as options to the plot command.

For a more complex example, let's plot a torus, with radii 3 and 1. This can be plotted with:

```
>> plot3d([Mode = Surface,
          [(3+cos(u))*cos(v), (3+cos(u))*sin(v), sin(u)],
          u = [0, 2*PI], v = [0, 2*PI]]);
```

This will give a wireframe torus with three numbered axes—not a particularly elegant result. We can add more options to this basic command to give us a nice filled-in torus with hidden lines, nestling in a box with no numbers:

```
>> plot3d(Title = "A torus", Axes = Box, Labels = ["","",""], Ticks = 0,
          [Mode = Surface,
          [(3+cos(u))*cos(v), (3+cos(u))*sin(v), sin(u)],
          u = [0, 2*PI], v = [0, 2*PI]
          Style = [ColorPatches, AndMesh]]);
```

All these options can now be changed again with VCam. We can also change the perspective; zoom in and out; modify the colour scheme; and many other things. We can draw several surfaces at once, by placing multiple surface definitions in the one **plot3d** command. For example:

```
>> plot3d(Title = "Two tori", Axes = Box, Labels = ["","",""], Ticks = 0,
          [Mode = Surface,
             [(5+cos(u))*cos(v), (5+cos(u))*sin(v), sin(u)],
             u = [0, 2*PI], v = [0, 2*PI],
             Style = [ColorPatches, AndMesh]
          ],
          [Mode = Surface,
             [sin(u), 5+(5+cos(u))*cos(v), (5+cos(u))*sin(v)],
             u = [0, 2*PI], v = [0, 2*PI],
             Style = [ColorPatches, AndMesh]
          ]
          );
```

The result of this is shown in Figure 3.

Figure 3. Two Interlocking Tori

When dealing with such long expressions as this, it is convenient to first use your favorite editor to create a small file, say "**tori.mu**", to contain the above command, then read it into MuPAD:

```
>> read("tori.mu");
```

This obviates using MuPAD's own text editing facilities, which are very basic.

Domains and Programming

Domains

We have loosely tossed about the term “domain”. We shall now look at this in a bit (but not much) more detail. Domains are fundamental to the way in which MuPAD works, and we need to have a basic understanding of them in order to use MuPAD effectively.

A domain in MuPAD is either an algebraic structure (such as finite field or permutation group) or a data type (such as Matrix, Polynomial or Fraction), for which overloaded operators or functions defined on that domain always return results in the domain (or the result FAIL if no result exists).

To give some examples, suppose we investigate matrices over the integers modulo 29. Since 29 is prime, these integers form a Galois field, and so the matrices should respond to all standard arithmetic operations.

First the definition:

```
>> M29:=Dom::Matrix(Dom::IntegerMod(29));
                     Dom::Matrix(Dom::IntegerMod(29))
```

We have two domains being used here: **Dom::Matrix**, which creates a matrix domain, and **Dom::IntegerMod(29)**, which creates the field of integers modulo 29.

```
>> A:=M29([[100,200,-30],[47,-97,130],[13,33,-1001]]);
                      +-                                  -+
                      |  13 mod 29, 26 mod 29, 28 mod 29  |
                      |                                    |
                      |  18 mod 29, 19 mod 29, 14 mod 29  |
                      |                                    |
                      |  13 mod 29,  4 mod 29, 14 mod 29  |
                      +-                                  -+
```

Notice that the result returned by MuPAD is automatically normalized so that the matrix elements are in the required field. If we enter values which can't be normalized (say, decimal fractions), MuPAD will return an error message.

```
>> 1/A;
                      +-                                  -+
                      |   3 mod 29,  8 mod 29, 15 mod 29  |
                      |                                    |
                      |  28 mod 29,  9 mod 29, 22 mod 29  |
                      |                                    |
                      |  12 mod 29, 19 mod 29, 13 mod 29  |
                      +-                                  -+
```

Here the inverse operator returns a suitable result. Let's check this.

```
>> %*A;
                    +-                                -+
                    |  1 mod 29, 0 mod 29, 0 mod 29  |
                    |                                |
                    |  0 mod 29, 1 mod 29, 0 mod 29  |
                    |                                |
                    |  0 mod 29, 0 mod 29, 1 mod 29  |
                    +-                                -+
```

This is the identity for our particular matrix ring. Now we can try a few other matrix operations.

```
>> linalg::det(A);
                                 12 mod 29
>> linalg::gaussElim(A);
                   +-                                   -+
                   |  13 mod 29, 26 mod 29, 28 mod 29  |
                   |                                   |
                   |   0 mod 29, 12 mod 29,  2 mod 29  |
                   |                                   |
                   |   0 mod 29,  0 mod 29,  9 mod 29  |
                   +-                                   -+
>> linalg::gaussJordan(A);
                    +-                                -+
                    |  1 mod 29, 0 mod 29, 0 mod 29  |
                    |                                |
                    |  0 mod 29, 1 mod 29, 0 mod 29  |
                    |                                |
                    |  0 mod 29, 0 mod 29, 1 mod 29  |
                    +-                                -+
>> A^10;
                   +-                                   -+
                   |  22 mod 29, 10 mod 29,  3 mod 29  |
                   |                                   |
                   |   3 mod 29, 21 mod 29, 16 mod 29  |
                   |                                   |
                   |   4 mod 29, 18 mod 29,  5 mod 29  |
                   +-                                   -+
>> exp(A);
                                   FAIL
```

The matrix exponential exp(X) is defined as 1 + X + (X^2)/2 + (X^3)/6 + (X^4)/24 + . . . + (X^n)/n! + . . . As you might expect, this is not defined for matrices over our field. For another example, consider polynomials over the integers modulo 2. The definition is similar to the matrix definition above.

```
>> PK:=Dom::Polynomial(Dom::IntegerMod(2));
                    Dom::Polynomial(Dom::IntegerMod(2))
```

Now we'll create a polynomial in this domain.

```
>> p1:=PK(x^17+1);
                                   17
                                  x   + 1
```

For good measure, we'll create a second polynomial which looks the same, but is not in our domain.

```
>> p2:=x^17+1;
                                   17
                                  x   + 1
```

Even though they look the same on the screen, MuPAD knows all about them; the type command will tell us.

```
>> type(p1);
                    Dom::Polynomial(Dom::IntegerMod(2))
>> type(p2);
                                  "_plus"
```

(The result of this last command is that **p2** is an object formed by adding things together.)

```
>> Factor(p1);
                3    4    5    8            2    4    6    7    8
    1 (x + 1) (x   + x   + x   + x   + 1) (x + x   + x   + x   + x   + x   + 1)
>> Factor(p2);
          2        3    4    5    6    7    8    9    10    11    12    13
(x + 1) (x  - x - x  + x  - x  + x  - x  + x  - x  + x   - x   + x   - x
      14    15    16
   + x   - x   + x   + 1)
```

The **domains** package is part of MuPAD which is very much in a state of constant revision and enhancement. For example, at present, it is not possible to perform polynomial division in a polynomial domain.

Programming

MuPAD comes with a very full-featured programming language. So much so, that I would say that this is one of MuPAD's greatest strengths. In contradistinction to other CASs, MuPAD allows:

- procedural programming
- functional programming
- object oriented programming
- parallel processing

It would take far more than an introductory exposition such as this to even scratch the surface of MuPAD's programming power, so we shall just look at a few simple examples. For more involved examples, try the **expose** command, or look at some of the files in your **$MuPAD/share/examples** directory.

For procedural programming, MuPAD offers all the standard programming requirements: loops of various sorts, branching, and recursion. Here, for example, is a simple procedure designed to implement Hofstadter's chaotic function, with line numbers as shown:

```
1   q:=proc(n) option remember;
2   begin
3   if n<=2 then
4       1
5   else
6       q(n-q(n-1))+q(n-q(n-2))
7   end_if;
8   end_proc;
```

Suppose we create a small file called **Hofstadter.mu** containing these lines, and we read it into MuPAD with

```
>> read("Hofstadter.mu");
```

This will make the function **q(n)** available to us. We can then list the first 20 values:

```
>> q(i)$i=1..20;
      1, 1, 2, 3, 3, 4, 5, 5, 6, 6, 6, 8, 8, 8, 10, 9, 10, 11, 11, 12
```

Or we can graph the first ten thousand values:

```
>> plot2d([Mode = List,[point(u,q(u)) $ u=1..10000]]);
```

A few points about this simple procedure: the **option remember** in line 1 means that values generated by the function are automatically stored where they can be retrieved if needed. With all the nested recursion, evaluating function values without using previous values would be hideously slow. The syntax is pretty standard, and is very similar to that of Maple.

We can define simple functions very readily by using an “arrow” definition:

```
>> f:=x->x^2+1;
```

Curiously, I can't find any reference to this in the manual.

For a slightly more involved example, suppose we try to write a program to generate truth tables. That is, so that something like

```
>> tt(p and q);
```

will produce

```
p, q, p and q
T, T, T
T, F, F
F, T, F
F, F, F
```

To make things easy for ourselves, we shall include the variables of the boolean expression in the function call, so as to save the bother of parsing the expression to obtain its variables. And here is such a program:

```
tt:=proc()
local i,j,n,l,ft,f,r,rf;
begin
    if args(0) <= 1 then error("Must include all variables");
    end_if;

    ft:=[FALSE,TRUE];
    n:=args(0)-1;
    print(Unquoted,args(j)$j=2..args(0),expr2text(args(1)));

    for j from 2^n-1 downto 0 do
       f:=args(1);

       for i from 1 to n do
          l[i]:=floor(j/(2^(n-i))) mod 2;
       end_for;

       for i from 1 to n do
```

```
19            f:=subs(f,args(i+1)=ft[l[i]+1]);
20         end_for;
21
22         for i from 1 to n do
23            r[i]:=_if(l[i]=0,"F","T");
24         end_for;
25
26         rf:=_if(simplify(f,logic),"T","F");
27
28         print(Unquoted,r[i]$hold(i)=1..n,rf);
29      end_for;
30  end_proc:
```

Some points here: the value **args(0)** in lines 4 and 8 gives the number of arguments; the **expr2text** function in line 9 simply returns a string version of a given expression; the **error** statement in line 4 immediately exits the procedure; the **_if** command (lines 23 and 26) is a functional form of the standard if statement. The rest of the program simply lists all possible truth values for the variables (this is done in lines 14 to 20), and prints them out, along with the value obtained when those truth values are substituted into the function.

This program may be considered as skeletal only; for example, it does no type checking.

Learning about MuPAD

There are several ways of obtaining information about MuPAD: through the extensive on-line help facility, or from published articles.

Online Documentation

A few MuPAD books are currently available, but I haven't seen one. The on-line documentation is excellent. It exists in two forms: plain ASCII (for use with MuPAD in terminal mode); and dvi with hyperlinks, when using MuPAD under X.

For ASCII, the help commands are obtained with a question mark:

```
>> ?<command>
```

will provide a few screens of information about **<command>**. Notice that this is the only MuPAD command which doesn't require a terminating colon or semicolon. The amount of documentation varies; but usually includes a basic description, some examples, and a list of related commands. Some help files are very large indeed; others very small. For example

```
>> ?stats::median
```

is short; it basically just tells you that this command returns the median of a list of values. However

```
>> ?Dom::Matrix
```

is very long, with a detailed discussion about creation of matrices, and operations defined for matrices.

Information about a MuPAD library can be obtained with the **info** command; for example

```
>> info(numlib);
```

provides a list of all commands in the **numlib** library. You can also use

```
>> ?numlib
```

to obtain a brief description of all of **numlib**'s commands.

You can see the MuPAD programs which define a function with the use of **expose**. Again, programs vary greatly in length. For example

```
>> expose(length);
```

returns a short function defined by a six-line program, but

```
>> expose(D)
```

returns a 164-line program. You can also use **op** to obtain information; again the amount given varies;

```
>> op(Dom::Matrix);
```

returns all 1482 lines of the MuPAD program defining matrices.

The documentation for xmupad is provided in the form of a dvi file of the manual, with lots of hyperlinks. The manual can actually be read independently of MuPAD with the **hypage** command. As with the commands for starting MuPAD, this is a call to a shell script. However, hypage is started automatically with xmupad.

All the relevant documentation is provided in the form of hyTeX dvi files; that is, dvi files with embedded hyperlinks. If you are used to TeX and dvi viewers, this approach will seem very sensible: the mathematics is superbly typeset, as you'd expect, and there are plenty of included graphics. It may seem a bit odd at first to use a book (with all the trimmings: table of contents, chapters, an index) as the paradigm for on-line documentation, but in fact it works very well. And after all, everybody is familiar with the structure of a book, so there is no need to learn the particular exigencies and peculiarities of some new and strange help system.

The amount of information is superb. As well as discussions on all commands and functions, there are extensive chapters on graphics, programming, and debugging, as well as two excellent demos. Moreover, as well as the manual, there are hytex dvi files covering all the MuPAD libraries, as well a handy quick reference.

Once xmupad and hypage are up and running, the help command

```
>> ?<command>
```

will open up the manual at the relevant page. Once inside the manual, you can use the hyperlinks to travel through other similar commands. I find hypage very easy to use. If you have installed MuPAD, you should find this entire article redundant, as everything in it is covered in far greater depth in the manual.

A screen shot of hypage is shown in Figure 4.

Figure 4. Hypage—MuPAD's Graphics Hypertext Manual

However, I have a few niggles with hypage. The main one is that there is no way of changing the font size or page layout. If you reduce the size of the hypage window, for example, the text is not reformatted to fit the available window size, and so you have to use inconvenient scroll bars to view the text. This makes hypage nearly unworkable on a 640x480 VGA screen (such as on my laptop). Moreover, all the text is in the same colour (hyperlinks are given by underlines), which I find a bit dull. There also doesn't seem to be much in the way of keystroke movements. I found three (they aren't listed in the manual): p for previous page; f for forward, and r for return (to the linking page).

On the other hand, with a higher resolution screen, hypage is a delight to use.

Other Documentation

The canonical MuPAD site is

www.mupad.de

but most information is at

www.sciface.com

to which you are directed from the MuPAD site. Here you will find a few FAQs, some examples of MuPAD in different areas of mathematics, and some comparisons with other CASs. Unfortunately these comparisons are fairly old, and at the time of writing they have not been upgraded to the latest version of

MuPAD. There are also a few technical articles written by members of the MuPAD team discussing aspects of MuPAD internal workings. These articles should be read if you wish to master MuPAD.

You will also find some MathPAD journals. These are published by the MuPAD team, and cover CASs and their uses in general. Only a few of them are devoted to MuPAD, and these few are well worth reading, as they give fascinating insight into the development of MuPAD, and the reasons for various design considerations.

As yet, there are no books on MuPAD, aside from the published manual, and few articles in the open literature. The two sites above, and the on-line documentation, pretty much cover the lot.

Installation of MuPAD

Couldn't be easier! You need to download two files: one containing the binaries (mupad, xmupad, vcam, hypage and such); and the other the “shared” material: the documentation and MuPAD libraries. Uncompress them, and untar them in a suitable directory (/usr/local/MuPAD is a good spot, but anywhere will do), and add /usr/local/MuPAD/share/bin (or whatever) to your path. As I mentioned before, this bin directory contains only shell scripts, which set all necessary environment variables before calling the binaries. Thus there is no need for any extra fiddling.

To use the graphical capabilities of MuPAD, you also need the xview libraries installed on your system.

You will find that MuPAD, thus installed, has certain memory restrictions: you will be unable to deal with commands requiring intensive memory use. To overcome this, you need to register your copy of MuPAD. This will provide you with a license key which you can use to unlock the MuPAD's memory use.

Comparisons, Conclusions, and Other “C” Words…

One of the most commonly asked questions to the sci.math.symbolic newsgroup is: “What's the difference between...?” The best answer I've seen so far is: “They're spelled differently!” In that spirit, I intend not to give a detailed anaysis of the differences between MuPAD and other CASs, but instead to make a few general points.

First, MuPAD can't be beaten on price: it's free! Strictly speaking, this is not true; there are circumstances for which you must pay to use MuPAD. But for a single-user Linux user (and which of us is not?), it costs nothing.

The interface is pretty clumsy compared with the lovely worksheet interface of Maple and Mathematica. The interface for MuPAD under MS Windows 95/NT is much more refined, but you have to pay for it. For this reason alone, I would say that MuPAD is not as well suited as its rivals for elementary teaching. It's nice to see input and output in different colours, and for mathematics output to be presented properly typeset. I also like having graphical output as part of the worksheet. Currently, none of these are possible in MuPAD.

Again, MuPAD does not have the mathematical breadth of its rivals. An example will illustrate my point: the attempt to solve the Riccati differential equation dy 2 2 -- = x + y , y(0) = 1. dx Maple can solve this with no problem; the solution is a complicated expression involving Bessel functions. However, MuPAD can't solve this; it doesn't know enough about Bessel functions to recognize all possible places where they may apply. Also, its **solve** command, as applied to differential equations, does not have the power of Maple's **dsolve** command. However, MuPAD is a much younger product, and wheras both Maple and Mathematica are produced by large companies with enviable resources, MuPAD is produced by one small and chronically underfunded research team.

On the flip side, MuPAD is as extendible as its rivals. Its programming power is easily equal to that of Maple and Mathematica. What's more, it provides different programming paradigms, and doesn't force you into any particular style. Again, it offers full parallel programming, and comes with an excellent graphical debugger. To do these topics justice would require at least another article the same size as this one, so get MuPAD and read its documentation!

The use of domains in MuPAD means that it is possible to explore deep aspects of mathematics, and to write very general routines. Thus MuPAD has a depth equal to or greater than its rivals, even if it loses out on breadth.

So why use MuPAD? If you are after a CAS to be used as a “black box” to churn out solutions to equations; then MuPAD is not as suitable as its rivals (at least, not yet). However, if you are exploring mathematical relationships and structures, then MuPAD would appear to be the tool of choice.

I am extremely impressed with MuPAD; free software of this excellence is produced very rarely indeed. MuPAD deserves the full support of the Linux community, and if you use mathematics in any way, then MuPAD should find a home on your system.

Alasdair McAndrew lives in Melbourne, Australia, with his wife, three young children and a grumpy cat. He is a Senior Lecturer at Victoria University of Technology, where he teaches mathematics and computing. He is an enthusiastic and satisfied user of Linux, and has been since kernel 0.99; currently he is running Linux on both a desktop and a laptop. He enjoys trawling the Internet for Linux software suitable for children, and when he has time, playing the viola da gamba.

Archive Index Issue Table of Contents

Advanced search

Introduction to Sybase, Part 2

Jay Sissom

Issue #63, July 1999

This month Mr. Sissom shows us how to set up and use a Sybase client written in Perl through examples.

Last month, we installed a Sybase database server. This month, we will install a client to access our server. First, we need to understand how the Sybase network interface works.

A Sybase client must create a network connection to a database server when it needs to access resources in the database (see Figure 1). Sybase has created a protocol to communicate over the network. This protocol is called the Tabular Data Stream (TDS) protocol. It operates on top of other networking protocols such as TCP/IP on UNIX systems or IPX/SPX on Novell networks. TDS is a proprietary protocol and not documented by Sybase. Fortunately, Sybase has created client libraries which can be used to communicate with the server. A group of people have tried to reverse-engineer the TDS protocol. Look at http://sunsite.unc.edu/freetds/ for more information.

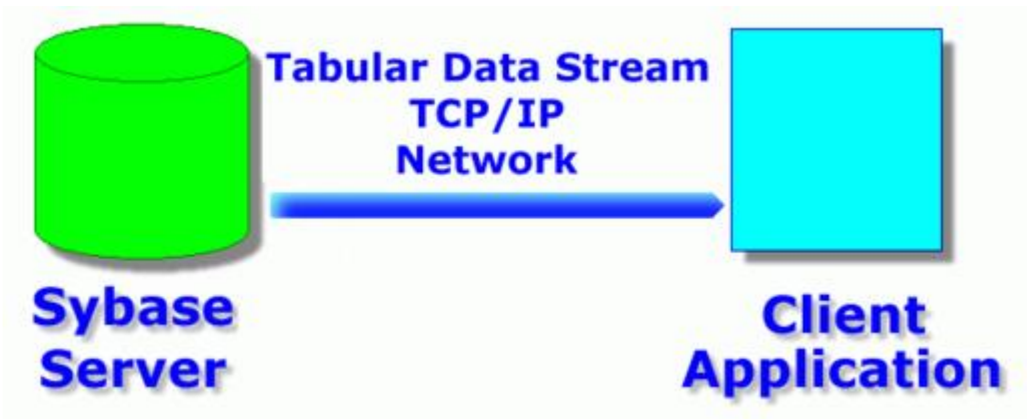


Figure 1. TDS Protocol

Sybase supports two interfaces to the database. DB-Library is an API that has been used for quite awhile in Sybase products. I believe it is supported for backwards compatibility, and may not be supported in a future version of the product. CT-Library is the API Sybase created for version 10 and higher products. It supports advanced features such as cursors and asynchronous query processing. You don't need to understand these features to do basic processing with your database server. We will use CT-Library to communicate with our server.

We could write our client using C or C++. The libraries required to do this are included with the server. Look for examples in the sample directory under the server directory. There is a subdirectory for DB-Library and one for CT-Library. We don't have to use C or C++, however. An extension to the Perl language called **sybperl** enables the use of Perl to write clients to access the database.

Most Linux distributions come with the Perl language. On my system, I have installed Red Hat 5.1 which includes Perl version 5 by default. Fortunately, it is possible to install sybperl without recompiling Perl. Using this method precludes the use of the DB-Library, which is why we have chosen to use CT-Library.

If Perl is not installed on your system, install it now. If your distribution does not provide perl, you can download the source from CPAN (http://www.cpan.org/).

Install sybperl

First, we must download sybperl from http://www.perl.com/CPAN-local/authors/Michael_Peppler/

The newest version available at the time of this writing is version 2.09_05 in the file sybperl-2.09_05.tar.gz (148KB). Change directories to the location of the sybperl tar file, and issue the command:

```
perl Makefile.PL
```

Change to the sybperl directory just created, and edit the CONFIG file. In the line **DBLIBVS=1000**, change the 1000 to 0. Make sure the line **SYBASE=/opt/sybase** contains the correct directory for the Sybase server. The line **EXTRA_LIBS=-ltli** must be changed to **EXTRA_LIBS=-linsck**.

We will build sybperl to work with CT-library. Most Linux distributions come with the Berkeley DB library. If Perl is configured to use this library, a conflict arises when using DB-library at the same time, since both use the call **open_database**. If you recompile Perl to leave out the Berkeley DB library, you can leave the line **DBLIBVS=1000** in the CONFIG file and use DB-library.

Save the changes to the CONFIG file, then issue this command:

```
perl Makefile.PL
```

This will create a file that will build the software. It looks for the Perl installation in your path. If Perl isn't in your path, you'll need to change your path to include it. Now issue the **make** command to build the software; it will take a few minutes to run. **sybperl** has tests that can be run to ensure it is built properly. To run these tests, edit the PWD file to put your **sa** password and the name of your Sybase server on the proper lines. If you installed the server following the directions in the last issue, the name of the server is linux_dev. Save the file, then type the command

```
make test
```

This command will run a series of tests. If everything is working properly, the message “All tests successful” will be printed.

Now, let's install sybperl. If your Perl installation is in a directory that requires root access to modify, change to root using **su**. Run the command

```
make install
```

Perl and sybperl are now installed, so it is time to write some programs.

Writing a sybperl Program

All of the programs here are available for download. If you type in these programs, be sure to use **chmod** to make them executable.

Listing 1.

Writing a sybperl program is quite simple. Listing 1 is our first example program. This will list the names of all the databases in the server. Here is a line-by-line explanation of the program.

Line 1 tells Linux which program to use to run this script. This must be the new version of Perl you just installed. Make sure you change this line to point to the correct version of Perl on your system.

Line 3 tells Perl to use the CT-library interface to Sybase. It should be at the beginning of all Perl scripts you write that access a Sybase server.

Line 5 attaches to the correct Sybase server. The first parameter is the user name, the second is the password and the third is the name of the server.

Line 7 is the SQL to run.

Lines 9-10 are commands that run the query on the server and return a reference to an array of rows, **@rows**. Note this command loads the entire result set into memory. This is fine for small result sets, but if you are expecting a large result set, you shouldn't use **ct_sql**. Later, I will give you an alternative method for executing commands and receiving large result sets.

Lines 12-14 will print all rows that were returned.

Listing 2

Listing 2 is an example of a sybperl program that updates data. In Line 7, we use the same ct_sql command to send the SQL to the database, except this time a set of rows is not returned. The **insert**, **delete** and **update** SQL commands also do not return rows. The SQL command **use pubs2** tells Sybase to make the pubs2 database the default database for the rest of this session. In Line 10, we again use the ct_sql command to run the SQL. This time, we add a row to the discounts table. You can use the **isql** program to run an **SQL SELECT** command to verify that the row was added.

Writing a Perl CGI Client Program

Linux is mostly used as a web server, and Perl is primarily used to write web applications. So, we will create a Perl program to access the Sybase database.

Listing 3

Writing a CGI program to access Sybase is quite simple. Listing 3 is the complete code of a CGI program to let you know who's logged in to your Sybase server. Place this program in your web server's cgi-bin directory. On a default Red Hat system, the directory is /home/httpd/cgi-bin/. For this example, name the program listing3.pl.

In lines 5 and 6, we set two environment variables. The Sybase DB-Library and CT-Library must find these environment variables, or an error will occur. When you run a CGI program, very few environment variables are passed to your program. These two environment variables must be set in each CGI program that needs to access Sybase. If you have many CGI programs, place these commands in a file included in all your CGI programs.

The **SYBASE** environment variable contains the directory of the Sybase software. The **DSQUERY** variable contains the name of the default server.

The only other difference between this example and the others is it outputs HTML to a browser.

Other Considerations

These example programs show the basics of accessing a Sybase database server. In production programs, a few more things must be taken care of in your programs.

Errors from the server must be handled properly. If you ignore them, your program will stop when it encounters a server error.

In all our example programs, ct_sql was used to run queries. It works fine for SQL commands and stored procedures which don't return result sets, but would have severe problems for queries returning large result sets.

Listing 4

Listing 4 shows how to handle errors and demonstrate a replacement for the ct_sql command. In lines 3 and 4, we establish both a client and a server message call-back routine. These routines will be called when the server or client generates an informational or error message.

In lines 7-20, we handle a single SQL statement. Sybase allows a single statement to return multiple result sets. Lines 8-10 will process each result set. Lines 17-19 will handle each row in a result set. Lines 11-16 will look at the result set and print the names and types of each column. A Sybase result set contains more than just data—it also includes column definition information.

Lines 23-50 are the two call-back routines. These routines are called each time there is a message from the server or client. An example of a client message is the one returned if you can't log in to the server. An example of a server message is the one returned if you have an error in your SQL.

All this information can be found within the sybperl and the Sybase documentation.

Conclusion

The Sybase database server is a powerful piece of software. Unfortunately, all that power comes with a price. Setting up and supporting a Sybase server isn't as easy as using Postgres or MySQL, but if you need a heavy-duty, high-performance database, Sybase is your best bet.

Next month, we'll discuss application development using Sybase on Linux. This article will appear in the “Strictly On-line” section.

Jay Sissom is responsible for the web-based front end to the financial decision support data at Indiana University. He has installed and supported Sybase databases on many operating systems and has written database clients for the Web using C, C++ and sybperl and for Windows using tools like Visual Basic and PowerBuilder. When he isn't programming, he enjoys amateur radio and playing his bass guitar and keyboards. If you have questions, you can contact him via e-mail at jsissom@indiana.edu.

Archive Index Issue Table of Contents

Advanced search

Precision Farming and Linux: An Expose

Gordon Haverland

Issue #63, July 1999

Farming is not a place one would expect to fine Linux, but there it is. Mr. Haverland tells us how Linux is used in this unusual area.

Ideally, precision farming is a process where one manages the production of a crop on a plant-by-plant basis. Practically speaking, we are not yet at that level of resolution, but we are beyond treating an entire field as one homogeneous unit.

Modern electronics, global positioning satellites, signal processing and a host of other technologies make precision farming possible. The driving forces for development and acceptance are the possibility of increased profitability and the realization of improved stewardship of the land (less pollution, optimal use of chemical inputs, etc.).

Introduction

Precision farming is evolving around three technologies: global positioning (GPS), geostatistics (GIS) and remote sensing. Other developed technologies such as sampling, soil analysis and others are also playing a role.

Alberta Agriculture, Food and Rural Development has been in the Precision Farming business since 1993. We work with selected farmers across the province who grow a variety of crops: wheat, barley, canola, peas, potatoes, etc. Our objectives are the following:

- Help develop a workable system that interested farmers can implement on their farms.
- Transfer this technology to private industry.

Global Positioning

Global positioning satellites and receivers were the technologies that started this revolution in agriculture (see Resources). Before then, positioning via radio was possible, but not very accessible to the researcher, or ultimately the farmer.

Two families of global positioning satellites are in orbit around the earth: Navstar (U.S. system) and Glonass (Russian). The Navstar system is capable of better positioning than the Glonass system. However, the US military uses something called “selective availability” to decrease precision to something on the order of 100 meters (unless one is using military GPS receivers). Agricultural applications require positioning at sub-meter precisions, most often in real time.

Some enterprising person realized if a GPS receiver were put over some known position, a correction to the position as determined by GPS could be calculated. This correction factor could then be applied to nearby (in space and time) GPS positions. How good the correction is depends on how close the roving receiver is to the fixed known station, and how long ago the correction was calculated. The best precision I am familiar with is typically on the order of 10cm. However, due to the changing “constellation” of GPS satellites, we can't always get precision this good.

For most people, it is not practical to own two GPS receivers plus radio-modems to gather positioning information. For 20cm precision, this type of setup costs on the order of $20,000 US. Fortunately, it is possible for many people to obtain usable differential corrections without going to the trouble of owning a differential base station. There are a number of sources of differential corrections and means of receiving them. Some corrections are broadcast over FM radio, some on other frequencies and some from satellite. The Coast Guard has a number of DGPS (differential GPS) beacons along navigable waterways and coastlines. This includes locations such as the Mississippi and Missouri River basins and the Great Lakes.

Geostatistics

One of the most important factors in precision farming, is the ability to store, display and manipulate geo-referenced information. On a local scale, geostatistical packages can look something like a spreadsheet (raster) or a CAD (vector) package. Some types of information access work better in the raster model, some in the vector model.

Agriculture is one of the smaller users of GIS packages; therefore, it is necessary to bend the analytical techniques to methods developed for other applications. The unavailability of theoretical models also affects what can be done with an analysis.

A couple of “free” GIS packages are available, but by far the most comprehensive is the Geographical Resources Analysis Support System (GRASS) (http://www.baylor.edu/grass/). This package was originally developed by the U.S. Army Corps of Engineers Construction Engineering Research Laboratory (CERL) and extended by other government and university researchers and users over the years. Active support by CERL has been abandoned, but was recently taken up by Baylor University.

GRASS is now advertised as an “Open GIS”. Among the new developments is an interface based on Tcl. An OpenGIS Consortium (http://www.OpenGIS.org/) also exists.

Remote Sensing

Commercial satellite remote sensing (see Resources) typically has a resolution on the order of 10m, each pixel covering an area on the order of 100m (far worse than the (rumored?) 10cm resolution the CIA has with some of its satellites). In order to statistically detect some change in an image, many pixels in close proximity must be significantly different than expected. With 10m pixels, this means that many hundreds of square meters of land must be affected before the change can be detected from space. For things like disease or pests, this is not very useful, but it is acceptable for crop maturity. To detect disease or pests, aircraft-based remote sensing is presently required. In the near future, high-resolution satellite imagery is expected to become commonplace.

Adventures in Analysis

Precision farming is filled with analytical adventure. Conventionally, precision farming starts with a map of crop yield from the field. To acquire this map, we put a DGPS receiver on some known location on the combine harvester, and a yield monitor somewhere in the path grain takes from entering the header to the clean grain tank. (The closer this is to the threshing part of the combine, the shorter the throughput delay time.)

Position Information

The DGPS antenna is fixed on some point on the combine harvester, usually the cab roof. The grain is removed from the field along the leading edge of the header, i.e., a line segment remote from the point where the GPS antenna is. If we are interested in only coarse resolutions, the difference between the GPS receiver location on the leading edge of the combine header is not important.

At fine resolutions, we do need to make this correction, and in order to make it, we need to know the orientation of the vehicle in 3-D space. This information is not currently collected—it must be calculated later.

Velocity Information

Combine harvesters typically move at about two meters per second. With high accuracy DGPS equipment, errors are typically on the order of 10cm. This results in velocities having errors on the order of 10%. GPS position errors are highly correlated in time. Events called “blunders” can occur, which result in relatively huge position errors. Once a blunder has occurred, its presence may live on for several seconds. This results in two huge velocity errors on either end of a “short” track of biased position estimates.

Combine harvesters usually operate under the conditions of constant mass and wheel power traveling in straight lines. (This means the position as a function of time must be twice continuously differentiable. The only places where third and higher order derivatives can exist are on corners. This information can be used to help smooth the position information.) Therefore, we should be able to use low-order polynomials or splines to smooth the position as a function of time data.

Digital Elevation Models (DEMs)

If one assumes the vehicle is always traveling in a forward direction and there is no side slip of the wheels, it is possible to calculate the orientation of the vehicle in space. However, if one has an available model of the elevation as a function of position in the field, they can obtain a much better estimate of the surface normal to the ground.

Stereo photography is one way to obtain a DEM. Another is to start from GPS position information and correct it for the GPS receiver mounting position and the vehicle's geometry and orientation in space.

The DEM is quite useful, as landscape is one of the contributing factors in crop yields. For example, during a wet year, low-lying concave parts of the field are typically too wet to produce good crops. Another example is lower yields on high-elevation convex hilltops during dry years. The DEM is most useful when it is interpreted with terrain analysis and combined with crop yield and other data sets in a GIS. Sometimes (more often than desired) precision farming requires new and specialized procedures.

Yield Time Lags

Basically, the time it takes for the crop to travel from the leading edge of the combine header to the yield sensor is not a single, unique value. Some of the crop takes a relatively short time, and some takes longer. This process smears and averages crop yield over a period which is typically 25 seconds long.

Linux and Open Source Software

Linux is a good platform for doing this research-oriented work. Much of the analysis can be translated into such mainstream topics as signal processing or multi-dimensional statistics. Some of the best software for exploring software in these topics is the product of government and university research and is "free"--an important quality in tight budgets. GRASS, xldlas and Santis are three packages which have helped in precision farming.

Perl has also proven itself to be quite useful. Our participation in precision farming has lasted six years now. Several kinds of GPS equipment and yield monitors have been used, even updates of individual systems. Of course, everybody has to have his own "standard" data format. Our method of analyzing the data evolves, which for us means re-analyzing the previous year's data with the current best method. This results in many format changes in data—an area where Perl excels. The ability to use pseudo-ttys in Perl has been useful when it was necessary to change coordinate systems on thousands of data points using GRASS programs.

One area which has not seen a lot of development is the safe storage of data. A farmer does not want to discover that the data is not "on the card" *after* he has harvested a crop—this has been known to happen. Using radio modems to transmit the data to a home computer would be one way of avoiding this situation.

Summary

Precision farming is here to stay. However, the data storage and analysis needs of precision farming are beyond the resources of most farmers. Procedures and analytical processes are not always available in "canned" packages (certainly not in any one package), so a powerful, open development environment is needed.

If you are a person who understands the philosophy of statistics and are academically inclined, precision farming might be a very interesting topic/ career in which to get involved.

Resources

Gordon Haverland works for the Precision Farming Project of Alberta Agriculture, Food and Rural Development. He can be reached via e-mail at haverlan@agric.gov.ab.ca.

Archive Index Issue Table of Contents

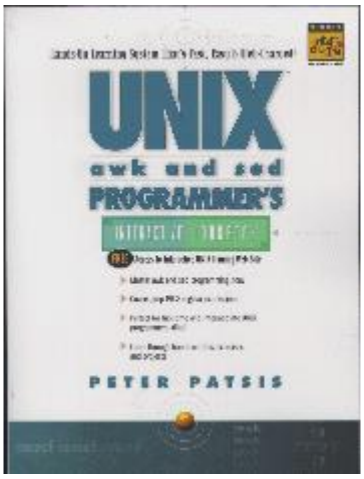
Advanced search

Book Review: UNIX awk and sed Programmer's Interactive Workbook

Paul Dunne

Issue #63, July 1999

Despite that mouthful of a title, this is not a book about UNIX.



- Author: Peter Patsis
- Publisher: Prentice-Hall
- URL: http:www.prenhall.com/
- ISBN: 0-13-082675-8
- Price: $34.99
- Reviewer: Paul Dunne

Despite that mouthful of a title, this is *not* a book about UNIX. It is about three UNIX utilities, **grep**, **sed** and **awk**, each on their own, one after the other. Now, I learned these tools and others from the classic text, *The UNIX Programming Environment* by Kernighan and Pike. Although dated in parts, the treatment there crams a good deal of information into a short space and never loses sight of the fact that AWK, sed and similar tools are designed to be used in a UNIX environment, working with the shell and other UNIX tools. Ripping them out of context causes them to seem much less useful than they are. In this book, there is hardly a pipeline in sight. Not only that, we don't even see sed and AWK working together through temporary files. Each tool is considered in complete isolation. There is little point in learning about AWK and sed on their own. If one is really determined to do everything without the shell and other utilities, Perl is a proper superset of the two. AWK and sed come into their own only when considered as part of a toolset.

This book has too much padding. I don't know how many times we are told where the web site associated with the book is, but once would have been enough. I personally hate those little icons popular in books of this sort today. Each time our intrepid author mentions the web, we have a picture of a spider's web (I'm not making this up), each aside or technical tip (I don't really know what to call them—the only clue is the little icon beside each one) has a top hat affixed to it, and so on. I don't know whether this is all IDG's fault, but it was in their “For Dummies” series where I first encountered this style. There, while still objectionable to my mind (and don't get me started on said company referring to customers as “Dummies”!), it is at least well-thought-out and integrated into the book; here, it's just pointless. If I want pictures, I'll go read a comic book.

No mention is made of *which* AWK or sed or grep. For instance, one might be using GNU awk (gawk), nawk, mawk, MKS awk or any one of several vendor implementations, although hardly the original awk. These things matter. Are we talking strict POSIX sed and AWK here? We are not told. This topic needs to be addressed, however briefly, in any book dealing with these tools.

Is there anything I like about his book? Well, the exercises and self-review questions are frequent and, although often simple, do have the merit of being connected to the preceding subject matter, covering what you are supposed to have learned. They've been given consideration and aren't just tacked on as an afterthought.

This book might be useful for those who find the treatment in Kernighan & Pike too concise; but it needs to be supplemented, if not replaced, by a book which discusses these tools in their proper context. Personally, I find the right thing is just such a dual approach. Rely on a good learner's book, but have at hand a reference work which goes into much greater detail and can be consulted when problems or questions arise. Unfortunately, this book is neither. For instance, there is an “Awk, sed and grep” reference listed as an appendix. Turning to the appendices, we find a rather sparse reference for AWK only, with no mention of the other two.

It isn't that this is a bad book for what it does. However, what it does isn't enough and is aimed at the wrong market. The cover proclaims, “Perfect for first-time and intermediate UNIX programmers alike”. Very little is contained here for either first-time or intermediate UNIX programmers or users. It would be better suited to a Windows user who has the Cygnus Win32 toolkit and wants to know what to do with it. Even then, some coverage of the shell is almost essential. Realistically, the people who want a Win32 environment to look like UNIX are going to be people with UNIX experience, and this book is not for them. So what is the market? Well, that's the biggest problem with this book. Within its rather narrow remit, it does things well; but it doesn't have a well-defined audience. If you need to learn about sed and AWK, then the best resources remain Kernighan & Pike and O'Reilly's *UNIX Power Tools*.

I find it hard to recommend this book to anyone. It might possibly be useful, if supplemented by a general work on using the shell, for those who need a slow and easy introduction to sed and AWK.

Paul Dunne (paul.dunne@bigfoot.com) is an Irish writer and consultant who specalises in Linux. The only deadline he has ever met was the one for his very first article. His home page is at http:dunne.home.dhs.org/

Archive Index Issue Table of Contents

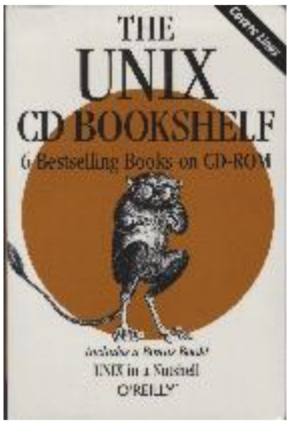
Advanced search

Book Review: The UNIX CD Bookshelf

Derek Vadala

Issue #63, July 1999

The CD-ROM contains a complete copy of *UNIX in a Nutshell: System V Edition*; *UNIX Power Tools, 2nd Edition*; *Learning the vi Editor, 5th Edition*; *Learning the UNIX Operating System, 4th Edition*; *sed and awk, 2nd Edition*; and *Learning the Korn Shell.*



- Publisher: O'Reilly & Associates, Inc.
- E-mail: info@ora.com
- URL: http://www.ora.com/
- Price: $69.95 US
- ISDN: 1-56592-406-1
- Reviewer: Derek Vadala

The UNIX CD Bookshelf, packaged by O'Reilly & Associates, Inc., marks the third installment in an emerging series by my favorite publisher. The CD-ROM contains a complete copy of *UNIX in a Nutshell: System V Edition*; *UNIX Power Tools, 2nd Edition*; *Learning the vi Editor, 5th Edition*; *Learning the UNIX Operating System*, *4th Edition*; *sed and awk*, *2nd Edition*; and *Learning the Korn Shell*. These electronic files are laid out in HTML format on the CD-ROM, making them easy to read, navigate, share across local area networks and travel.

O'Reilly has also included all the software from *UNIX Power Tools*, i.e., many useful utilities, shell scripts and software.

For those not familiar with these books, a brief summary follows.

- *UNIX in a Nutshell* is a desktop quick reference outlining the fundamentals of many core UNIX commands while touching on such topics as shell scripting, pattern matching and text processing and editing.
- *UNIX Power Tools* is a collection of articles containing tips and tricks for effective UNIX usage. Organized into more than fifty topics, *UPT* also includes useful scripts and oodles of pre-compiled open-source software.
- *Learning the vi Editor* covers the ins and outs of this popular editor and its pattern matching and text replacement functions.
- *Learning the UNIX Operating System* provides a swift introduction to the art of UNIX and includes tutorials on basic command use, redirection and process management.
- *sed and awk* focuses on the usage of these two powerful text processing utilities and provides practical examples of their use.
- *Learning the Korn Shell* introduces the fundamentals of the Korn shell as both a user environment and a scripting language, while expanding on many UNIX basics.

While the static pages of these books are exceedingly valuable, the true power of *The Bookshelf* is unleashed with the included search facility, which makes every book searchable, allowing users to quickly mine information from some of O'Reilly's top titles. Want to know how to copy files using **tar**? Merely enter **tar** and **copy** into the convenient index page, and the article "Copying Directory Trees with tar" from *UNIX Power Tools* pops up.

The included Java-powered search engine requires installation of the Java run-time environment (JRE), which is conveniently located on the CD. I did notice some minor glitches while using the Java search facility. The Java searcher proved unreliable in searching individual books, but worked flawlessly when milling through the collective titles. Buyers might also find it a tad frustrating that book titles are not returned as part of the results when searching the entire set, making it harder to quickly decide which article will be most relevant to the problem at hand.

The setup of the JRE is straightforward and should be workable by even a UNIX novice. If you cannot make heads or tails of the README file included with the tar archive file or the hassle-free install script, an HTML page (included on the CD and linked from the main index page) includes help for those who need a little push. JRE binaries are included for Linux (libc5 and glibc), Solaris (x86 and

Intel), FreeBSD, SunOS and HP/UX. A polite note on the help pages indicates that AIX, Digital UNIX and SCO UNIX binaries could not be included because of licensing issues, but a working link to more information on getting the JRE for any of these architectures, and many others, is provided. Power users might consider using an alternate method to search through the electronic texts. I was able to easily index the lot with **htdig**. Similar effects could be produced with many other free packages, making the Java annoyances no excuse for passing over this formidable ensemble.

As if the collective texts of these invaluable books coupled with a search facility was not enough, a print copy of *UNIX in a Nutshell* is also included. Unfortunately, it has not been revised for this release, other than some changes to its artwork and layout. While the included 1994 edition is still wonderfully helpful as a reference book, a more up-to-date revision would have been welcome. This will undoubtedly make many Linux users go running for man pages or a copy of *Linux in a Nutshell* instead. It would also have been nice to include the 6th edition of *Learning the vi Editor*, rather than the outdated 5th edition which does not include any of the long-awaited chapters on vi clones. GNU/Emacs users might also find it a tad offensive that they have been usurped here.

The UNIX CD Bookshelf is a must for any UNIX beginner who hasn't been previously exposed to the usefulness of these wonderful books. It makes for a verbose yet portable UNIX desk reference that will surely save UNIX users hours of time. At $69.95, it is also a bargain compared to the aggregate price of the individual titles. Expert users who own most of these titles might want to think twice before shelling out the cash for this bundle, although it would make a serious asset to any consultant on the road. The power of searching through these UNIX titles can save even veterans from many headaches. Looking down the “CD Bookshelf” road with future releases including *The Networking CD Bookshelf* and *The Perl CD Bookshelf*, I wonder when we'll see an O'Reilly compendium on DVD-ROM.

Derek Vadala (derek@soundwave.usfca.edu) lives in San Francisco and works as a UNIX consultant for Taos Mountain. He wishes more people would use Linux as it would make his life fairly hassle free. Send praise or flames to derek@usfca.edu.

Archive Index Issue Table of Contents

Advanced search