Linux Journal Issue #75/July 2000

Focus

Science & Engineering *by Marjorie Richardson*

Features

Gri: A Language for Scientific Illustration *by Dan E. Kelley and Peter S. Galbraith*

This scripting language avoids integrating analysis and display capabilities and instead focuses on providing precise and flexible control over the display of technical material.

Tracking Satellites with PREDICT *by John A. Magliacane*

A look at the development and use of an open-source satellite-tracking and orbital-prediction program.

Detecting Chaos in the Field *by Juergen Kahrs*

All that is real is reasonable, and all that is reasonable is real. — G.W.F. Hegel, 1770-1831

THOR: A Versatile Commodity Component of Supercomputer Development *by Robert A. Davis*

CERN continues to use Linux as their OS of choice for modeling and simulation studies.

A GNU/Linux Wristwatch Videophone *by Steve Mann*

This fully fuctioning prototype, designed and built by Steve Mann in 1998, was demonstrated in 1999, and later used to deliver a videoconference at ISSCC 2000.

Forum

Reviews

Columns

Focus: Science and Engineering

Marjorie Richardson

Issue #75, July 2000

I must say this issue ended up being more science than engineering, but perhaps that is not surprising considering my scientific leanings.

I must say this issue ended up being more science than engineering, but perhaps that is not surprising considering my scientific leanings. I thought about dropping the word “engineering”, but since the S&E focus has become traditional, I decided to keep the word and wave my hands a bit. Our main feature article is from Dr. Steve Mann of wearable computer fame, and he has certainly pulled off a slick piece of engineering in his videophone watch shown on our cover. Dr. Mann is an accomplished photographer as well as inventor, and he gets credit for the picture of the watch on the cover and the one with his article. Dr. Mann certainly lives on the leading edge of this technology, and we are happy to have another article from him to keep us abreast of developments in wearable computers.

For those readers who like to watch the skies, we have a little of everything: satellite tracking, astronomy (even our French chef, Marcel, has something to say on this one), storms and lightning. Science articles can be found everywhere: in Features, Forum and Strictly On-Line—enjoy!

Linley Gwennap becomes a regular columnist for us this month with his column “Linley on Linux”, where he will keep us up to date on some of the latest happenings in the electronics market for Linux. This month, he tells us what's happening in the field of voice recognition.

Speaking of columnists, Moshe Bar will also be joining us on a regular basis to teach us about kernel issues in Kernel Korner, and next month, we'll have an all-new column on cross-platform programming written by Michael D. Crawford. Both should keep our inner penguin quite happy.

—Marjorie Richardson, Editor in Chief

LINUX JOURNAL

Gri: A Language for Scientific Illustration

Dan E. Kelley

Peter S. Galbraith

Issue #75, July 2000

This scripting language avoids integrating analysis and display capabilities and instead focuses on providing precise and flexible control over the display of technical material.

Like other computer users, scientists sometimes find themselves torn between simple tools and complex tools, between ease of use and power.

Take writing, for example. The simplest tools for writing are those of the office-suite variety, because they are GUI-based. If you can click and point, you can produce results quickly without climbing a steep learning curve.

However, you might also pay a price for this shallow learning curve. Many technical writers who make extensive use of mathematical notation find GUI-based products to be both limiting and annoying when used for anything but a one-pager. For example, graduate students in our research field (Physical Oceanography) often write hundreds of equations in their dissertations. Almost without exception, these students use markup languages (mostly TeX and LaTeX) for this work. It must be admitted that markup-based writing tools are harder to learn than GUI-based tools, but the effort of learning is rewarded with precise control over output that is aesthetically pleasing and flexible enough to meet all reasonable demands.

What goes for words also goes for pictures. For quick jobs, it's lovely to use a GUI-based graphing package or a spreadsheet. However, many users prefer a markup-based system for complicated illustrations, for the same reasons they prefer a markup-based system for complicated text. An additional factor is that GUI-based systems cannot help with illustrations that must be generated automatically without human intervention.

An interesting example is provided by storm-surge forecasts prepared by oceanographers at Dalhousie University in Nova Scotia, Canada. Storm surges are unusual elevations in sea level that are driven by anomalous wind stresses and low atmospheric pressures associated with storms. These surges can cause considerable damage to low-lying areas. Since damage is worsened if a surge happens to occur at the time of a high tide, it is important to make precise predictions of surge arrival times. Surge-induced damage can be greatly reduced if people are given sufficient warning of impending storm surges. Dalhousie researchers have developed numerical ocean models, akin to numerical weather models, for predicting the incidence of storm surges in the northwest Atlantic Ocean. The goal is to provide advance-warning systems that display surge forecasts graphically on the Web. Gri is used for the graphics in this system, because it can be run without human intervention. It is so flexible, researchers can tailor the illustrations to be quickly understood by non-technical users.

We mention this storm-surge example mostly because we find it interesting. Many folks use Gri, so we could have picked other examples of Gri applications in different disciplines of science and engineering. To help you decide whether Gri might help you in your own work, we will explain how to use Gri for some basic scientific illustrations (line graphs, contour graphs and image graphs). This will be enough to get you up and running in a few minutes. After that, we'll outline a few examples from our own work and that of colleagues. In this second part, we won't be hesitant about explaining the scientific background of the work, since we've enjoyed reading such things in this magazine.

Getting Started

One feature scientists like about Gri is that it provides fine control over nearly all aspects of the appearance of the output. This is relevant because scientists have diverse needs, ranging from complicated working plots to pared-down and elegant diagrams for use in proposals, conference presentations and publications. Many Gri users report it is flexible enough to satisfy the full range of applications, removing the need to learn one tool for working plots and another for “publication-quality” illustrations.

Gri achieves its flexibility by being highly configurable; it has many knobs you can twiddle. Just because the knobs are there doesn't mean you need to touch them, since the Gri defaults are reasonable for many applications. The reasonableness of the defaults may well result from the fact that Gri was designed by a scientist for his research work, not by a committee or a corporate structure that had surveyed (or imagined) a market.

Line Graphs

Perhaps the best way to illustrate the defaults and the simplicity of Gri is to show how one would produce the most common form of scientific illustration, a line graph describing x,y data. To be specific, let's suppose we have an ASCII file named linegraph.dat, containing the following x,y pairs:

```
0.05 12.5
0.25 19.0
0.50 15.0
0.75 15.0
0.95 13.0
```

Creating a line graph then takes just three Gri commands:

```
open linegraph.dat
read columns x y
draw curve
```

If these commands were stored in a file called linegraph.gri and if Gri were invoked as **gri linegraph.gri** in a UNIX shell, the result would be a PostScript file named linegraph.ps as shown in Figure 1. Gri does not draw to the screen, because screen-drawing libraries are more limited than PostScript and high-quality PostScript viewers are freely available on all platforms.

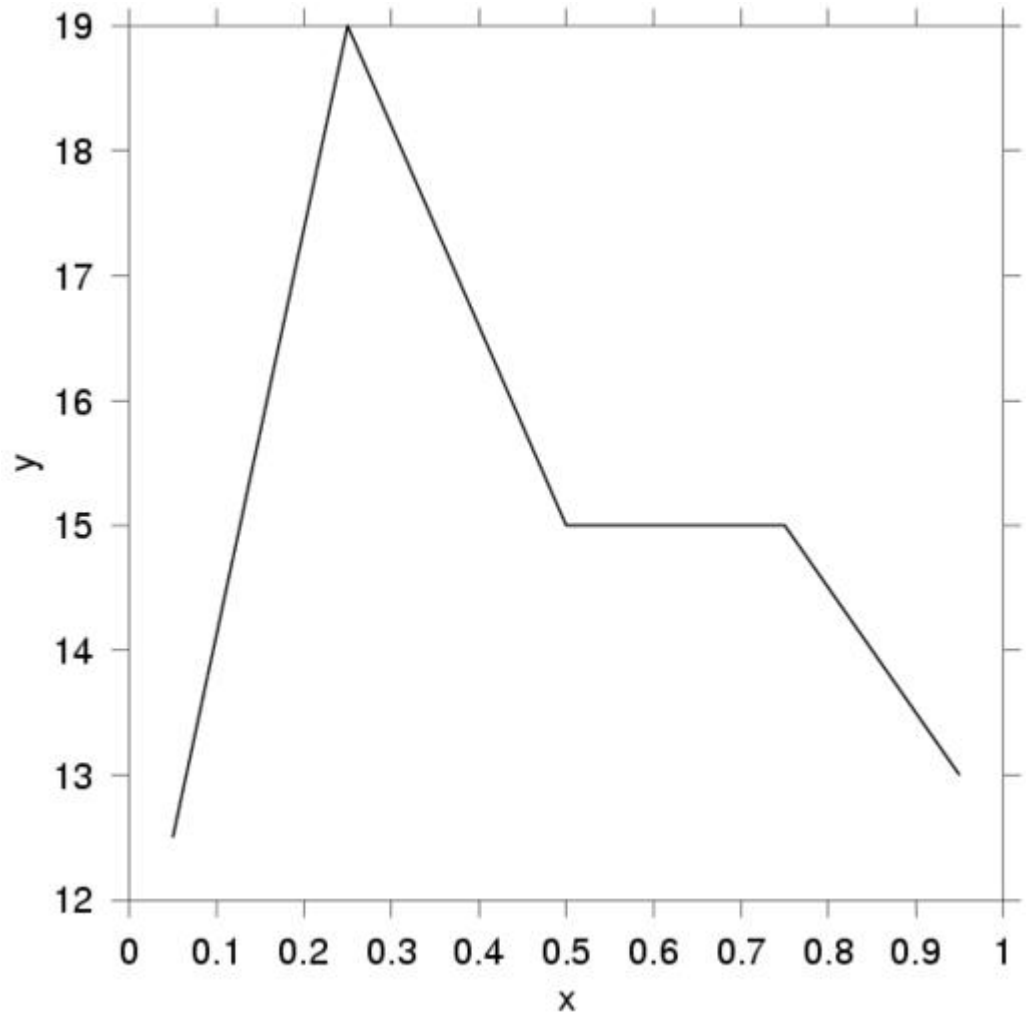


Figure 1. A Simple Line Graph Produced by Gri

The example illustrates a few things. To begin with, Gri syntax is simple, being patterned on English and purposefully using a minimum of punctuation. For example, the **open** command takes as an argument the name of the file containing the data. This file name is not enclosed in parentheses, as it would be in a subroutine-based language. We think the lack of parentheses makes Gri easier to read than some languages. Nor is there any need for the file name to be enclosed in quotation marks, although these are required for file names with spaces in them (and for pseudo-file names created by UNIX pipe commands, which are supported in the Perl style).

Now let's move on to the read line. Gri knows of four types of data: scalars, columns, grids and images. These types are what you'd expect: scalars hold numerical values or strings, columns hold lists of numbers, grids hold two-dimensional matrices and images hold pixels. Each data type has a host of associated read commands and data-processing commands. Folks who are used to programming languages should note that Gri is an illustration-oriented language, not a programming-oriented one, so it attaches special meanings to columns. Thus, the columns named x and y in the example above correspond to the x,y values in a Cartesian space. Gri also has a column named z used for

gridding, a column called w used for weights in statistical operations, columns called u and v for vector fields, etc.

The **draw curve** command tells Gri to draw a curve composed of line segments connecting these points. You may not be surprised, given the focus on drawing, to learn that Gri has many options for **draw**--over 40, in fact. For example, to get symbols drawn at the data points, you could add the line **draw symbol** someplace after the read command. This will draw bullets at the x,y points. Bullets are the default, but Gri provides a dozen other symbol choices. The symbols are designed according to the recommendations of scientific journals and technical-drafting texts. For example, efforts have been made to ensure that the superposition of any two symbols does not result in a symbol that can be mistaken for a third symbol type. The default symbol size is a diameter of 0.1 cm (defined in the expected way for a bullet, and in a reasonable way for non-circular symbols). If this doesn't suit your application, you could issue a command such as **set symbol size 0.2** to create larger symbols (in this case, 0.2 centimeters). This is by no means the only **set** command. It has many options, since Gri is designed to be configurable. Indeed, there is an average of nearly two set commands for each draw command.

In addition to such basic commands and a few others discussed next, Gri provides programming features such as “if” statements, loops and system calls. It also provides a simple scheme for creating new commands to complement the existing commands; e.g.,

```
Draw Logo
{
... Gri commands to draw a logo
}
```

creates a new command to draw a logo. This command is invoked by **Draw Logo**, just as if it were a built-in command. Many users store such definitions in a file called ~/.grirc and thus customize Gri for the sort of work they do.

Gri updates its PostScript output file step by step as it processes commands. When Gri executes any drawing command, the item is drawn with whatever settings (fonts, colors, line widths, etc.) are in place at that time. To use an artistic analogy, draw will paint with whatever color has been stored on the brush with the most recent set command. Many other scripting graphic languages follow a different approach, with some commands being able to alter the form of material that has already been drawn (e.g., axis in matlab). Such languages are meant to be used interactively in an exploratory way, so it makes sense to have this out-of-order processing. However, Gri processes commands in order, because it is designed to be used non-interactively in a planned way.

Gri scripts are not typically written in one large chunk. Instead, Gri users typically write their scripts a bit at a time, running Gri at frequent intervals to see what the results look like. Some like to keep a PostScript viewer open at all times, clicking on an update button every time they run Gri. Most Gri users run it from inside Emacs, which has a button that runs Gri and displays the output. This is but one helpful feature of the Gri Emacs mode. It also provides syntax coloring, indentation, linkage to the on-line Gri documentation, command-specific help, command completion and a search facility that lists all Gri commands containing a given word.

Contour Graphs

Whereas line graphs can be accomplished in as little as three commands, contour graphs are not much more complicated. Suppose we have a file called temp.dat that contains measurements of ocean temperature made once per decade, over the period from 1950 until 2000, at depths of 500 meters, 400 meters, 300 meters and so on, up to the surface at 0 meters. The natural way to store such a dataset is in a matrix or “grid” in Gri. There are three steps to drawing this in Gri.

```
# First, set up the x,y vertices of
# the grid ...
set x grid 1950 2000  10
set y grid  0    500 100
# then read the grid data ...
open temp.dat
read grid data
# and plot contours:
draw contour
```

As you can see, comments in Gri start with the pound sign, as in many other scripting languages. In this example, a uniform grid is used (i.e., time increasing by 10 years between samples), but non-uniform grids are also easy to handle. For many applications, grids are read in from data files instead of being specified in the command file. As for the drawing of the contours, that's fairly easy since Gri can determine, by scanning the grid, a reasonable default range of contours.

Image Graphs

Images are not much harder to deal with. There are many image formats in this world, and Gri handles only a few. This causes few problems, since good conversion programs are available (e.g., ImageMagick). Let's suppose we have a raw (headerless) image file in 8-bit resolution. The 8-bit pixels have 256 possible values, in the range 0 to 255. Satellite-derived measurements of ocean temperature are typically in this format. A common scale for the data is that 4.9 Celsius gets a pixel value of 0 while 30.4 Celsius gets 255, with linear variation in between. Let's say that the image is 512 pixels by 512 pixels, and that the geographical coverage spans a box with the lower-left corner at x=0 and y=0

and the upper-right corner at x=20 and y=20. We want the image drawn with blue for cold water and red for warm water. Listing 1 shows how to handle this in Gri. As you can see, it is quite easy to make Gri create line graphs, contour graphs and image graphs.

Listing 1.

A quick note about file names and other properties. In the examples given so far, we always specified the names of the data files in the script, but that means users have to modify the file for use with other data. For this reason, many scripts use a command called **query** to ask the user the name of the input file. This command stores a user's answers into variables that may later be used in open commands. The query command has an option to provide a list of permitted answers (e.g., “yes” and “no”) so that users cannot make mistakes. In many laboratories, query is used extensively, so that novices can use complicated Gri scripts without having to edit Gri scripts they don't yet understand. This lets students dive into their research instead of getting too distracted with the tools of the work.

Radionuclides Reveal Ocean Mixing

Let's turn to a more complicated example, based on an illustration from one of our research papers. Here, and in the remainder of this article, we will show only fragments of the Gri code, to illustrate topics not already covered.

A key issue in understanding the climate system is the interaction between the ocean and the atmosphere. The heat capacity of the top three meters of the ocean is equal to that of the entire atmosphere, but everyone knows the ocean is more than three meters deep. In fact, the mean depth is more like three kilometers. Given this, it is not surprising that the ocean is important to the heat balance of the planet and thus to the climate system. Beyond the ability to soak up heat from the sun (or re-release it), the ocean has currents that transport heat from one location to another. The exact pathways of this transport are not fully mapped out yet, and we are still unsure of some fundamental dynamical aspects of the system. A great deal of evidence suggests that vertical mixing in the ocean is very important to these patterns of heat flow. Thus, ocean mixing is important not just to the ocean itself, but also to the whole climate system. This is a prime motivation for studies of ocean mixing, which is our research specialty.

One good way to observe mixing is to insert a tracer and see where it goes. This is not as easy as it sounds, so oceanographers also make great use of tracers that are naturally occurring or introduced without the specific intention of studying mixing. An example of the latter is a suite of radionuclides introduced into the atmosphere by atmospheric bomb testing carried out by the U.S. and

the U.S.S.R. in the 1960s. These radionuclides have found their way into the ocean. One radionuclide, a hydrogen isotope named tritium, is especially useful for studying ocean mixing. Figure 2 shows a plot from a paper (see Resources) in which one of us, together with a geochemist colleague named Kim Van Scoy, tried to work out the rate of ocean mixing by tracing the movement of tritium over several decades.

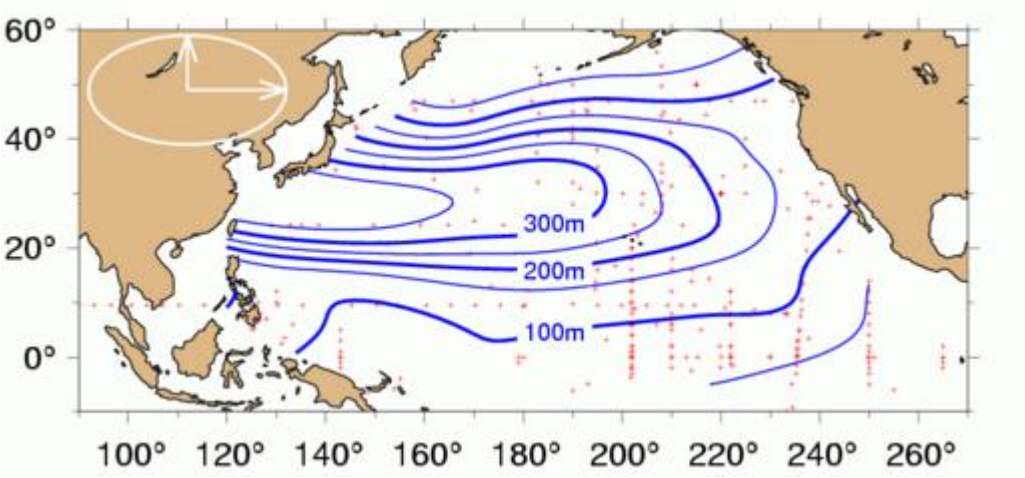


Figure 2. A Contour Illustration, Showing the Depth of Maximal Concentration of Tritium in the Ocean

Our approach was to do time derivatives of properties that are spatially averaged over a particular geographical area. The area is defined by the streamlines of the average ocean circulation pattern. Now, averaging gridded data is easy: just sum along the rows and columns of the grid. Unfortunately, however, our observations were not made on a grid. As Figure 2 shows, the observations are at manifestly non-uniform locations which correspond to ship tracks on cruises designed with other goals in mind. Our first step, then, is to cast these x,y,z data onto a uniform x,y grid. After reading in the columns and setting up an x,y grid (exactly as discussed above), we create the grid by using

```
convert columns to grid barnes
```

This is our first example of a **convert** command. Commands in this category perform conversions from one data type to another. Since contouring works on a grid, we must convert column data into the grid. In this case, we have chosen to use the so-called “barnes” method of gridding, which is but one of several gridding methods in Gri (see Resources). The method applies a Gaussian-weighted low-pass filtering (averaging) scheme that is run iteratively. Initial iterations map out the large-scale variation in the field, while later iterations fill in details in regions where the data sampling is intense enough to justify doing so.

In the previous example, we let Gri pick contours; in fact, that is how we started out with the working plots that led to the diagram shown in Figure 2. For

publication, however, we ended up deciding to highlight certain contours by drawing them with thicker lines:

```
set line width rapidograph 00
draw contour 50 unlabelled
set line width rapidograph 2
draw contour 100
```

where the “rapidograph” method of naming line thicknesses by the scheme used for Rapidograph technical fountain pens has been used. Line widths may also be given in units of printer points.

At this stage, we had the scientific result we wanted. We used the Gri command **write grid** to write the gridded data into a file, then we integrated the values with a Perl script (called from within Gri with a system command), then we were done with part of the work. For presentation, we wanted to draw this in a map format, so we started by customizing the axes a bit:

```
set x name ""
set y name ""
set x format %.0lf$\circ$
set y format %.0lf$\circ$
```

The first two commands remove the names x and y from the axes by setting the names to blank strings. The second two set the format of the axis numbers, using the notation of the C programming language. We also added a LaTeX-like part to the format (enclosed in dollar signs) to make Gri draw degree signs for longitude and latitude. Gri can draw Greek letters and mathematical symbols in this way, although the emulation of TeX is by no means complete, since it is a daunting task.

A map isn't too helpful without coastlines. Since coastline files are quite big, our system retains the data in a binary format for quick reading. We use the format called netCDF, which is quite popular in earth science (see http://www.unidata.ucar.edu/packages/netcdf/). Out of the many advantages to this format, one is quite relevant to Linux users: it is fully binary-compatible across big-endian and little-endian computers and across computers with different word lengths. It also permits naming of data entities, so you don't have to remember that the first column is latitude and the second longitude, or vice versa. Gri handles netCDF format easily:

```
open map_land.nc netCDF
read columns x="lon" y="lat"
```

is all it takes to read the coastline data. We'll fill in the coastline with a light brown color, and then draw a black coastline:

```
read colornames from RGB "/usr/lib/X11/rgb.txt"
set color burlywood
draw curve filled
```

```
set color black
draw curve
```

In addition to X11-based colors and a dozen or so familiar colors above like black, Gri permits you to specify colors in either an RGB or an HSV framework.

Viewing the Ocean with Satellites and Ships

Speaking of colors, a common application of Gri is generating illustrations of color images. Within oceanography, such images often fall into two broad categories: fields generated by numerical models and fields generated by satellite observation. In each case, the advantage of Gri over tools that are more image-based is that Gri invites the user to draw other graphical elements as well as the images.

Figure 3 provides a good example, showing some of the work in the ECOLAP program, spearheaded by oceanographers at the Rio Grande University in Brazil (see http://www.peld.furg.br/). This research group has a ten-year program to measure and understand the physical and biological variability of the Brazil Rio Grande estuary and the adjacent sea. Figure 3 shows a satellite image of ocean temperature, and the location of ship-based observations made on 22 February 2000. Land is colored a ruddy brown in the figure, and the palette indicates sea surface temperature as measured by the satellite. The processing of the satellite image is done exactly as described in the more hypothetical example given near the beginning of this article. The two panels are drawn simply by changing axes and redrawing. The palette was drawn with a command called **draw palette**, the guiding lines were drawn with the commands **draw box** and **draw line** and the labels were drawn with **draw label**. By now, you may be getting the impression that it's pretty easy to guess the names of Gri commands. This guessing isn't necessary; just type “draw” in the Emacs mode and press the **ESC** key followed by the **TAB** key, and the mode will display all commands starting with the word “draw”, i.e., all drawing options.

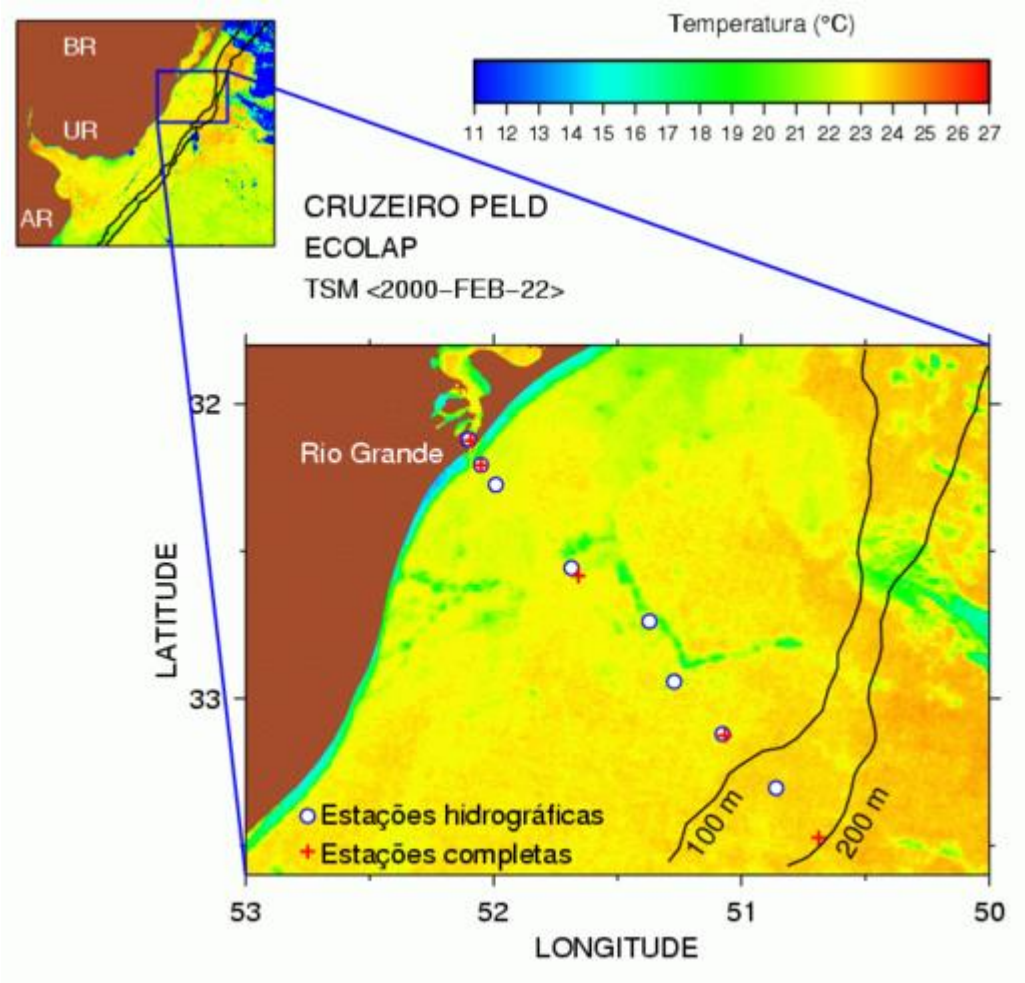


Figure 3. Image Showing Satellite-Derived Sea Surface Temperature East of Brazil, Uruguay and Argentina

A noteworthy feature of Figure 3 is the use of symbols to indicate the locations at which ship measurements of ocean properties were made. These ship-based observations usually run from the ocean bottom to the surface, and typically involve measurements of physical properties of the water as well as biological properties, such as the occurrence of different species through the depths. In past decades, oceanographers were greatly challenged to explain patterns in ship-based observations, and Figure 3 illustrates why. Consider the ship sample near the middle of the larger image. It lies in a thin filament of cold water (green color), whereas the other samples lie in warmer water. To some extent, the biology is just “along for the ride” as currents move water from one place to another, so it might not be surprising if this middle sample had different biological characteristics (e.g., species typical of cold water) than the nearby stations. The superposition of satellite and ship data on one graph, which is so easy to accomplish in Gri, provides a powerful insight into the systems under study.

It almost goes without saying that the script-based nature of Gri is important in constructing such diagrams. Nothing about this diagram was prepared with a mouse, and nothing in the Gri script requires modification for another cruise of

the ship (since query commands are used to set up all file names). As soon as the ship does an observation and a latitude-longitude pair is written in a data file, the Gri script can be rerun and a new diagram prepared.

New Ways to Measure Ocean Mixing

Understanding turbulent flow is one of the grand challenges in physics, and ocean mixing provides a good example. The ultimate goal is to be able to predict mixing, which is a small-scale phenomenon that is difficult to measure, based on large-scale properties that are easy to measure. One proposed technique is to examine vertical variation of water density on intermediate scales. If this can be done, it will greatly expand our database of ocean mixing knowledge, since density measurements are common. But can it be done? This question was addressed in the Ph.D. dissertation of the second author. Figure 4 shows a diagram patterned after a paper about this work. The illustration shows two things. The red image shows a theoretical prediction of where mixing should be occurring, through time and through depth. The blue-filled curves show where mixing actually occurred.

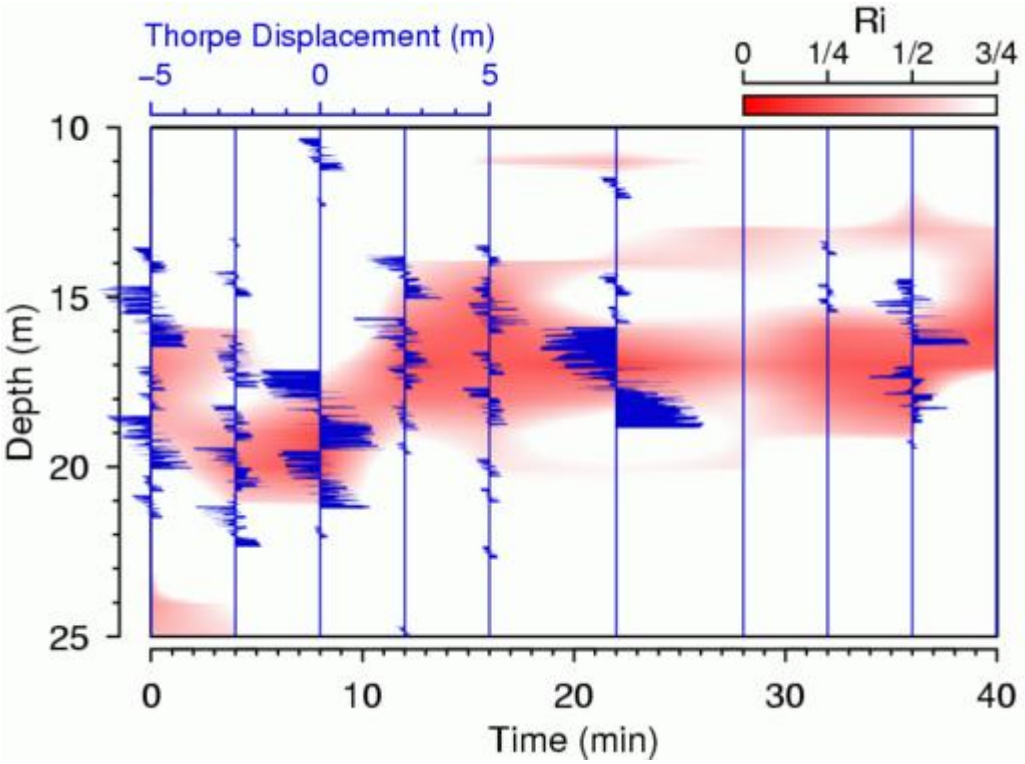


Figure 4. Indicators of Mixing in Surface Regime of Ocean

To be more specific, the red image shows the so-called Richardson number. Theory indicates that the type of mixing known as Kelvin-Helmholtz overturning can occur only when the Richardson number, a measurement of the competition between stabilizing and destabilizing effects, falls below a critical value of 1/4. We measured the variation of the Richardson number over depth and time on a grid. Our first inclination was to contour this, but since we

wanted to superimpose other things, we decided to use an image instead for clarity. The gist of the Gri code can be guessed from what you've read so far, the only new feature being the use of the command **convert grid to image** to transform our grid data into an image that can be colored. We drew the image in shades of red by running the color scale across intensities of the red hue, instead of across hues as in the previous example. We did this because we wanted to superimpose another curve of a certain color, and that would be difficult to discern with a spectrum below. The image indicates that mixing should occur in a band that lies roughly at 17 meters deep and that this mixing band should bob up and down over the time of observation.

With the theory painted in red, we'll turn to the observations. Blue seemed to look pretty, so we started by drawing blue vertical lines corresponding to the times when a density-measuring probe was lowered from the side of the ship. We call the density variation with depth a density “profile”. The seawater density in the ocean normally increases with depth, because heavy water sinks and buoyant water rises. However, eddying mixing motion can overturn this density profile, momentarily lifting heavy water above light water. With sufficiently precise density probes, this sort of mixing can be revealed graphically by plotting the difference between the observed density profile and an artificial profile created by reordering the density data to make density increase monotonically with depth. We keep track of the distance individual points had to be moved in the reordering process and call this the “Thorpe displacement profile”. This profile gives an indication of the intensity and extent of mixing patches. We draw these Thorpe profiles with a filled curve, as in the command

```
draw curve filled to x 0.0
```

We do this once for each profile, after first redefining the axes so that x=0 corresponds to the time of the ship observation.

You may agree that the observations are in rough agreement with the theory, since the observed depths and times of high mixing rates (blue curves) appear to match with the theory (red image). The main implication of this is not that low Richardson numbers yield mixing; we already knew that, from experiments in the field and in the laboratory. Rather, the main implication is that our density probe is capable of picking up mixing signals of this particular strength. This is important, because the instrument we were using is much more common than the instruments normally used to measure mixing. For more on how and why the technique works, we encourage you to consult our paper, which, we might add, employs Gri for every figure.

Notice that the axes in this diagram lie outside the box in which the data are drawn. The second author prefers this style, while the first prefers the conventional style. In this, as in most things, Gri offers you a choice.

Resources

Dan Kelley (Dan.Kelley@Dal.Ca) is an associate professor of oceanography at Dalhousie University in Nova Scotia, Canada.

Peter Galbraith (psg@debian.org) is a research scientist with the Canadian Department of Fisheries and Oceans.

Dan wrote Gri and Peter wrote the Emacs mode. The fact that neither author is a professional programmer may explain the limited practical nature of these tools.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Tracking Satellites with PREDICT

John A. Magliacane

Issue #75, July 2000

A look at the development and use of an open-source satellite-tracking and orbital-prediction program.

When Sputnik 1 was launched into orbit on October 4, 1957, the space age was born and the fields of science, engineering and technology were changed forever. At last count, there were over 8500 payloads from over 30 countries in orbit around the earth. All of these spacecraft are bound to their home planet by the Earth's gravitational field, and their motions can be described by simple principles of gravity and planetary motion discovered by scientists such as Isaac Newton and Johannes Kepler hundreds of years ago.

Today, earth-orbiting satellites serve many purposes and play important roles in global positioning and navigation, communication networks, scientific exploration, earth resource research, national defense, weather monitoring and education. USSPACECOM, the United States Space Command (formerly NORAD), along with NASA, the National Aeronautic and Space Administration, use radar and optical-ranging techniques to keep close track of the thousands of man-made objects in earth orbit and provide orbital data suitable for orbital modeling and open-ended tracking of unclassified payloads. With an accurate set of orbital parameters in one's possession, it is possible to determine velocities and the past, present and future positions of a satellite in its orbit around the earth with a degree of accuracy suitable for many science and engineering applications.

The Development of PREDICT

PREDICT is an effort to bring a versatile, open-source, satellite-tracking and orbital-prediction application to the Linux operating system. PREDICT was adapted from ideas developed in earlier satellite-tracking and orbital-prediction software written nearly a decade ago for use on the then-popular Commodore 64 home computer.

The original version of PREDICT was created as a replacement for the QuickTrak satellite orbital prediction program that was also available for the C64. While QuickTrak was written in BASIC and its source code was interpreted at runtime, PREDICT was written in C and compiled into 6502 machine code. The sole reason for writing PREDICT was to be able to quickly forecast passes of amateur radio communication satellites in advance of their arrival.

For real-time satellite tracking, a separate program called SpaceTrack was written, using a combination of BASIC and hand-assembled machine code. SpaceTrack was sophisticated enough to permit the display of a satellite's position on a bit-mapped Mercator projection map of the world. It even had the ability to articulate the tracking coordinates of a satellite through a voice synthesizer connected to the Commodore 64's user port. The speech synthesizer was used to relay tracking coordinates to a visual observer in real time over a short-range radio link so that the Mir space station and other large spacecraft could be easily located and identified in the evening sky. The speech routines were written entirely in hand-assembled machine language and executed through the same address vector as the computer's hardware interrupt routines. This essentially created a multitasking environment, with the voice synthesizer receiving data through a background process that in no way interrupted the numerical processing taking place in the foreground by the BASIC interpreter.

Although neither program was ever released to the public, they served me quite well for several years until a switch from the aging Commodore 64 to a more modern MS-DOS-based PC was made. In many ways, the switch to the MS-DOS platform was a significant step backward from the C64, especially in terms of programming flexibility and the requirement to relearn the programming process under the new environment. PREDICT was eventually ported to MS-DOS, but the MS-DOS environment simply was not enticing enough to add many new features to the program. Furthermore, there was seemingly no simple way of multitasking and passing parameters between processes under MS-DOS as was possible (as odd as it sounds) on the older and less-sophisticated C64.

PREDICT was also ported to several multi-user UNIX machines around the same time, but hardware differences and the lack of a true understanding of the operating environment prevented further development of the program. Nevertheless, the DOS version of PREDICT was polished up and released to several popular Internet and dial-up software repositories as free software in May 1994, and became quite popular among amateur radio operators involved in satellite communications.

By the time Windows 95 was released, it was time to switch computing environments entirely to Linux. PREDICT was successfully ported from DOS to Linux, and has functioned well in the Linux environment for many years. A pre-compiled Linux binary of PREDICT was released as free software to several FTP sites in 1996. Then in 1999, major portions of the program were rewritten, and in an effort to enhance PREDICT's functionality, several real-time satellite-tracking modes similar to those available in the original SpaceTrack program were added to the program. Speech routines were also added, but instead of using a voice synthesizer to produce vocal announcements, audio samples were sequentially directed to the system sound card using a separate program that was invoked by PREDICT. Much like the original design of SpaceTrack, the speech routines were executed as background processes so as not to delay the execution of real-time orbital calculations while the announcements were being made.

The Benefits of Open Source

The newly enhanced version of PREDICT was released as open-source software under the GNU General Public License and uploaded to Metalab and several other FTP sites in December 1999. Several weeks later, major portions of PREDICT Version 2 were successfully ported from Linux to DOS using Caldera's DR-DOS operating system and the DJGPP software development environment. This was done to serve as a replacement for the earlier DOS version of PREDICT that was released in 1994.

Within weeks of the Linux release of PREDICT Version 2, the benefits of open source and the GNU General Public License were quickly realized. Bdale Garbee, amateur radio operator N3EUA, built and packaged PREDICT for inclusion in the “potato” release and all later versions of Debian Linux. The Debian PREDICT package is available for all Debian-supported CPU architectures.

Jean-Paul Roubelat, amateur radio operator F6FBB, modified PREDICT to allow the program to control the azimuth and elevation rotators that support his satellite antennas using a hardware interface known as a Kansas City Tracker. Using PREDICT, Jean-Paul was able to have his Linux-based computer automatically track his satellite antennas with the passage of OSCAR satellites in range of his home.

Ivan Galysh, KD4HBO, working at the U.S. Naval Research Laboratory in Washington, Maryland, selected PREDICT for tracking the Stensat satellite. With the source code in his possession, Ivan was able to turn PREDICT into a socket-based server, allowing the program to make real-time tracking data available to external programs through UDP socket connections. One of the client programs Ivan created was an antenna-rotator-control program similar to the

function of Jean-Paul Roubelat's program, except it used a different hardware interface. Another was an XForms-based GUI map display program that plots the location and orbital path of satellites being tracked by PREDICT on a Mercator projection map of the world.

By March, a third program that reads Doppler shift information calculated by PREDICT and uses that data to correctly adjust the operating frequency of uplink transmitters and downlink receivers used in satellite communication systems was under development. With the benefits of the socket-based server code clearly evident through these modular client applications, Ivan's server code was made an integral part of PREDICT's official source code and was released in version 2.1.0 in early April 2000.

Features of PREDICT Version 2.1.*x*

The latest version of PREDICT may be downloaded from ftp.amsat.org/amsat/software/Linux/predict-latest.tar.gz or from metalab.unc.edu under the /pub/linux/apps/ham subdirectory.

As of version 2.1.0, PREDICT's major features included:

- A *fast* orbital prediction mode that predicts passes of satellites, providing dates, times, coordinates, slant-range distances and sunlight and optical visibility information. Predictions are displayed in tabular form and may be saved to a log file for later reference, printing or parsing by other programs.
- An optical visual orbital prediction mode that displays satellite passes that may be optically visible to the ground station.
- A solar illumination prediction mode that calculates how much time a satellite will spend in sunlight each day.
- A real-time tracking mode that provides dynamic information such as sub-satellite point, ground station azimuth and elevation headings, Doppler shift, path loss, slant range, orbital altitude, orbital velocity, footprint diameter, orbital phase, the time and date of the next AOS (acquisition of signal or LOS, loss of signal, of the current pass), orbit number and sunlight and visibility information for a single satellite, while providing live azimuth and elevation headings for both the sun and moon.
- A multi-tracking mode that provides sub-satellite point, azimuth and elevation headings, sunlight and visibility and slant-range distance information for all 24 satellites in the program's current database on a real-time basis. Azimuth and elevation headings for the sun and moon are also provided, as a well as a listing of the AOS dates and times for the next three satellites expected to come into range of the ground station.

- Static information such as semi-major axis of ellipse, apogee and perigee altitudes, and anomalistic and nodal periods of satellite orbits.
- Command-line options which permit alternate ground station locations to be specified or alternate orbital databases to be read and processed by the program, effectively allowing an *unlimited* number of satellites to be tracked and managed. Additional options allow any orbital database file to be updated automatically using NASA Two-Line element data obtained via the Internet or via pacsat satellite without having to enter the program and manually select menu options to update the database.
- A voice mode that allows live azimuth and elevation headings of a satellite to be articulated to an observer to assist in locating a satellite by optical means.
- A socket-based server mode that permits PREDICT to be used to supply real-time tracking data, such as azimuth and elevation headings, footprint diameters, sub-satellite point latitude and longitude values, normalized Doppler shift data and next-predicted AOS times to external programs such as rotator control software, graphical map-tracking software or radio frequency control programs residing either on the host machine or any networked client.

Software Installation

The process of installing PREDICT differs somewhat from that of most open-source software requiring compilation at installation time. An ncurses-based installation program is used to probe the system hardware for the existence of a sound card, and header files are built according to this information and with reference to the installation directory chosen by the user. The program is then compiled, and the resulting executable file is symbolically linked between the installation directory and /usr/local/bin. The same procedure of using symbolic links is also used for PREDICT's man page. The main advantage of this installation method is that it is simple, relatively error-proof and distribution-independent. It also keeps all program files under a single subdirectory, rather than scattering them throughout the entire file system.

Once the program is built and installed, the user is asked to enter his latitude, longitude and altitude above sea level. A database of Keplerian orbital data is also required for the satellites of interest to the user. A database of amateur radio and several “high-interest” satellites is included with the program to get things started. Since the accuracy of Keplerian orbital data seldom remains high for long periods of time, facilities are included in the program to permit the database to be updated from more recent element sets. A simple shell script is included with the program to facilitate this update through an anonymous FTP connection to ftp.celestrak.com. This script may even be invoked through a crontab, permitting automatic updates of PREDICT's orbital

database to take place on a regular basis without the need for user intervention.

Running PREDICT

Figure 1. PREDICT's Main Menu

PREDICT is a text-based program, and its start-up screen (see Figure 1) lists all of the program's main functions. Several tracking and orbital-prediction modes are available, as well as several utilities to manage the program's orbital database.

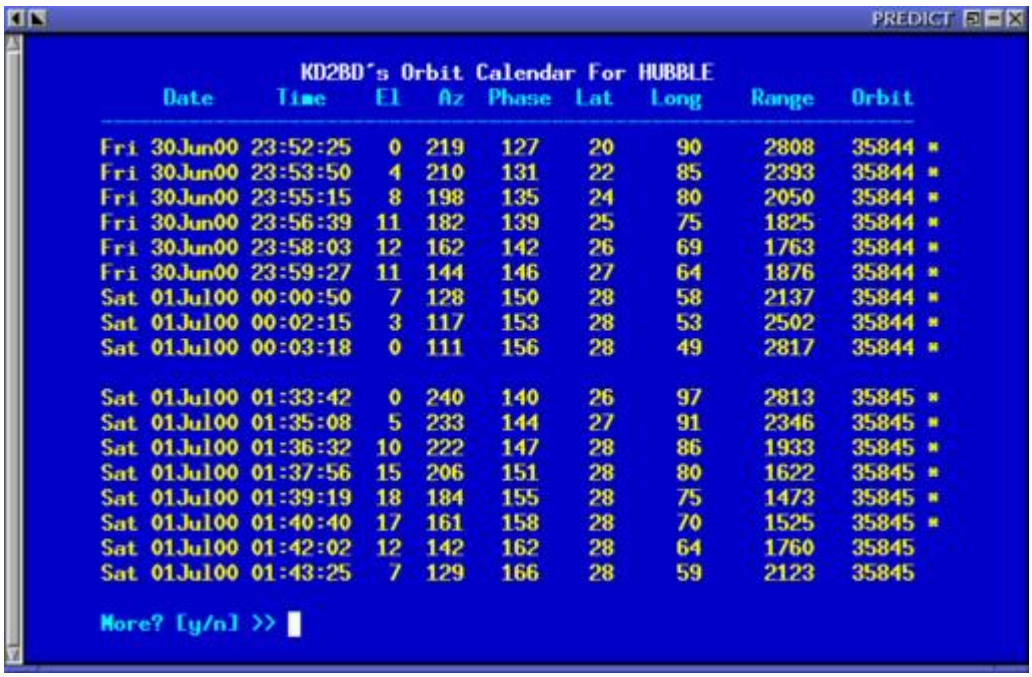


Figure 2.Orbital Prediction Mode

PREDICT includes two orbital-prediction modes to predict any pass above a ground station, or list only those passes that might be visible to a ground station through optical means. In either case, predictions will not be attempted for satellites that can never rise above the ground station's horizon, for satellites in geostationary orbits, or satellites that appear to have decayed in the earth's atmosphere since the last Keplerian orbital data update. If a satellite is in range at the starting date and time specified, PREDICT will adjust the starting date back in time until the point of AOS, so that the prediction screen displays the pass in its entirety from beginning to end. Figure 2 shows the orbital prediction mode of several passes of the Hubble Space Telescope in range of New Jersey in early July, 1999.

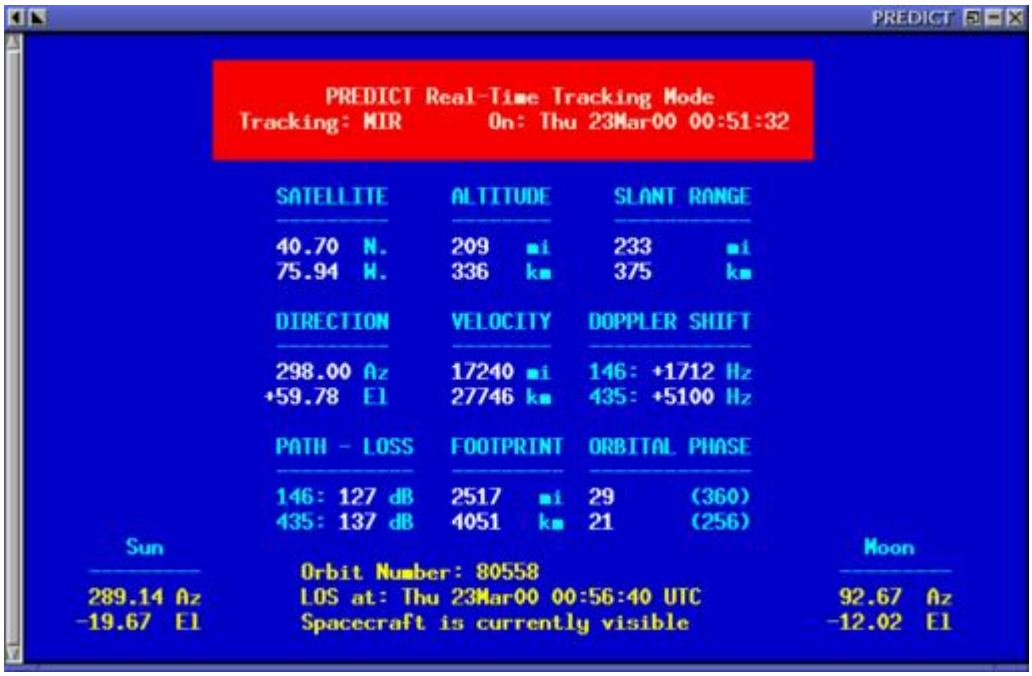


Figure 3. The Single-Satellite-Tracking Mode

In addition to predicting satellite passes, PREDICT allows satellites to be tracked either individually or as a group of 24 using the program's Multi-Satellite Tracking Mode. The positions of the sun and moon are also displayed when tracking satellites in real time, as are the eclipse and optical visibility conditions of the satellites in the database. Real-time tracking data is available to socket-based clients when PREDICT is running in either the single-satellite or the multi-satellite-tracking mode. Figure 3 displays tracking coordinates for a single satellite in real time. Real-time positions for 24 satellites are shown in Figure 4, along with a schedule for upcoming satellite passes. Satellites currently in range are highlighted for easy identification.

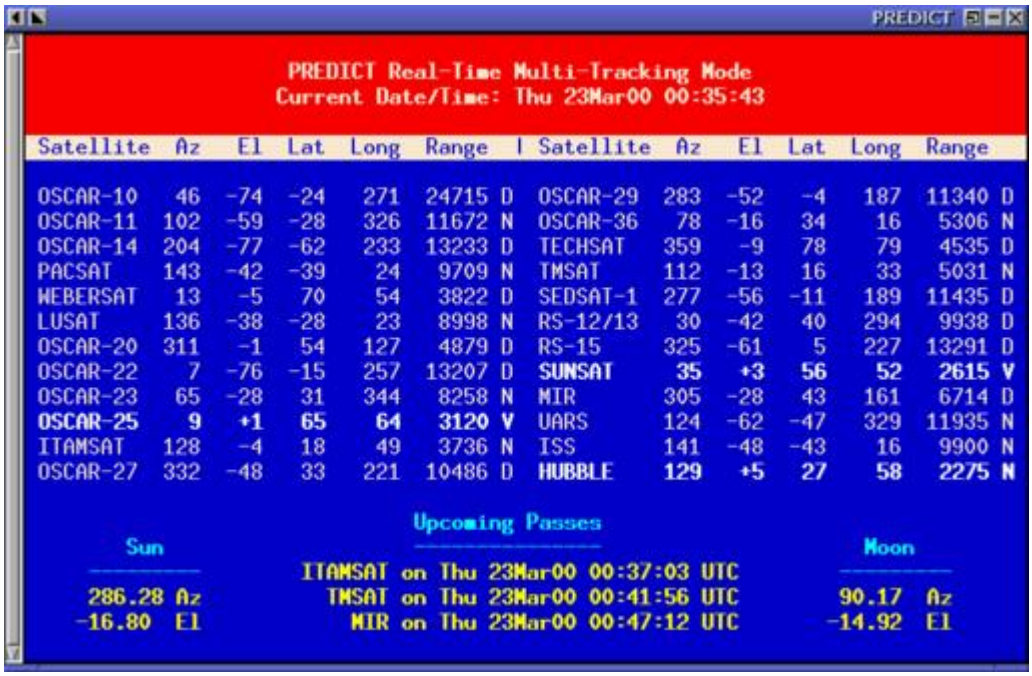


Figure 4. Multi-Tracking Mode

Applications and Use

PREDICT was designed primarily to aid in facilitating communication through amateur radio satellites. Nearly 20 satellites currently in orbit carry some form of communication transponder or telemetry beacon intended for amateur radio use. OSCAR spacecraft (orbiting satellite carrying amateur radio) that contain analog transponders relay signals they receive back to earth in real time. Those that carry digital transponders relay files between sender and recipient anywhere on the planet on a store-and-forward basis. Some OSCAR satellites also carry earth-imaging cameras and scientific and educational experiments, the results of which are transmitted by low-power beacon transmitters carried on-board the satellites. Even the U.S. space shuttles and the Mir space station have carried amateur radio equipment into orbit for use by astronauts and cosmonauts working in space. The International Space Station (ISS) will carry a multi-mode amateur radio station for use by astronauts working on the space station. The image of Melbourne, Australia in Figure 5 was taken by the earth-imaging camera on-board the TMSAT-OSCAR-31 amateur radio satellite.

Since none of these spacecraft are in geostationary orbit, some form of tracking and orbital prediction must take place before radio contact may successfully occur. PREDICT provides all the information needed to predict passes of these spacecraft over a particular ground station location, and track them in real time once they have arrived. A graphical orbital-display program operating as a socket-based client of PREDICT is shown in Figure 6. The footprint as well as several consecutive orbits of the Mir space station are shown on the map.

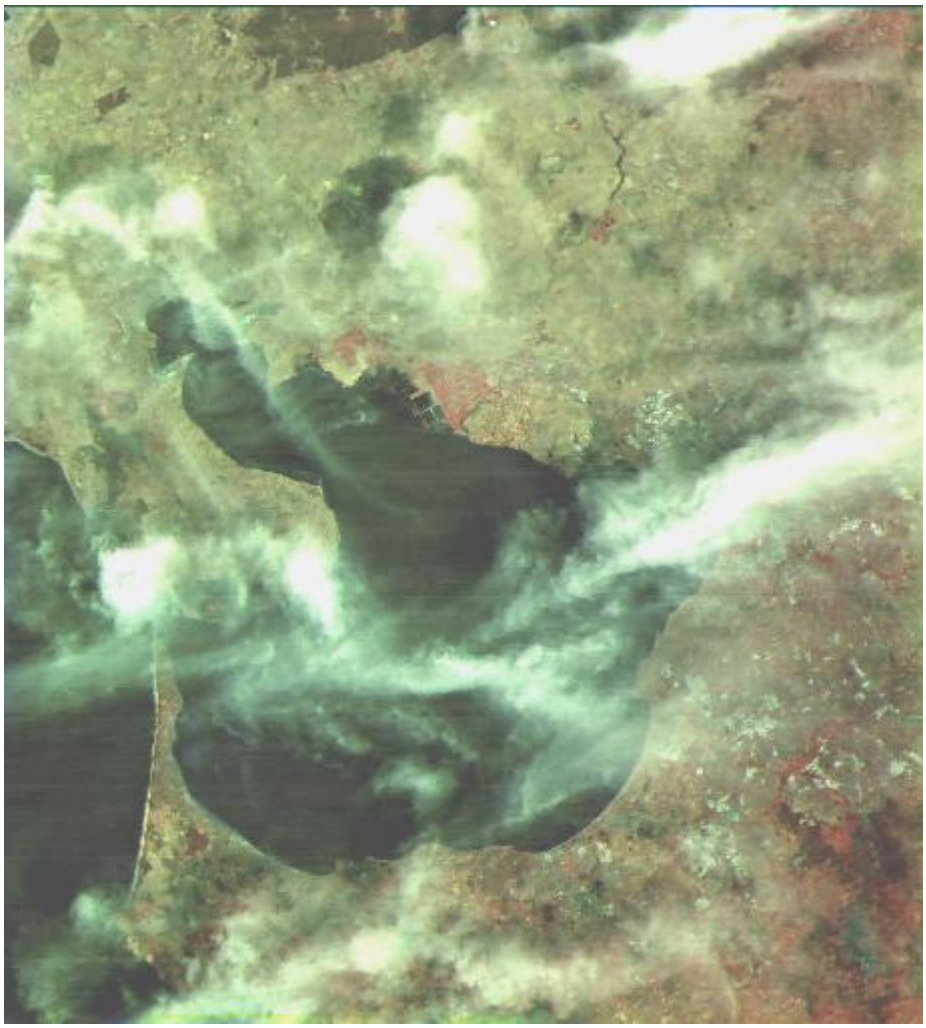


Figure 6. Socket-Based Client Graphical Orbital Display

In addition to providing real-time tracking coordinates, PREDICT also calculates the amount of Doppler shift expected at any given moment during a pass, so that compensation in uplink transmitter and downlink receiver frequencies may be accurately made. Path loss calculations for determining the RF (radio frequency) link budget between ground station and satellite are also provided.

Since PREDICT also tracks the sun and moon in real time, the sky locations of these celestial objects can be used to determine geographical bearings at the user's location. This information is particularly helpful in accurately aligning directive antenna systems to known directions prior to tracking satellites. Since PREDICT also tracks the position of the sun and earth with respect to satellites tracked by the program, spacecraft telemetry can be better analyzed knowing when the satellite being studied has spent considerable periods in sunlight or in eclipse. The Solar Illumination Prediction mode can determine in advance when or if a spacecraft will enter a “solar orbit” and experience periods of continuous sunlight. These periods are also typically the best times for astronauts to plan extra-vehicular activities in space.

Images from weather satellites in geostationary orbit are often seen on television and via the Internet, but a host of weather satellites from the United States, Russia and China in low-earth orbits provide high-resolution images over regional areas. Receiving images from these satellites is a rewarding hobby for many, and PREDICT can provide all the information required for predicting passes of weather satellites and tracking them from horizon to horizon once they have arrived.

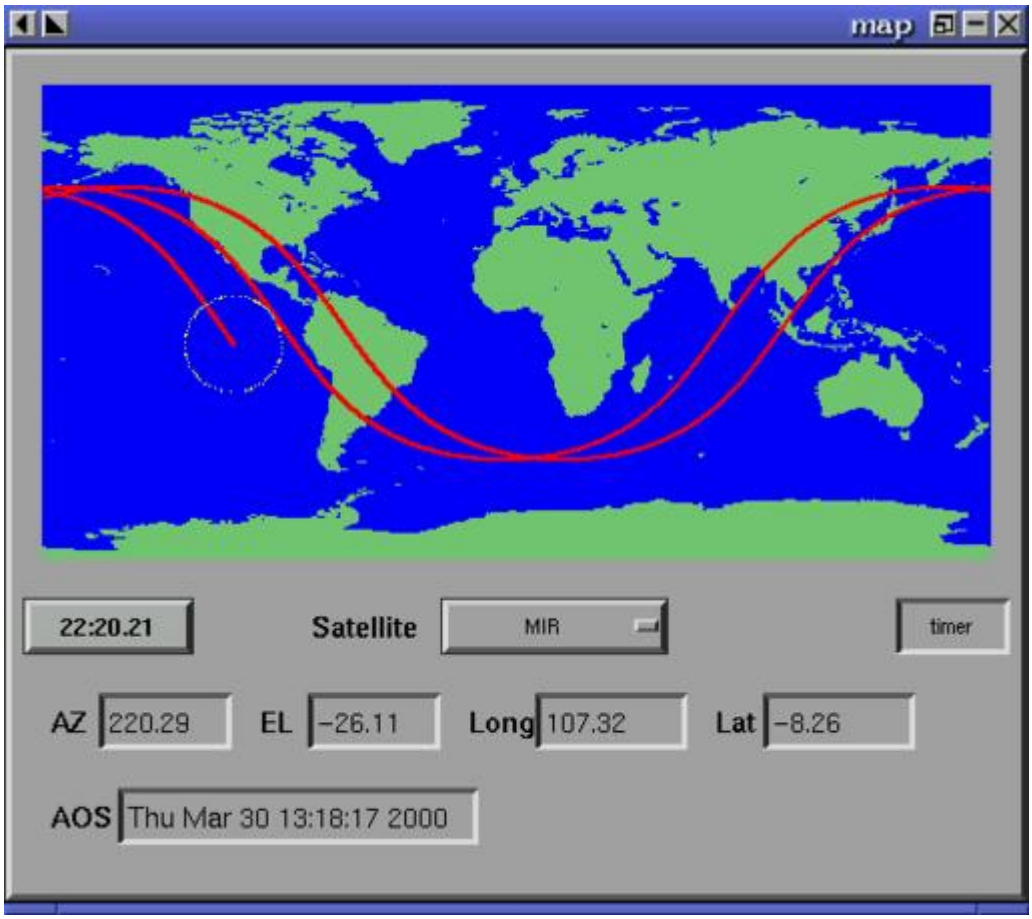


Figure 5. Melbourne, Australia

Finally, PREDICT also determines periods when large spacecraft may be visible in the evening or predawn skies. There are approximately 150 satellites that are classified as being “visible”, all of which can be accurately tracked through PREDICT. Large spacecraft, such as the U.S. space shuttles, the Hubble space telescope, the Mir Space Station (see Figure 7) and the International Space Station are easily seen by the naked eye under the right viewing conditions. The International Space Station will be particularly interesting to watch as it slowly expands with construction scheduled to take place over the next few years. PREDICT's voice capabilities are ideal for relaying tracking coordinates to satellite observers, effectively eliminating the need to read a computer monitor or printout for real-time tracking information.

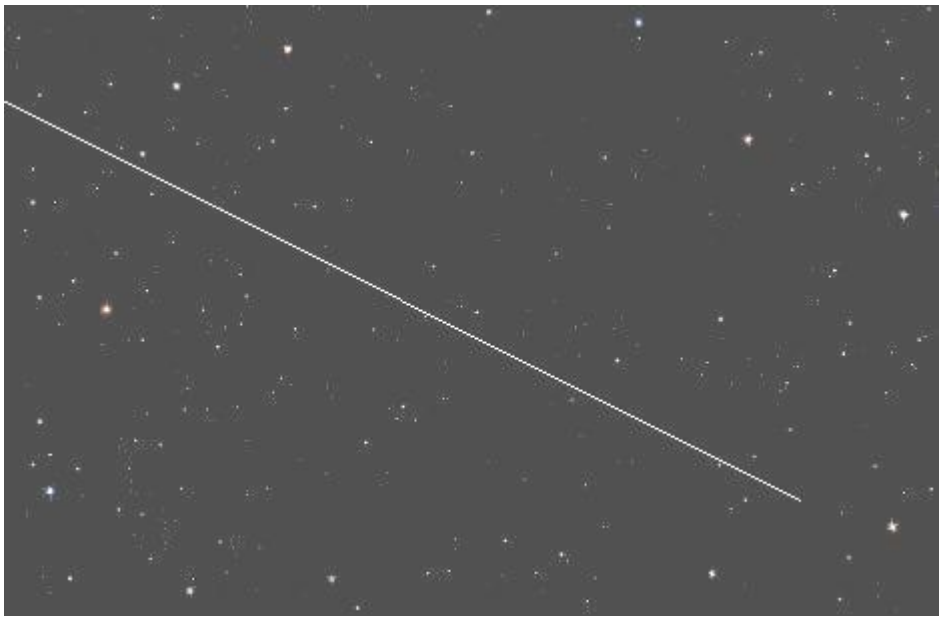


Figure 7. The Mir Space Station

Conclusion

Development of PREDICT continues on an almost-daily basis after having been released as open-source software under the GNU General Public License last year. Through the development of PREDICT, the Linux operating system has clearly shown itself as being a superb platform for the design, development and implementation of applications relating to science, engineering and education. The free exchange of ideas, the open scrutiny of those ideas among peers, and the constructive feedback gained from such open discussions is not unlike the long-held traditions of the science and engineering fields. This environment will surely contribute to the continued success of Linux, not only in the fields of science and engineering, but in many other areas as well.

Resources

email: kd2bd@amsat.org

John A. Magliacane has been using Linux since 1.1.59. He holds an advanced class FCC amateur radio license (KD2BD) as well as a commercial FCC radio operator's license. His interests include satellite communication systems, Linux software development and hardware design. John may be reached via e-mail at kd2bd@amsat.org, or via the KITSAT-OSCAR-25 satellite.

LINUX JOURNAL

Detecting Chaos in the Field

Juergen Kahrs

Issue #75, July 2000

All that is real is reasonable, and all that is reasonable is real. —G.W.F. Hegel, 1770-1831

Scientists and engineers were among the first to notice what a powerful combination the Linux kernel and the GNU tools are. Thus, it is no surprise that it was the sober scientists who started replacing expensive *supercomputers* with inexpensive networks of GNU/Linux systems. In spite of the strong position of GNU/Linux in all areas of scientific computing, there are still some aspects of the Linux kernel which have been neglected by engineers. One of them is the sound card interface.

In the early days of Linux, sound cards were notoriously unreliable in their ability to process data and continuous signals. They were supposed to handle sounds in games, and nothing more; few people tried to record data with them. Today, modern sound cards allow for simultaneous measurement of signals and control of processes in real time, and good sound cards can compete with expensive *data acquisition cards* which cost more than the surrounding PC. This article demonstrates how to abuse a sound card for measurements in the field.

With appropriate software, an ordinary PC can do much more than just record data in the field and analyse it off-line in the office. Due to the extreme computing power of modern CPUs, it is possible to analyse data while recording it in real time. On-line analysis allows for interactive exploration of the environment in the field, just like oscilloscopes of earlier days, but with an added dimension.

Sound Installation

First, you will need to install the sound card and its drivers in your Linux system. As usual, the book that comes with your Linux distribution should help

a lot, and HOWTOs on the Internet provide the necessary instructions (see sidebar “Sound Installation”). When your sound card and its drivers are installed, you should do some testing with it. Make a recording while adjusting the mixer gain, and play the recording back through your loudspeaker. The mixer is an important feature of your sound card, because it allows you to adjust the sensitivity of the A/D converter to the level of the signal to be recorded. This is even more important when recording with a resolution of 8 bits. Keep in mind that we are looking for fast and robust measures of qualitative effects, rather than precise quantitative measurements.

Phase Space

If your GNU/Linux system does not already have some audio applications installed, see Resources for a collection of addresses on the Internet. There, you will find applications like **smix**, a mixer with a well-designed user interface. It can handle multiple sound cards, and has the features needed for serious work:

1. interactive graphical user interface
2. command-line interface
3. configuration file settings

It makes no difference to your sound card whether the recorded signal comes from an acoustic microphone or an industrial sensor. Signals of any origin are always stored as sequences of values, measured at fixed time intervals (i.e., equidistant in time). Acoustic signals on a CD, for example, are sampled 44,100 times per second, resulting in one (stereo) value every 1/44100 = 22.7 microseconds.

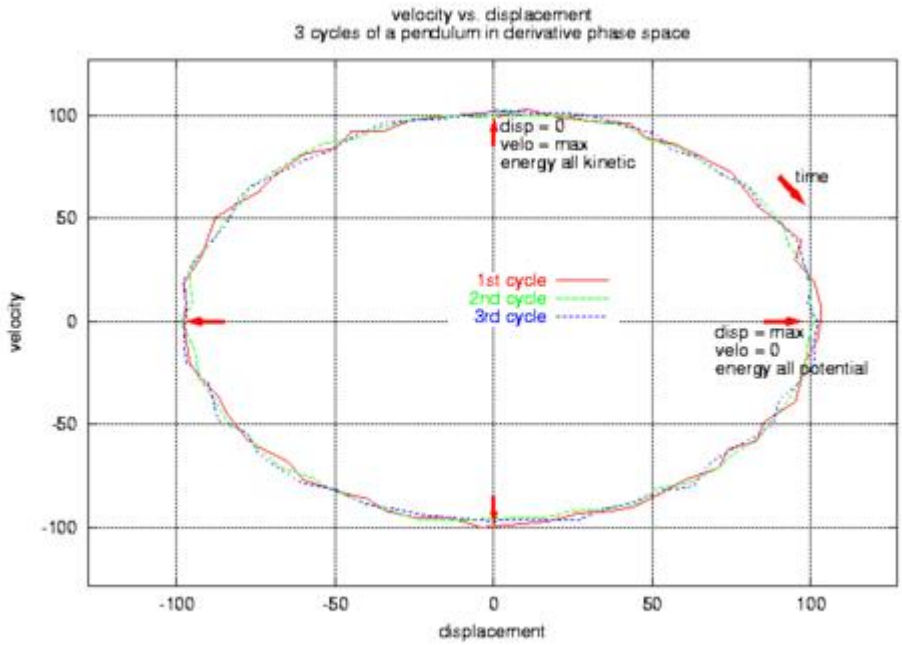


Figure 1. Phase Space Portrait of a Pendulum

While musicians may be interested in filtering these signals digitally (thereby distorting them), we are more interested in finding and analysing properties of the measured, undistorted signal. We are striving to find the rules of change governing the sampled series of values. The major tool in Nonlinear Signal Processing for finding the laws of motion is *delay coordinate embedding*, which creates a so-called *phase space portrait* from a single time series (Figures 1 and 2). If you are not interested in technical details, you may envisage it as a tool which turns a sequence of sampled numbers into a spatial curve, with each direction in space representing one independent variable. If you are interested in technical details, you will find them in the *Phase Space* sidebar. In any case, you should look at the list of FAQs of the newsgroup sci.nonlinear (see Resources). These explain the basic concepts and some misconceptions, and provide sources for further reading.

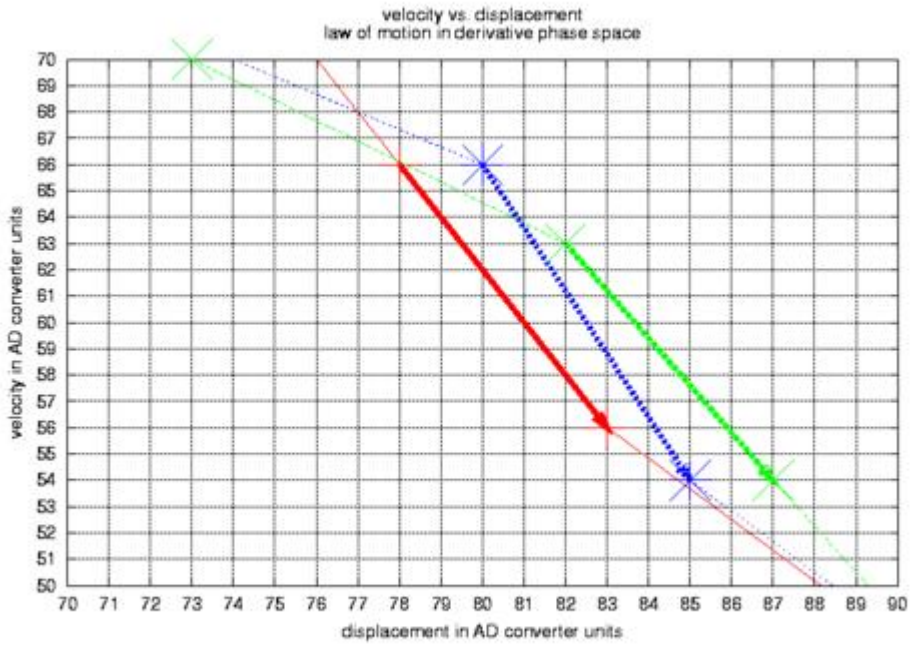


Figure 2. Progress of the Pendulum in Phase Space

Before delving into multi-dimensional space, let us look at an example. Plug a microphone into your sound card's jack, whistle into the microphone and use any of the many freely available sound editors (see Resources) to record the whistling. The resulting recording will look very much like the sine wave in Figure 3. It is not surprising that your recorded wave form looks similar to the wave form of the pendulum in Figure 3. The vibrating substances (gas or solid) may be different, but the laws of motion are very similar; thus, the same wave form.

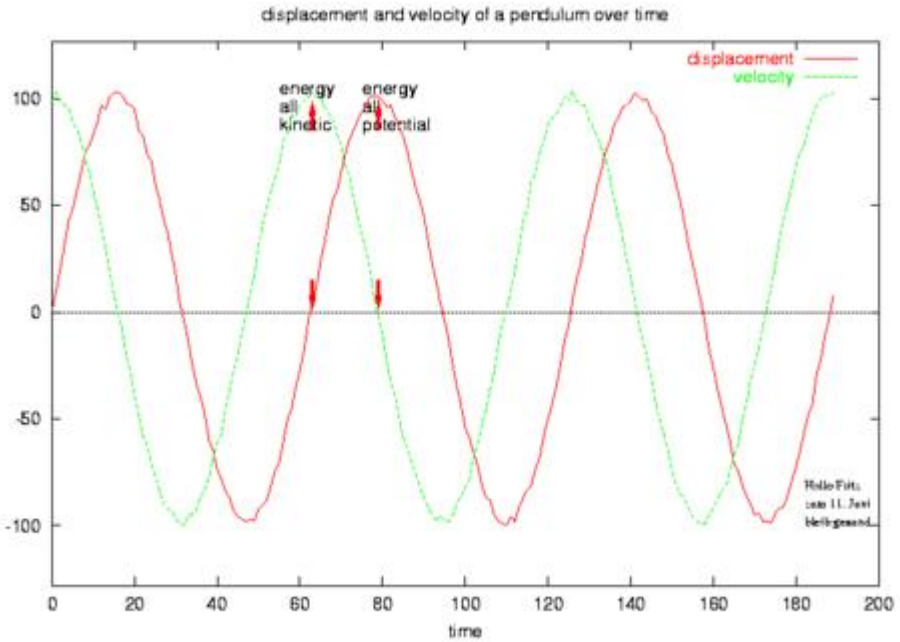


Figure 3. Wave Form of Pendulum

You can download some software from the FTP server of *Linux Journal*, which records your whistling and does the phase-space analysis for you (see Resources). Instead of displaying the wave form, the software just extracts important measures of the signal which help you refine your measurements (Figure 4). Remember, it is not my objective to show you how to present stunning, glossy pictures; rather, it is to demonstrate what a valuable tool your Linux machine is when analysing real-world signals in the field.

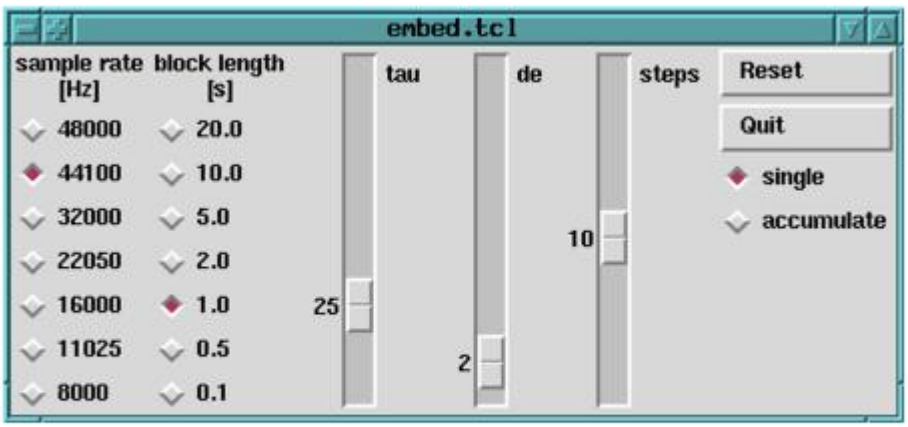


Figure 4. Finding a Good Embedding by Adjusting Independent Parameters

Start the software by typing

```
wish -f embed.tcl | dmm
```

A window will pop up that looks like Figure 4. In this window, you can control the way the software measures the whistling sound from your microphone. You can change the sample rate (click on 44,100Hz) from left to right, the length of the analysed blocks (click on 1 second) and the parameters *tau* and *de*, which are needed for reconstruction of the *phase space portrait* from just one measured signal. This is comparable to the displacement of Figure 1; we have no velocity measurement here.

Embedding

Each time we analyse an unknown signal, we have to start with some educated guessing of the two parameters:

- *tau* is the temporal distance, or delay, between spatial coordinates
- *de* is the number of dimensions of the reconstructed phase space

These parameters are of crucial importance for a good unfolding of the portrait of the signal in reconstructed phase space (see the sidebar *Embedding*). In Figure 5, you can see how the choice of tau influences the portrait. Values which are too small cannot unfold the portrait; values which are too large are not shown, but often lead to meaningless (noisy, uncorrelated) portraits. The software supports you in finding good values for *tau* and *de*. After years of intense research, scientists still rely on some heuristics for choosing suitable values for these parameters.

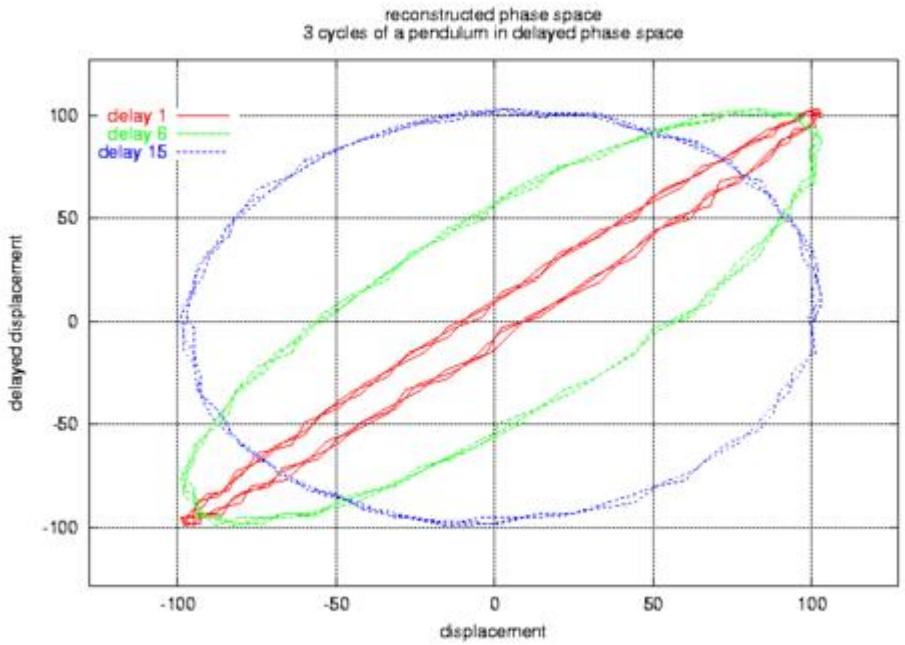


Figure 5. Differences in Unfolding, Depending on Parameter

In the particular case of your whistling, the values in Figure 4 should result in a good unfolding of the portrait. Do not worry about the text lines which fill your terminal window. They are needed for checking the quality of the unfolding. Restart the software by typing

```
wish -f embed.tcl | dmm | wish -f out.tcl
```

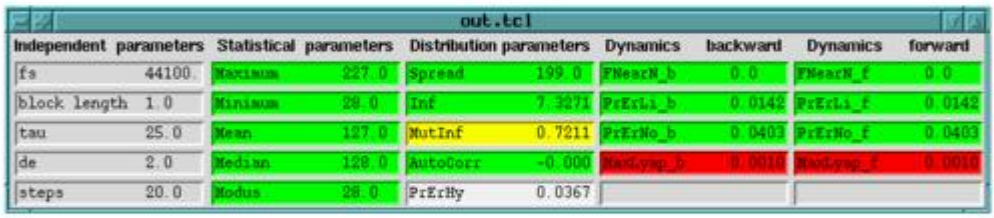


Figure 6. Parameters of a Sine Wave

and the text lines will be converted into the more readable form of Figure 6. In the additional window popping up now, you will see several columns. On the left (in grey), the independent parameters of the first window are repeated. The second column from the left tells you if the loudness of the measured signal is well-adjusted to the sensitivity of your sound card's input. The general rule is, as long as there are red lines in the second column, you have to adjust sensitivity (with a separate mixer software) or loudness. Now, turning to the third column, we see more-advanced parameters:

- *Spread* is the difference between the largest and the smallest value, measured in AD converter units. Small values indicate insufficient strength of the signal.
- *Inf* is the average information content of one sample, measured in bits. A constant baseline signal yields 0 bits (minimum) and random noise has 8 bits (the maximum with 8-bit samples).
- *MutInf* is the average mutual information of one sample and the delayed one. Thus, it tells you how *similar* the signals of both axes in Figure 5 are. A value of 1 means they are perfectly coupled (in the sense of probabilistic dependency), 0 means completely independent.
- *AutoCorr* (autocorrelation) is another measure of similarity. Since the late 1980s, there has been a (questionable) rule of thumb saying that a value near 0 indicates a good unfolding of the reconstructed portrait. The maximum is 1.

- *PrErHy* measures predictability and therefore determinism of the signal. The underlying algorithm of prediction is the conventional linear predictive filter as used in many adaptive filtering applications like modems. The minimum 0 indicates perfect (linear) predictability, while the maximum 1 indicates complete unpredictability by means of linear filtering.

Determinism, Prediction and Filtering

Again, the rule is, as long as there are red lines in the third column, phase space is not reconstructed properly. Now, turning to the last two columns, you will notice they look identical. Indeed, they are. The difference is this: when evaluating the parameters of the fourth column, the software uses a reversed time axis. When reversing time, i.e., exchanging past and future, prediction turns into postdiction and vice versa. Reversing the time axis is a simple and effective way of checking the validity of parameters which are especially susceptible to measurement errors. In general, if reversal of time changes a parameter, it is not trustworthy, which bring us to the next parameter:

- If **FNearN** (percentage of false nearest neighbours) is reliable, the lines will turn green and be the same in both columns (near 0). Otherwise, it will turn red and indicate that the neighbourship relation of points in phase space is not preserved when changing the parameter *de*, indicating an insufficient embedding.
- **PrErLi** is the result of re-calculating parameter **PrErHy** over the whole data block. They should always be roughly the same. If not, there must be a reason for it, and things get interesting.
- **PrErN** measures the predictability with a nonlinear prediction algorithm. Signals originating from a linear system are usually predicted more precisely by *PrErLi* while signals from nonlinear sources are often predicted better by nonlinear prediction.
- **MaxLyap** measures separation (progressing over time) of points nearby in phase space. By definition, values larger than 0 indicate chaos.

When measuring signals from nonlinear systems, *PrErLi* often turns red (indicating insufficient linear predictability) while *PrErN* stays green (indicating sufficient nonlinear predictability). In case of a truly chaotic signal, *MaxLyap* will turn green (valid) and have opposite signs on the right-most columns. This indicates nearby points are separating over time when time is going forward, and they are approaching each other at the same rate when moving backward in time.

Embedding

For the moment, the number of parameters and values may be overwhelming. If you start by playing with the software and actually analysing some signals in the field, you will soon become acquainted with the parameters in their colours and columns. The first time, you should look only at the two left-most columns in Figure 6. All parameters there have intuitive meanings, and you will soon be able to foretell how they change when applied to a different signal, a clipped signal or an oversampled signal. Here are some typical situations and how to recognize them:

- Sine wave: just as in Figure 6, *de* (embedding dimension) should be 2 or 3. *Mean* (i.e., average) and *Median* (i.e., "middlest") are the same. *Modus* is jumping back and forth between *Maximum* and *Minimum*. If the *Spread* reaches its maximum (256), *Inf* gets near 8 (bits).
- Zero baseline (short-circuit or switched off) can be recognized by looking at column 2. All values are identical. In column 3, *Spread* and *Inf* are almost 0.
- Switching on a microphone, there is a short and sharp impulse resulting in a sudden change of *Spread*; few others change.
- Sawtooth (Figure 7) looks much like the sine, except for *Modus*, which jumps wildly. *MutInf* is at its maximum, linear prediction works only with higher-order filters, while nonlinear prediction works better with low embedding dimensions.
- Noise comes in many different flavours, all of them having low values of *AutoCorr* and most with a low *MutInf*.

Figure 7. Parameters of a Sawtooth Signal

Why not calculate some kind of *fractal dimension* of a signal? By definition, calculation of dimensions must look at the values over a wide range of scales. With 8 bits of resolution, this is impossible or questionable. But even if we had some *fractal dimension* value, it would not be as useful as the largest Lyapunov exponent. Furthermore, are all these measurements any good? Yes, there are some areas of application:

- **System Identification**: in some applications, the focus of attention is more on the quality of the signal (stochastic or deterministic, linear or nonlinear, chaotic or not).
- **Prediction**: today, linear prediction is one of the most important algorithms for digital signal processors (DSPs) in telecommunication, be it mobile telephony, modems or noise canceling. If there are systems with nonlinear behaviour involved, nonlinear prediction can be advantageous (see sidebar "Determinism, Prediction & Filtering").
- **Control**: if you know the structure of your system's phase space well enough, you can try to control the system like this:

1. Identify periodic orbits in phase space.
2. Look for an orbit which meets the given requirements (goes through a certain point, or has minimum energy or cost).
3. Modify a suitable parameter or a variable just slightly to stabilize the desired periodic orbit.
4. When Ott, Grebogi & Yorke first published successful application of this method (called the OGY method) in 1990, they even managed to control a system in the presence of chaos.

Projects

In the late 1990s, several people analysed the time series of the financial markets in order to find signs of nonlinearity or chaos (for example, Blake LeBaron in Weigend & Gershenfeld's book, page 457). Some hoped to be able to predict time series of stock values in this way. Kantz & Schreiber took the idea one step further, and contemplated the application of the OGY method to control the stock market. But in a footnote on page 223 of their book, they admit, "We are not aware of any attempts to control the stock market as yet, though." When I looked at the chart of Red Hat stock in late 1999, I wondered whether someone had finally managed to apply the OGY method to the time series of the Red Hat share price.

Resources

email: Juergen.Kahrs@t-online.de

Juergen Kahrs (Juergen.Kahrs@t-online.de) is a development engineer at STN Atlas Elektronik in Bremen, Germany. There, he uses Linux for generating sound in educational simulators. He likes old-fashioned tools like GNU AWK and Tcl/Tk. Juergen also did the initial work for integrating TCP/IP support into **gawk**.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

THOR: A Versatile Commodity Component of Supercomputer Development

Robert A. Davis

Issue #75, July 2000

CERN continues to use Linux as their OS of choice for modeling and simulation studies.

The world's highest energy particle accelerator, the Large Hadron Collider (LHC), is presently being constructed at the European Center for Particle Physics Research (CERN) near Geneva, Switzerland. The planned date for first collisions is 2005. Since the demise of the US Superconducting Super Collider (SSC) in 1993, CERN has essentially become a world laboratory where American, African, European, Asian and Australian physicists work side by side. The LHC will penetrate deeper than ever into the microcosm to recreate the conditions prevailing in the universe just a millionth of a millionth of a second after the big bang when the temperature was ten-thousand-million-million degrees.

Our group is a small part of the team of approximately 1500 physicists, from over 100 institutions around the world, engaged in the construction of the ATLAS (A Toroidal LHC ApparatuS) experiment, one of two general-purpose detectors preparing to take data at the LHC. The experimental environment of ATLAS is punishing. For example, ATLAS has hundreds of thousands of detector channels and must keep up with a collision rate that can give rise to approximately 30 new events every 25 nanoseconds. Also, detectors and their accompanying electronics often must operate in high-radiation environments. It is obvious that the computing requirements in such an arena are, to say the least, demanding. CERN is no stranger to software developments required to solve the unique problems presented by international particle physics. For example, the World Wide Web was initially designed at CERN to help communication among the several hundred members scattered in numerous research institutes and universities.

Design Considerations

The particle physicists in our group are involved in two areas which pose large computing problems. The first is in the area of time-critical computing, where the raw rate must be reduced from an event rate of around one gigahertz to about 100Hz by a three-stage, real-time data selection process called triggering. We are involved, along with groups from CERN, France, Italy and Switzerland, in the final stage of triggering, called the Event Filter, that reduces the data rate from 1GB/s to 100MB/s, fully reconstructs the data for the first time and writes the data to a storage medium. It is estimated that this last stage of processing would currently require on the order of a thousand “Pentiums”, if current trends in the development of processor speed continue.

We are also actively involved in simulating the response of the ATLAS detector to the physics processes that will be, or might be, present. This second task is not time-critical, but requires large simulation programs and often many hundreds of thousands of fully simulated events. Neither of these applications requires nodes to communicate during processing.

In order to pursue our research aims in these two areas, we had to develop a versatile system that could function as a real-time prototype of the ATLAS Event Filter and also be able to generate large amounts of Monte Carlo data for modeling and simulation. We needed a cost-effective solution that was scalable and modular, as well as compatible with existing technology and software. Also, because of the time scale of the project, we required a solution with a well-defined and economical upgrade path. These constraints led us inevitably toward a “Beowulf-type” commodity-component multiprocessor with a Linux operating system. The machine we finally developed was called THOR, in keeping with the Nordic nature of the names of similar-type systems such as NASA's Beowulf machine and LOKI at Los Alamos National Laboratory.

During our design discussions on THOR, it soon became clear to us that the benefits of scalability, modularity, cost-effectiveness, flexibility and access to a commercial upgrade path make the commodity-component multiprocessor an effective approach for providing high-performance computing for a myriad of scientific and commercial tasks—capable of being utilized for both time-critical and off-line data acquisition and analysis tasks. The combination of commodity Intel processors with conventional fast Ethernet and a high-speed network/ back-plane fabric (Scalable Coherent Interface (SCI) from Dolphin Interconnect Solutions Inc.) enables the THOR machine to run as a cluster of serial processors, or as a fully parallel multiprocessor using MPI. It is also possible to rapidly reconfigure the THOR machine from a fully parallel mode to an all-serial mode, or for mixed parallel-serial use.

The THOR Prototype

In order to demonstrate the basic ideas of the THOR project, a prototype has been constructed. A photograph of a slightly earlier incarnation of THOR is shown in Figure 1. This prototype at present consists of 42 dual Pentium II/III MHz machines (40 450MHz and 44 600MHz processors), each with 256MB of RAM. Each node is connected via a 100Mb/s Ethernet 48-way switch. A 450MHz dual Pentium II computer provides the gateway into the THOR prototype. The prototype has access to 150 gigabytes of disk space via a fast/wide SCSI interface and a 42-slot DDS2 tape robot capable of storing approximately half a terabyte of data. The THOR prototype currently runs under Red Hat Linux 6.1.

Figure 1. The THOR Commodity Component Multiprocessor

Sixteen of the 40 nodes have been connected into a two-dimensional 4x4 torus, using SCI, which allows a maximum bi-directional link speed of 800MB/s. We have measured the throughput of the SCI to be 91MB/s, which is close to the PCI bus maximum of 133MB/s. This maximum will rise when the 64-bit version of the SCI hardware, in conjunction with 64-bit PCI bus widths, are available. The use of SCI on THOR permits the classification of these THOR nodes as a Cache Coherent Non-Uniform Memory Access (CC-NUMA) architecture

machine. This 16-node (32-processor) subdivision of the THOR prototype was implemented and tested as a fully parallel machine by a joint team from Dolphin Interconnect Solutions Inc. and THOR in the summer of 1999. A schematic diagram of the THOR Linux cluster is shown in Figure 2.

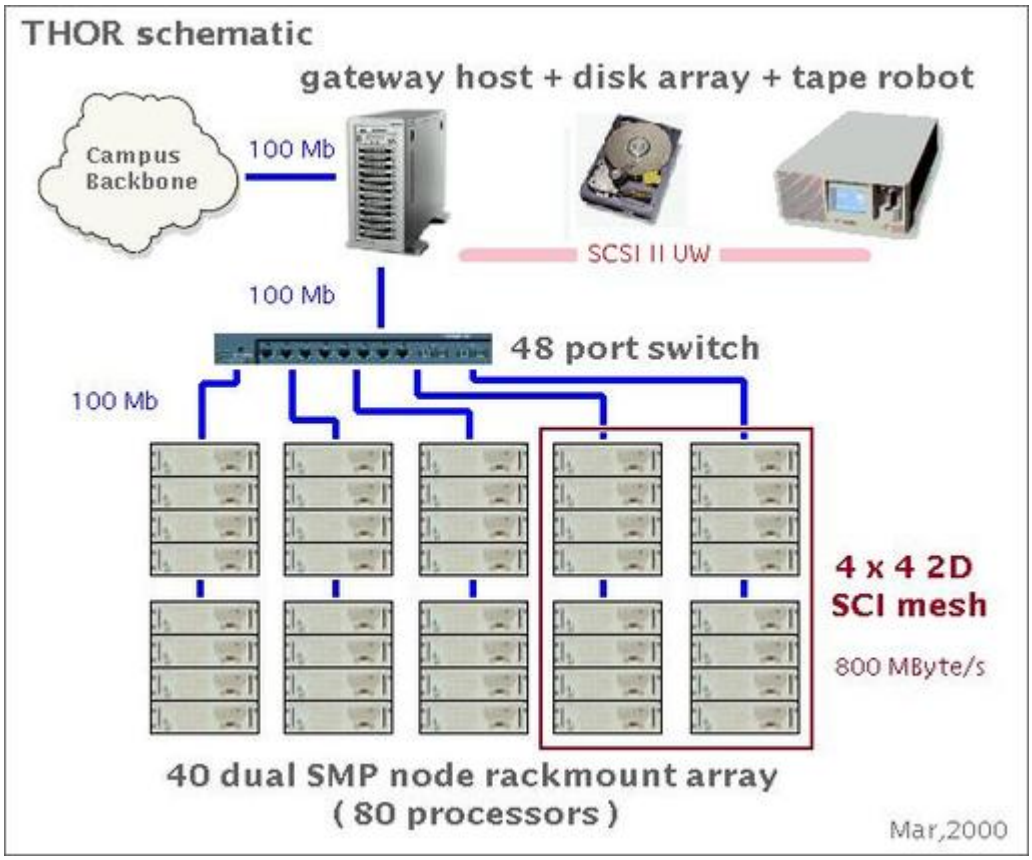


Figure 2. Schematic Diagram of the THOR Linux Cluster

The THOR prototype described above is now being benchmarked as a parallel and serial machine, as well as being used for active physics research. Researchers have access to full C, C++ and FORTRAN compilers, CERN and NAG numerical libraries and MPI parallel libraries for their research use. We also plan to acquire the recent Linux release of IRIS Explorer for THOR research use in the near future. PBS (a Portable Batch System developed at NASA) has been running on THOR since March, 1999.

Benchmarking

We benchmarked the THOR prototype extensively, when it consisted of 20 450MHz dual Pentium IIs. This benchmarking used a variety of software. The first of two benchmark programs reported here was a three-dimensional Fast Fourier Transform (FFT) program written in FORTRAN using MPI. This software uses the THOR cluster as a parallel machine rather than a serial processor. As can be seen in Figure 3, the speed of a single THOR node using this FFT is comparable to a 450MHz Cray-T3E and the Calgary DEC Alpha cluster (in 1999) and is roughly 50% slower than an SGI Origin 2000 (R10K). As a fully parallel

machine, one can see that THOR is competitive with the Calgary DEC Alpha cluster and with the Cray-T3E for this FFT procedure.

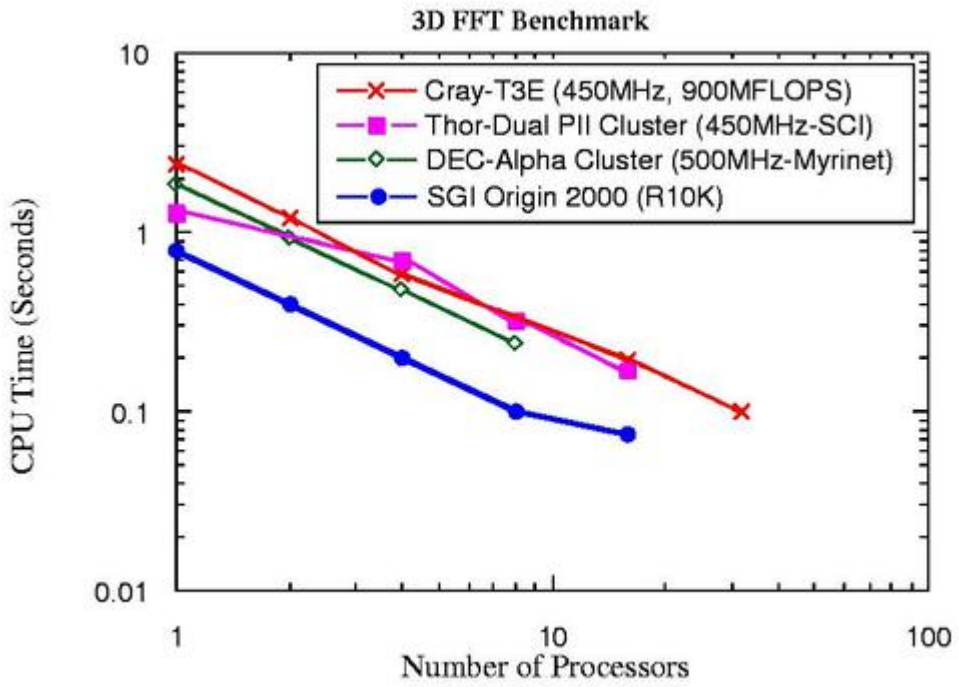


Figure 3. Benchmark Results from a 3-D Fast Fourier Transform Program

The second benchmark was obtained using the code developed to utilize the method of finite differences in the area of seismic modeling, and is written in C using MPI. In this case, the comparison was between the THOR multiprocessor using SCI interconnect, or 100 Mbp/s Ethernet and an SGI Origin 2000. The results are shown in Figure 4. As can be seen, the use of the SCI backplane/ network fabric improves the performance of the THOR multiprocessor as compared to an Ethernet network solution. According to this benchmark, the performance of the THOR multiprocessor using SCI becomes comparable to that of the SGI Origin 2000 as the number of processors approaches sixteen.

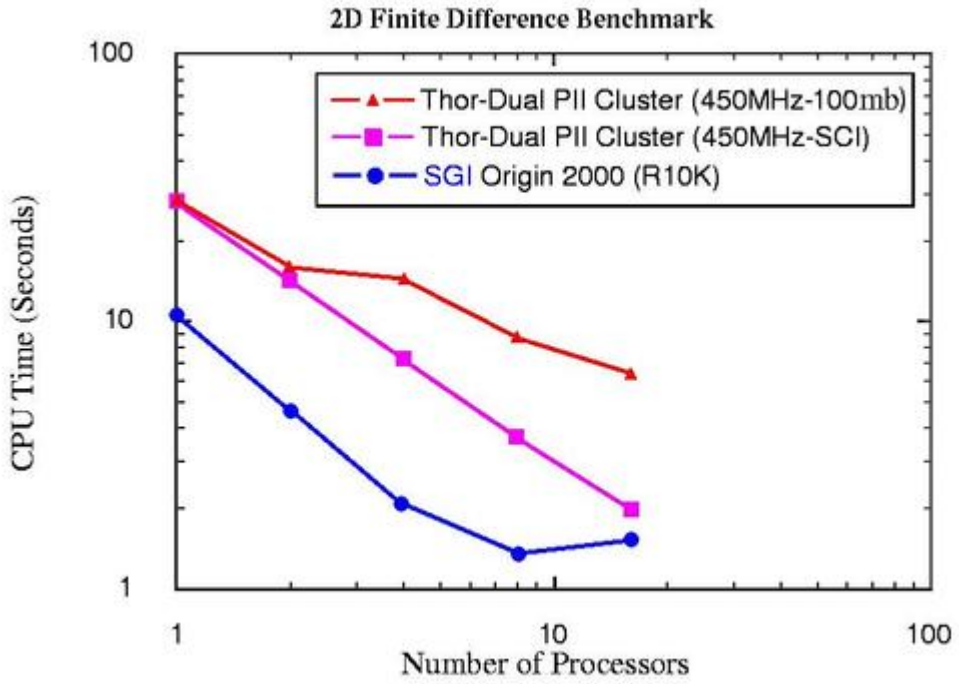


Figure 4. Benchmark Results from a Finite Differences Program for Seismic Modeling

Our Experience with THOR

Over the last year and a half, THOR has grown from just two dual Pentium II machines to over 40 dual Pentium II/IIIs. We found that the maintenance and operation of a two-node array was not much different from running a 40-node array. Another gratifying feature, which has been reported by other groups with Beowulf-type clusters, is the reliability of the THOR cluster. We have been running the PBS batch queuing system since March, 1999, and have logged over 80,000 CPU hours with only one system failure due to a power interruption, which led us to introduce non-interruptible power supplies for the complete cluster. Another important discovery was that the construction and maintenance of THOR did not need a team of highly skilled personnel. We found that only “one quarter” of a person skilled in PC networking and Linux was required to implement the system, with some initial assistance from Dolphin for the SCI network. Also, the use of commodity operating systems allows programs to be developed at researchers' desktop machines for later implementation on THOR, thus easing the task of software development.

Our experience running THOR has been in three main areas: multiple-serial Monte Carlo (or embarrassingly parallel) production jobs, prototyping an Event Filter sub-farm for ATLAS (a time-critical operation using specialized software developed for the ATLAS high-level trigger system) and as a fully parallel processor. Large Linux clusters are probably most effective when running applications that require almost no inter-processor communication where the network does not become a bottleneck. However, many applications that

require fully parallel machines for significant message passing still spend most of their time computing.

Because parallel computing is such an important topic, we spent some time assessing how the THOR multiprocessor running Linux can be used to implement applications requiring parallel processing. Since there is currently a high level of support for shared-memory programming under SMP Linux, we have used the THOR Linux cluster with the message-passing construct, Message Passing Interface (MPI). There are several advantages to using this programming model. For example, many parallel applications are limited more by floating-point performance than by inter-processor communication. Thus, SCI and even fast Ethernet are sufficient even when there is a relatively large amount of message passing. Another advantage is that MPI is a standard programming interface that runs on many parallel machines, such as the Cray T3 series, IBM SP series, SGI Origin, Fujitsu and PC/Macintosh clusters. Therefore, code can be moved easily between platforms.

One of the THOR groups involved in Plasma Physics research has developed simulations that follow the motion of about 12 million charged particles. A portion of this code utilizes multi-dimensional Fast Fourier Transforms. We achieved nearly perfect scalability for up to 32 processors (all that was available at the time) on the THOR cluster. The code, following the MPI model, was easily transferred from an SGI Origin 2000 platform to the THOR Linux cluster. The performance of this program on THOR compared with other platforms as shown in Figure 3. Other benchmarks are given in Figure 4. At the time these benchmarks were performed, THOR consisted of only 450MHz machines.

Cost of Ownership

Our experience with the THOR Linux cluster described above shows that if we divide the total cost of the machine by the number of processors, we end up with a cost of around $1,500 (CDN) per processor. This is cheaper than conventional supercomputers by more than a factor of ten, assuming reasonable discounts apply. Although there are certainly applications in which conventional supercomputers are irreplaceable, on a price-performance basis, THOR (or Beowulf)-type multiprocessors are more attractive. Another cost advantage of the THOR Linux cluster is the low software cost. GNU's compilers and debuggers, along with free message-passing implementations (MPI) and portable batch-queuing system (PBS), with no yearly fees, offer good low-cost solutions. Better compilers including FORTRAN90, such as the Absoft product, offer significant performance enhancements and debugging tools in the MPI environment.

The comparatively small upfront costs of the THOR Linux cluster are matched by its low running costs. Our experience indicates that, at least for machines as

large as THOR, the manpower costs involved with running the machine are low. For example, THOR requires only approximately 30% of the time of a networking/Linux expert. We think this is due to the reliability, design simplicity and accessibility of the commodity component multiprocessor approach. New nodes can be added to THOR on the fly without rebooting any machines; also, problem nodes can be hot-swapped. The node being a conventional PC, probably with a one-year warranty, can either be repaired or thrown away. In fact, hardware and software maintenance costs for THOR have proven to be negligible compared to the annual maintenance fees required by most conventional supercomputer producers. Such fees can be in excess of tens of thousands of dollars per year. The advantages of Beowulf-type clusters like THOR, running Linux, are so numerous that we are not surprised that more and more scientific and commercial users are adopting this approach.

Resources

Acknowledgements

email: pinfold@phys.ualberta.ca

James Pinfold (pinfold@phys.ualberta.ca) is Director of the Centre for Subatomic Research at the University of Alberta and leader of the THOR Linux cluster, a commodity component supercomputer project. His main research effort is in the area of high-energy collider physics, where he is currently working on the OPAL and ATLAS experiments at the European Centre for Particle Physics (CERN) near Geneva, Switzerland.

Archive Index Issue Table of Contents

Advanced search

A GNU/Linux Wristwatch Videophone

Steve Mann

Issue #75, July 2000

This fully fuctioning prototype, designed and built by Steve Mann in 1998, was demonstrated in 1999, and later used to deliver a videoconference at ISSCC 2000.

Videophone wristwatches are a science-fiction concept that is here today. The two key inventive concepts that make this new technology possible are:

- The use of a body-worn computer system (WearComp) as a base station (see my article “University of Toronto WearComp Linux Project” in *LJ*, February 1999). Images from the wristwatch are sent to the WearComp, and from the WearComp to the Internet. Images received from the Internet are sent to the wristwatch display (a full-colour VGA display). Full-colour broadcast quality is transmitted at six to eight frames per second using an experimental radio transmitter.
- The use of a concomitant cover activity. Unlike science fiction's vision of how a wristwatch videophone might work, the camera points ahead rather than up at the user. In this way, the wristwatch captures a video image of what the wearer is looking at, rather than merely a picture of the wearer. Thus, taking a picture or shooting a video may be masked by a concomitant cover activity, such as checking the time or just resting an arm on a countertop (see sidebar “Concomitant Cover Activity”).

Concomitant Cover Activity

A VGA screen is configured using XF86Config set to 640x480, 24-bit colour, allowing video to be displayed at the captured resolution. The camera also operates at 640x480 resolution with 24-bit colour video capture at 30 frames per second. Images may be processed or stored locally, or transmitted at a lesser rate. A future version will transmit images at the recording speed of 30 frames per second rather than the current six to eight frames per second limit imposed by the slow (2.4 megabits per second) radio link.

The wristwatch provides a computer output screen with XF86, upon which the viewfinder function operates, using one of the windows or the root window. Graphics, including a transparent oclock, appear over the top of the video viewfinder window.

Using the VideoOrbits image stabilizer, it takes pictures at 640x480, 24-bit colour, up to 5000 pixels across, in true 48-bit colour (convertible to 24-bit colour pictures suitable for high-quality prints). VideoOrbits is available under the GPL from http://wearcam.org/orbits/.

Figure 1. A Wristwatch Videoconferencing Computer

In Figure 1, Eric Moncrief is shown wearing the watch, and Stephen Ross is pictured on the XF86 screen as a 24-bit true-colour visual.

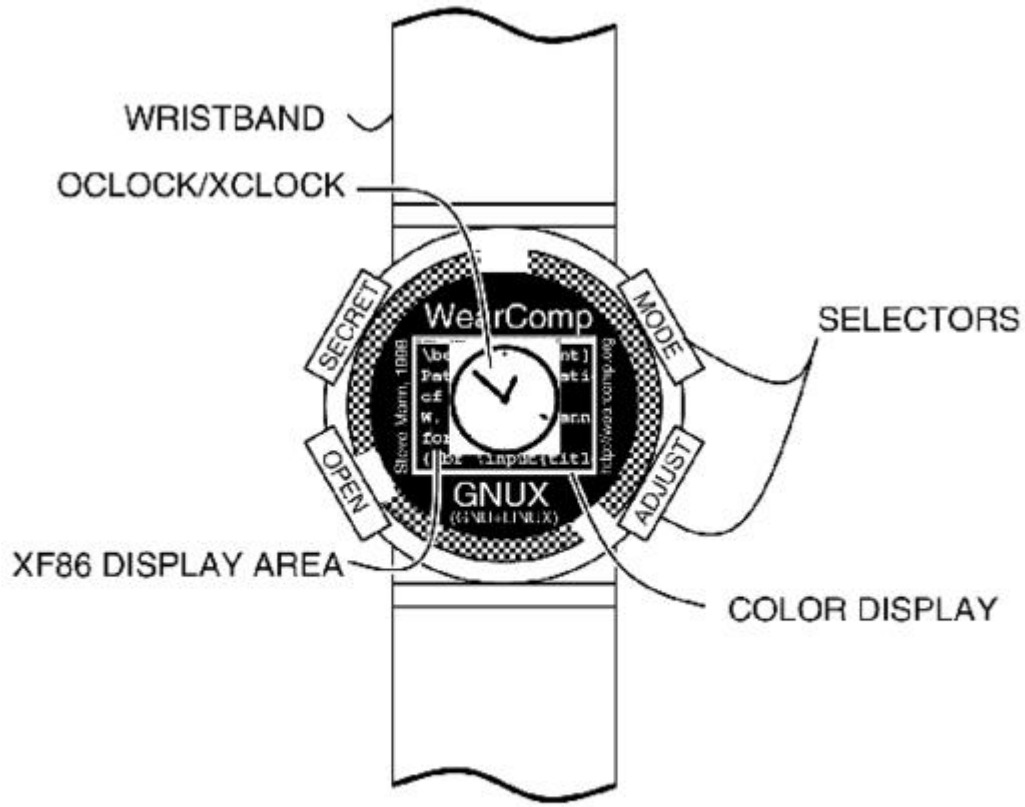


FIG. 2: GNUX WRISTWATCH
VIDEOPHONE CLOCKFACE

Figure 2. Clock Face

A SECRET function, when selected, conceals the videoconferencing window by turning off the transparency of the oclock, so that the watch then looks like an ordinary watch (showing just the clock filling the entire 640x480-pixel screen). The OPEN function cancels the SECRET function and opens the videoconferencing session up again.

Technical Problems and How They Were Overcome

One technical problem that arises from running GNU/Linux (GNUX) on a wristwatch is the input. We are experimenting with an input pie menu system. A user can easily select eight directions of the compass, but since this device has a clock face (at least, that is its concomitant cover usage), a 12-position pie menu makes the most sense.

The pie menu is described in “A Comparative Analysis of Pie Menu Performance” by Callahan, Hopkins, Weiser and Shneiderman, 1988.

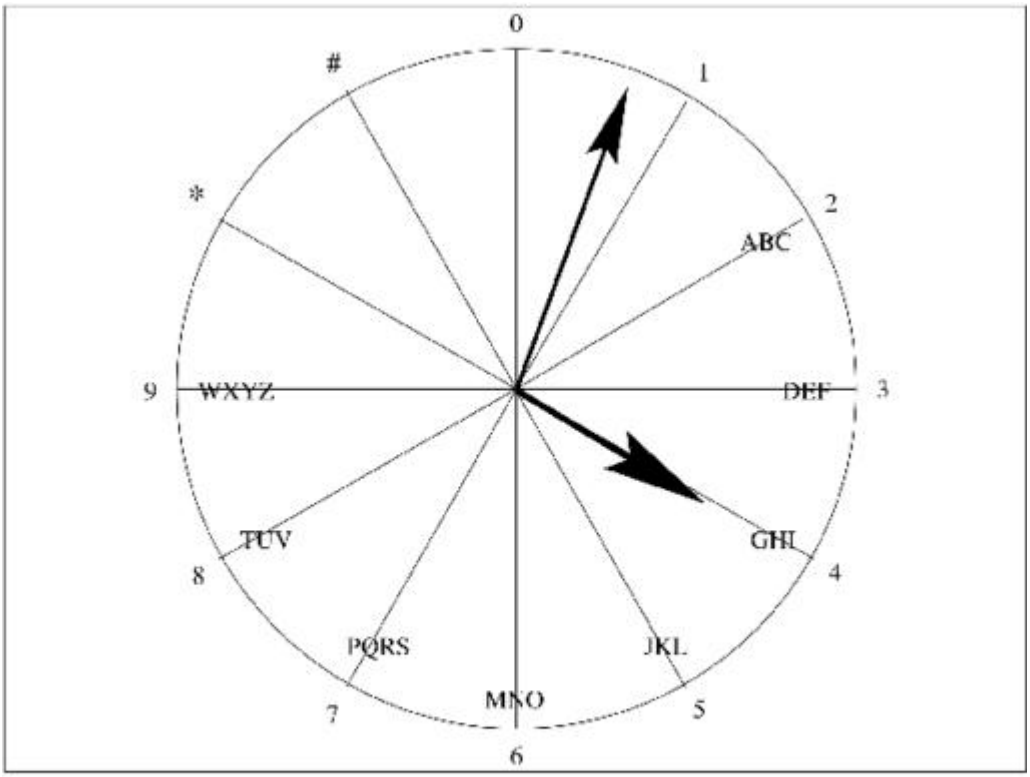


FIGURE 3: WRISTWATCH VIDEOCONFERENCING MENU

Figure 3. Wristwatch Videoconferencing Menu

Figure 3 depicts a natural choice of pie menu for a wristwatch display. The display is typically a computer screen with 480 pixels down and 640 pixels across, measuring approximately 0.7 inches on the diagonal. Upon the display is the image of a clock face, superimposed on top of a video signal from the camera. Time is displayed as a transparent xclock or oclock (or both, one superimposed upon the other). Our modified oclock is available from http://wearcam.org/gclock/ and an exclusive or (EOR) oclock is under development to reduce screen real estate use. In the figure depicted here, the time is 4:03.

The device truly looks like an ordinary wristwatch, although one in which the hands are displayed electronically, because it is in fact a wristwatch, among other things. It is natural for such a wristwatch to have a circle displayed on the screen (this is a feature of the original oclock), but unlike the oclock, it has numbers displayed around the periphery of the circle. In this way, it is easier to tell time, and the numbers may also be assigned a secondary meaning (e.g., select “0” to stop recording, “4” to kill all processes and halt the processor, “7” to wake up the system from sleep mode, etc.).

Since humans are quite good at telling time, the numbers are often missing from commercial wristwatches, and some wristwatches do not even have markings for each hour. Instead, we often rely on our heightened sense of

visual acuity to discern the angle of the hands upon the clock face. Thus, it is no surprise that the clock menu is usable without paying much attention to the face of the clock. The user just needs to stroke the face of the clock in the direction desired.

The entry of numbers on a touch-sensitive clock face in the context of the current invention may be done as vectors (e.g., with no regard to location, only to direction). Thus, a stroke from left to right is regarded as the number 3, regardless of where the stroke begins or ends. A downward stroke (e.g., from top to bottom) is regarded as the number 6 regardless of where the stroke begins or ends, and so on.

Thus, telephone numbers can easily be entered into the device, and similarly, an alphabet can be constructed much like the alphabet of an automated DTMF (dual-tone multi-frequency) answering system used for voice mail and the like in telephony.

Since there are 12 pushbuttons on a telephone and also 12 hours on a clock face, there can be a one-to-one correspondence between the numbers of the clock face and those of the telephone. The hours 10:00 and 11:00 are used for the symbols “*” and “#” of the telephone touchpad.

The data entered by way of the clock face menu is typically combined with the video recording made from the scene. The clock face menu is sufficient for entering a department store manager's name, which may be appended to the video file header, so that a large database of recorded video may be navigated later using these short text headers.

Going Further

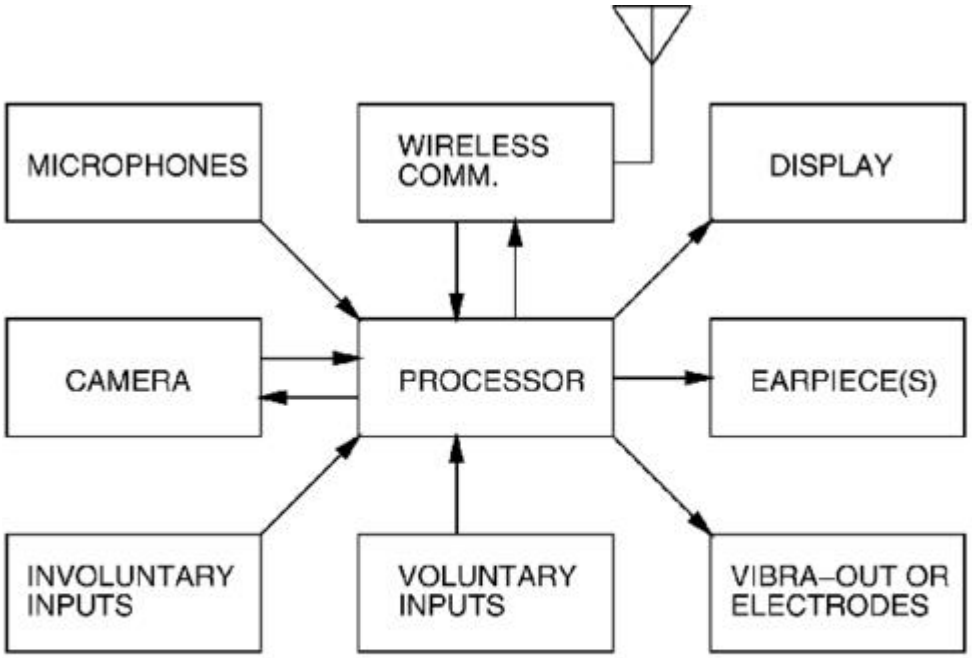


FIG 4: BLOCK DIAGRAM OF WRISTWATCH VIDEO PRODUCTION FACILIT

Figure 4. Block Diagram of Video Production Facility

Due to direct contact between the wristwatch and the body, the pulse (heart rate) as well as skin conductivity (sweatiness index) of the wearer may be determined, and this information may be appended to or recorded with the video signals. This may facilitate, for example, a future search through all video in which the wearer's heart rate exceeds a certain threshold. It has been found that when a department store manager is dishonest with respect to refund policies, or a clerk refuses to tell a customer his name, the customer's heart rate increases dramatically, and the customer often sweats profusely. Thus, this extra information can later help locate moments of tension in a previously recorded argument at the refund counter.

Programs Developed for the Wristwatch Videophone

Linaccess: GNUX for Low Vision

The small size of the display required the development of an X Window System configuration that was easy to read on a small screen. This gave rise to the Linaccess project, where GNUX was made accessible on low-vision systems. This project has two distinct but closely related goals. First, to make GNUX accessible to the visually challenged, or those suffering from low vision, such as age-related macular degeneration, glaucoma or the like. (Note that this project differs from the blinux project in the sense that the goal of Linaccess is to use visual output, but to make it more accessible to those with low vision.) Second, to make GNUX usable on small screens, such as the wristwatch system. In

many ways, we're all suffering from low vision when we're trying to read a 0.7-inch diagonal screen.

Resources

Dr. **Steve Mann** (mann@eecg.toronto.edu) is regarded by many as the inventor of the wearable computer (computing being distinct from special-purpose devices such as wristwatches and eyeglasses), and of the EyeTap video camera and reality mediator. He also built the world's first covert fully functional WearComp with display and camera concealed in ordinary eyeglasses in 1995, for the creation of his award-winning documentary “ShootingBack”. He is also the inventor of the wristwatch videophone, the chirplet transform, a new mathematical framework for signal processing, and of comparametric image processing. He is currently a member at University of Toronto, Department of Electrical and Computer Engineering.

Archive Index Issue Table of Contents

Advanced search

Three-Tier Architecture

Ariel Ortiz Ramirez

Issue #75, July 2000

Professor Ortiz presents a little of the theory behind the three-tier architecture and shows how it may be applied using Linux, Java and MiniSQL.

In the beginning, there were mainframes. Every program and piece of data was stored in a single almighty machine. Users could access this centralized computer only by means of dumb terminals. (See Figure 1.)

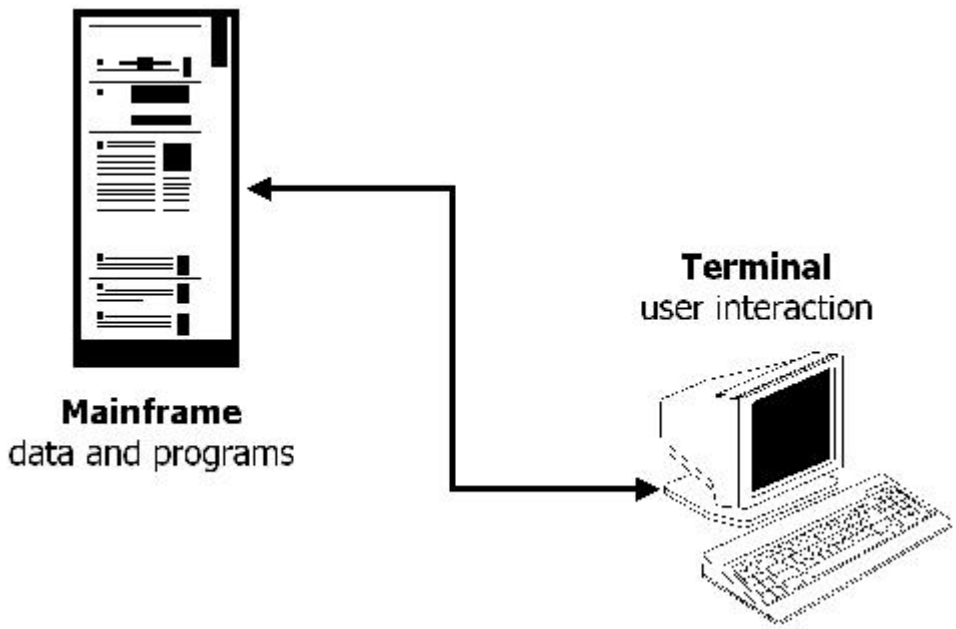


Figure 1. Mainframe Architecture

In the 1980s, the arrival of inexpensive network-connected PCs produced the popular two-tier client-server architecture. In this architecture, there is an application running in the client machine which interacts with the server—most commonly, a database management system (see Figure 2). Typically, the client application, also known as a fat client, contained some or all of the presentation logic (user interface), the application navigation, the business rules and the database access. Every time the business rules were modified,

the client application had to be changed, tested and redistributed, even when the user interface remained intact. In order to minimize the impact of business logic alteration within client applications, the presentation logic must be separated from the business rules. This separation becomes the fundamental principle in the three-tier architecture.

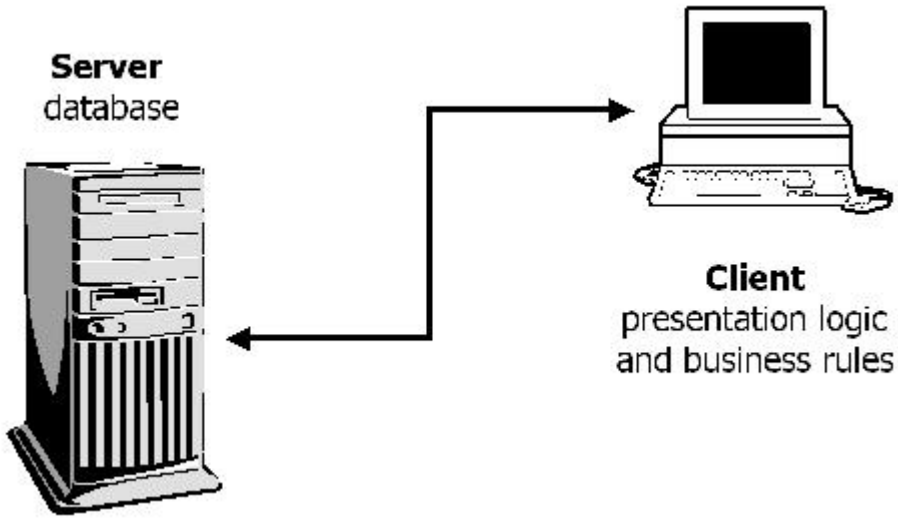


Figure 2. Two-Tier Client-Server Architecture

In a three-tier architecture (also known as a multi-tier architecture), there are three or more interacting tiers, each with its own specific responsibilities (see Figure 3):

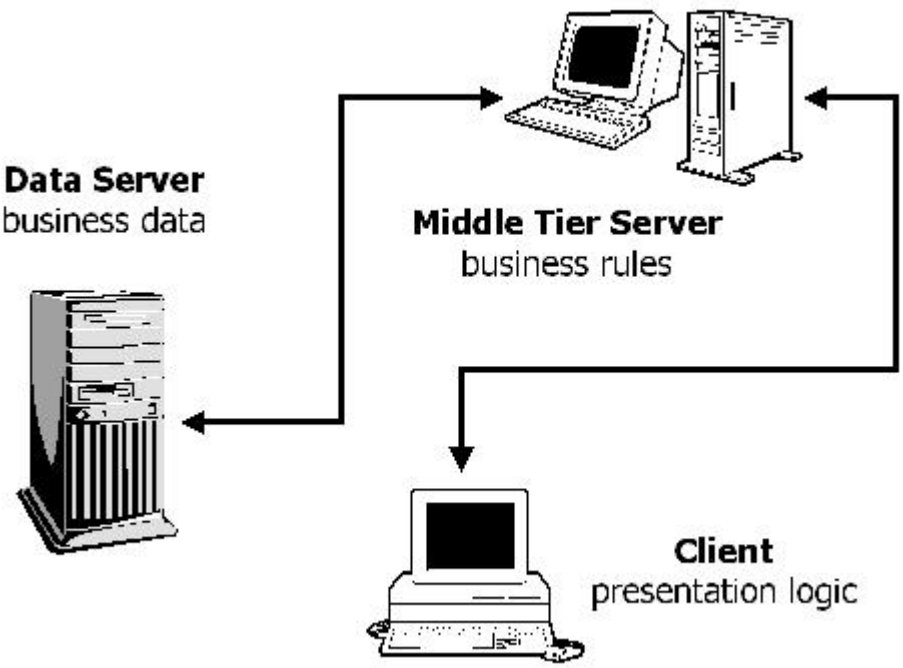


Figure 3. Three-Tier Architecture

- Tier 1: the client contains the presentation logic, including simple control and user input validation. This application is also known as a thin client.
- Tier 2: the middle tier is also known as the application server, which provides the business processes logic and the data access.
- Tier 3: the data server provides the business data.

These are some of the advantages of a three-tier architecture:

- It is easier to modify or replace any tier without affecting the other tiers.
- Separating the application and database functionality means better load balancing.
- Adequate security policies can be enforced within the server tiers without hindering the clients.

Putting the Theory into Practice

In order to demonstrate these design concepts, the general outline of a simple three-tier "Hangman" game will be presented (check the source code in the archive file). The purpose of this game, just in case the reader isn't familiar with it, is to try to guess a mystery word, one letter at a time, before making a certain number of mistakes.

Figure 4. Hangman Client Running in Windows 98

The data server is a Linux box running the MiniSQL database management system. The database is used to store the mystery words. At the beginning of each game, one of these words is randomly selected.

At the client side, a Java applet contained in a web page (originally obtained from a web server) is responsible for the application's graphical user interface (see Figure 4). The client platform may be any computer with a web browser that supports applets. The game's logic is not controlled by the applet; that's the middle tier's job. The client only takes care of the presentation logic: getting the user's input, performing some simple checking and drawing the resulting output.

The server in the middle tier is a Java application, also running within a Linux box. The rules of the “Hangman” game (the business rules) are coded in this tier. Sockets and JDBC, respectively, are used to communicate with the client and the data server through TCP/IP.

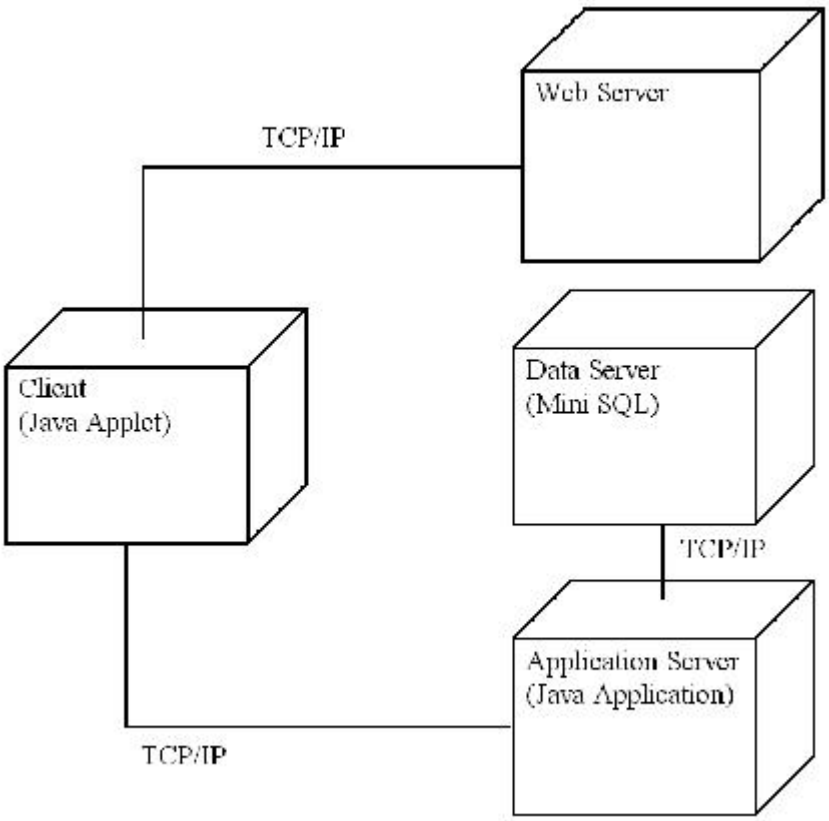


Figure 5. Diagram of Hardware Nodes

Figure 5 presents a UML (Unified Modeling Language) deployment diagram that shows the physical relationship among the hardware nodes of the system.

Even though the design described gives the impression of requiring a different machine for each tier, all tiers (each one running on a different process) can be run in the same computer. This means the complete application is able to run

in a single Linux system with a graphical desktop, and it doesn't even have to be connected to the Net!

Running MiniSQL

MiniSQL, developed by Hughes Technologies, is an exceptionally fast DBMS with very low system requirements. It supports a fairly useful subset of the Structured Query Language (SQL). Using it for commercial purposes requires purchasing a license, although free licenses are provided for academic and charity organizations.

The software is distributed in source code form, all bundled up in a gzipped tar file (currently, the latest distribution file is msql-2.0.11.tar.gz). It may be downloaded from the Hughes Technology web site (see Resources). The MiniSQL manual, with all the necessary installation and usage information, is contained in the files msql-2.0.11/doc/manual.ps.gz and msql- 2.0.11/doc/ manual-html/manual.html, once the distribution file is extracted. The reader is encouraged to carefully review and follow the instructions contained there. However, it must be noted that two important details are missing from the MiniSQL manual:

- The “system” section contained in the /usr/local/Hughes/msql.conf file has a parameter called Remote_Access that has a default value of **false**. It must be changed to **true** in order to allow access to the database from remote systems.
- Like other server dæmons (for example, the HTTP web server), the MiniSQL 2.0 server, called msql2d, should be run as a background process. Executing the following command as root should achieve this: **/ usr/local/Hughes/bin/msql2d &**

In addition to the database server, MiniSQL comes with some other useful utilities: a server administration program, an interactive SQL monitor, a schema viewer, a data dumper and a table-data exporter and importer. The server administration program is required to create the Hangman database that will contain the mystery words. The following command must be executed as root:

```
/usr/local/Hughes/bin/msqladmin create hangman
```

Afterward, a mystery-words table needs to be created. Only two columns will be contained in this table: word (the mystery word or sentence) and category (a classification for the mystery word: computers, animals, movies, etc.), both of them being character strings. Also, a few rows should be inserted. The interactive SQL monitor may be used for both purposes. Executing the command

```
/usr/local/Hughes/bin/msql hangman
```

enters the interactive monitor with the “hangman” database. The MiniSQL prompt should appear. SQL queries can now be issued, followed by “\g”(GO) to indicate that the query should be sent to the database server. Here are the SQL commands for the Hangman application:

```
create table mystery (word char(40), category char(15))\g
insert into mystery values ('elephant', 'animals')\g
insert into mystery values ('rhinoceros', 'animals')\g
insert into mystery values ('gone with the wind', 'movies')\g
```

Accessing the MiniSQL from a Java Program

The application's middle tier uses Blackdown's Linux Port Java Development Kit 1.2.2, release candidate 4, and CIE's mSQL-JDBC driver for JDBC 2.0. The Java tutorial is one of many excellent places to learn how to access databases from within a Java program; that's why only the specific issues on accessing MiniSQL will be dealt with here.

Before attempting to access the MiniSQL server from a Java application, the corresponding JDBC driver must be installed. The driver may be freely downloaded from The Center for Imaginary Environments web site (see Resources). The distribution file comes with many things, but the most important part is the JAR file that contains the driver itself (currently, the file is msql-jdbc-2.0b5.jar). The easiest way to install the driver is to copy the JAR file to the /usr/local/jdk1.2.2/jre/lib/ext directory (root privileges are required to copy files to this directory).

In order to load the driver from the Java program, the following statement should be executed:

```
Class.forName("com.imaginary.sql.msql.MsqlDriver");
```

The connection to the database server is established when executing this statement (ignore line wrap):

```
Connection con = DriverManager.getConnection
 ('jdbc:msql://localhost:1114/hangman');
```

Inside the JDBC URL, the URL of a remote system should replace “localhost” if the MiniSQL server is not running in the same machine. 1114 is the default port number to which the MiniSQL server is listening. The msql.conf file can be modified in order to specify another port number.

Conclusions

The three-tier architecture is a versatile and modular infrastructure intended to improve usability, flexibility, interoperability and scalability. Linux, Java and MiniSQL result in an interesting combination for learning how to build three-

tier architecture systems. Nevertheless, more convenient implementations than the one presented here may be produced using component technology in the middle tier, such as CORBA (Common Object Request Broker Architecture), EJB (Enterprise Java Beans) and DCOM (Distributed Component Object Model). The interested reader should review these topics to get a better understanding of the current three-tier architecture capabilities.

Resources

email: aortiz@campus.cem.itesm.mx

Ariel Ortiz Ramirez (aortiz@campus.cem.itesm.mx) is a faculty member in the Computer Science Department of the Monterey Institute of Technology and Higher Education, Campus Estado de Mexico. Although he has taught several different programming languages for almost a decade, he personally has much more fun when programming in Scheme and Java (in that order). He can be reached at aortiz@campus.cem.itesm.mx.

Archive Index Issue Table of Contents

Advanced search

cgimodel: CGI Programming Made Easy with Python

Chenna Ramu

Christina Gemuend

Issue #75, July 2000

Always look on the bright side of life and at a method for debugging CGI programs on the command line.

The Common Gateway Interface (CGI) is a way in which you can let others from all over the world execute a program that resides on your computer. CGI is dynamic, since it runs in real time. You can decorate the CGI output with HTML (Hyper Text Markup Language). Most of the time, CGI is used as a front end for existing applications. CGI can be easy or complex, depending on the complexity of your project. Most CGI developers know the frustration which comes with debugging the CGI programs.

We present a very simple and robust way of doing CGI programming with Python. Debugging your CGI is easy, since you can do it on the command line, and integrating existing applications to work with CGI is just one step.

For our work, we chose Python, an object-oriented scripting language with a clear syntax. It is very easy to use, widely available and is free software.

Our intended audience is both experienced and novice CGI programmers. We will use the words “function” and “method” interchangeably. Note that CGI can be written in any computer language.

The GET and POST Methods

There are two ways of invoking CGI programs: through a URL with all data included, or by submitting HTML forms.

The two methods defined in HTTP to send your data to the CGI are **GET** and **POST**. When the method is GET, the CGI program gets the input from the

QUERY_STRING environment variable. When the method is POST, the CGI program gets the input from standard input (STDIN). In both cases, one has to parse the input to obtain the input argument name,value pairs.

CGI may or may not be complicated, but when you have a larger application with many features, you might have problems in testing, debugging, etc. This is true with all software projects. Debugging becomes problematic with CGIs. For example, when the method is GET, you have to set up environment variables **QUERY_STRING** and **REQUEST_METHOD**. When the method is POST, you must set up **REQUEST_METHOD** and **CONTENT_LENGTH** (number of bytes) to read from standard input (STDIN). Moreover, when your program crashes, it is not visible to your browser—you do not know what happened. The only message you get in this situation is the error report made by the web server.

You can use either of these methods (GET/POST) depending on your need. If you will be sending more data to CGI, use the POST method. When you have less data to be sent to CGI, use GET to put all the data inside the URL. For example, on one line, type:

```
<A HREF="/cgi/cgimodel.py?fun=DisplayFile&fileName=
cgimodel.pycgimodel">cgimodel</A>
```

With HTML FORMS (for POST method), the same would be

```
<FORM METHOD="post" ACTION="/cgi-bin/cgimodel.py">
<INPUT TYPE=hidden name=fun value=DisplayFile>
<INPUT TYPE=hidden name=fileName value=cgimodel.py>
<INPUT TYPE=SUBMIT VALUE="cgimodel">
</FORM>
```

We all know the difficulties of and have adopted different styles for debugging CGI programs. Our intention is to build CGI that does not work in the traditional way, but like other programs which work on the command line. This means you can test your CGI the way you test any other program on the command line. When it works on the command line, it is guaranteed to exhibit the same behavior on the Web.

The cgimodel Module

Let us see how we can make life easier with **cgimodel**, which lets you integrate your existing application in an elegant way without much hassle. Basically it consists of two modules: **cgimodel.py** (see Listing 1) and **cgidisp.py** (see Listing 2).

Listing 1. cgimodel.py

Listing 2. cgidisp.py

cgimodel.py is a wrapper to Python's CGI module. It also encapsulates reading from the command line, so there is no real difference in invoking from HTML FORMS or a URL or the command line.

The **CollectArgs** function in the cgimodel.py module takes care of collecting arguments including name,value parameters from CGI or the command line. On the UNIX command line, you can supply the name,value parameters like this:

```
-name1 value1 -name2 value2
```

or like this:

```
name1 value1 name2 value2
```

The same is true for both URL and FORMS.

You do not have to modify anything in cgimodel.py. You just have to use it. The main section of cgimodel contains the following lines:

```
d = Dispatcher()
parDict = CollectArgs(parDict)
print mime_html
fun=parDict['fun']
if not fun:
   print "usage: cgimodel -fun functionName"
   d.ShowAvailableFunc()
   TraceIt(parDict)
else:
   try:
      d.dispatch(fun,parDict)
   except:
      TraceIt(parDict)
```

cgimodel.py tries to call the function you have given as an argument to the parameter **-fun**.

When there is no such function available, it tells you the names of functions that can be called. If there is an exception (because of a syntax error, etc.) in the program, the exception will be traced back and reported. You can use this feature to e-mail the exception to yourself and make your CGI program more stable.

The cgidisp Module

The other module, cgidisp.py, is the one in which you have to modify or insert an instance to the class **Dispatcher** for your application using one argument, namely **parDict**. For example, under class **Dispatcher**, if you define a method like

```
def cmd_myHello(self,parDict):
print "<H1>Hello</H1>"
```

then this function is immediately available to the outside world. You can call it on the command line this way:

```
cgimodel.py -fun myHello
```

with URL (GET method)

```
cgimodel.py?-fun=myHello
```

and with HTML forms as

```
<FORM METHOD="post" ACTION="/cgi-bin/cgimodel.py>
<INPUT TYPE=hidden name=fun value=myHello>
<INPUT TYPE=SUBMIT VALUE="Say Hello">
</FORM>
```

It's that easy!

The **dispatch** method under the class **Dispatcher** is called from cgimodel.py with one argument. This argument is the name of the function to be executed. Here is the interesting part. After prefixing the function name with the “cmd_” string, the dispatch method checks to see if such a function is available with **hassattr**. The Dispatcher maps the command to the function and executes it. This way, you do not have to use a lookup table to keep track of available functions. The additional overhead of adding a new command to the new function is not there; you just have to write the function and call it through the command line. The functionality is already there. This kind of pattern is possible with Python, since it is a highly dynamic language.

Please note that when calling the method, we are not using the prefix cmd_ of the method. This is explained later.

The main section of the **Dispatcher** class contains the following:

```
class Dispatcher:
   def __init__(self):
      self.debug = None
   def dispatch(self, command,args=None):
      mname = 'cmd_' + command
      if hasattr(self, mname):
      method = getattr(self, mname)
         if not args:
            return method()
         else:
            return method(args)
      else:
         print "<PRE>" self.error(command)<\n>
         self.ShowAvailableFunc()
         print "</PRE>"
   def cmd_Hello(self,parDict):
      print " Hello World !"
   def cmd_ShowDict(self):
      print "<PRE></H1>Debug Info:</H1><HR>"
      for k,v in parDict.items():
            print "%-30s :  %s " %(k,v)
      print "</PRE>"
   def error(self,s):
      print " #<B>Error<B>: <BB>Function ( %s ) not available\n " %s
      return
```

All your parameters are available in the parDict dictionary whether they are input from URL, FORM or command line—there is no difference. You can check for their existence in this way:

```
if parDict['param']:
   print " yes ", parDict['param']
else:
   print " No "
```

The **None** object is returned when there is no parameter, i.e., when you try to access an unspecified parameter.

The instances inside the class **Dispatcher** are of two types: those that are prefixed by the “cmd_” string are qualified for calling from outside; internal instances are not visible outside. For example, the **error** instance cannot be called from CGI, but the instances **cmd_Hello** and **cmd_ShowDict** can be called. This convention is made to differentiate between the instances that are for internal (used inside the class **Dispatcher**) and external (by cgimodel/cgidisp) use.

So, add a “cmd_” prefix to the instances you want to use with CGI. For example, cmd_TopPage can be called with

```
cgimodel.py -fun TopPage
```

on the command line and

```
cgimodel.py?-fun=TopPage
```

will be the corresponding URL. The **-fun** is mandatory. This way, you can indicate which function you want to call. Obviously, you can have as many functions as you want, and they are CGI-ready. This is the exact requirement of larger CGI projects.

A couple of functions come with the module for free. The function **DisplayFile** displays colorized Python source code on the Web. This one relies on the module **py2html.py**, available with the standard Python distribution.

```
cgimodel.py -fun DisplayFile -fileName cgimodel.py
```

URL equivalent:

```
cgimodel.py?-fun=DisplayFile&fileName=cgimodel.py
```

Note the name=value and the **&** to separate the name,value pairs—the traditional method of specification for CGI.

The method **cmd_ShowDict** shows all dictionary items in the parDict dictionary and is useful for checking whether you have supplied the correct parameters.

Adding Existing Modules to CGI

Assume you already have this module:

```
#!/usr/bin/env python
#  testmethod.py
def  Method1(name1,name2,name3):
        print name1,name2,name3
if __name__ == '__main__':
     Method1('one','two','three')
```

Edit the cgidisp.py module, inserting the following method under the class **Dispatcher**:

```
def cmd_TestMeth(self,parDict):
   import testmethod
   name1 = parDict['name1']
   name2 = parDict['name2']
   name3 = parDict['name3']
   testmethod.Method1(name1,name2,name3)
```

Now it is ready! You can call this on the command line by typing on one line:

```
cgimodel.py -fun TestMeth -name1 one -name2 two -name3 three
```

or by URL (all on one line):

```
cgimodel.py?-fun=TestMeth&name1=one&name2=two&name3=three
```

or by FORMS:

```
<FORM METHOD="post" ACTION="/cgi-bin/cgimodel.py">
<INPUT TYPE=hidden name=fun value=TestMeth>
<INPUT TYPE=hidden name=name1 value=one>
<INPUT TYPE=hidden name=name2 value=two>
<INPUT TYPE=hidden name=name3 value=three>
<INPUT TYPE=SUBMIT VALUE="Run">
</FORM>
```

It would be much better if you could separate HTML text from CGI modules, so that CGI looks thinner and more readable. You can use the template modules (see Resources) to do this. The template module keeps the text away from the CGI and has a page-paragraph structure. Each CGI call can be associated with a page, and each paragraph can be used to set up the view of your HTML page.

cgimodel can host any number of applications. The redundancy of writing a CGI front end is no longer necessary. Since many applications can be run by a single cgimodel, logging information particular to each application can be done for later analysis to improve server performance, stability of each application, better service, etc. Currently, this can be done with the log information generated by the web server.

With cgimodel.py, cgidisp.py and possibly the **template.py modules**, you should find writing and testing CGI programs easier.

Resources

Chenna Ramu (ramu.chenna@embl-heidelberg.de) holds a postgraduate degree in mathematics. He currently works for European Molecular Biological Laboratory in Heidelberg, Germany, in the area of biocomputing. Interests are theoretical study about DNA/protein sequences, database development, parsing, compilers, system administration and web technology. He came across Python recently (thanks to Gert Vriend) and found it quite nice for programming.

Christine Gemuend has a degree in computer science. She is interested in parallel computers and database systems, and is working in the area of informatics.

Archive Index Issue Table of Contents

Advanced search

Mapping Lightning with Linux

Timothy Hamlin

Issue #75, July 2000

NM Tech studies lightning to determine basic charge structures and learn more about storm morphology.

The New Mexico Institute of Mining and Technology (NM Tech) has developed a deployable system for mapping the lightning discharge activity of thunderstorms in three spatial dimensions and time. The array, which utilizes Global Positioning System (GPS) technology for accurate timing and is patterned after the Lightning Detection and Ranging system developed and utilized by NASA's Kennedy Space Center, is known as the Lightning Mapping Array (LMA) and consists of 10 to 15 stations deployed about a county-wide area of approximately 30-50km in diameter.

Each station detects and accurately times the arrival of impulsive VHF (very high frequency) events in a 6MHz passband centered at 63MHz. The time and magnitude of the impulsive radiation is recorded up to once every 100 µs, making it possible to store up to a maximum of 10,000 triggers per second. Triggers from each station are combined with data from the other stations at a central site, where differential time-of-arrival techniques are used to generate solutions for the locations of the source events, in both space and time. A typical lightning discharge may consist of several hundred to a few thousand located sources. The observations made by the system are found to reflect the basic charge structure of electrified storms, as well as provide invaluable information on the overall storm morphology.

Figure 1 demonstrates observations of a horizontally extensive lightning discharge from a storm over central Oklahoma on the evening of June 10, 1998. The discharge occurred over the southern edge of the measurement network and had an overall extent of 75km. The flash began as an intracloud discharge between the main negative and upper positive charge layers of the storm. Subsequent breakdown in the negative charge level continued the discharge to the north, where it produced a negative-polarity stroke to ground. This

discharge, which lasted approximately 2.5 seconds, produced more than 2400 located radiation events and demonstrates the LMA's ability to determine discharge structure at the individual flash level. The different panels show plan view, east-west and north-south vertical projections and height-time plots of the lightning activity. The color scale in this example represents the progression of time, blue being early in the flash and red being late (linear scale). The small boxes on the plan view detail the locations of the measurement stations.

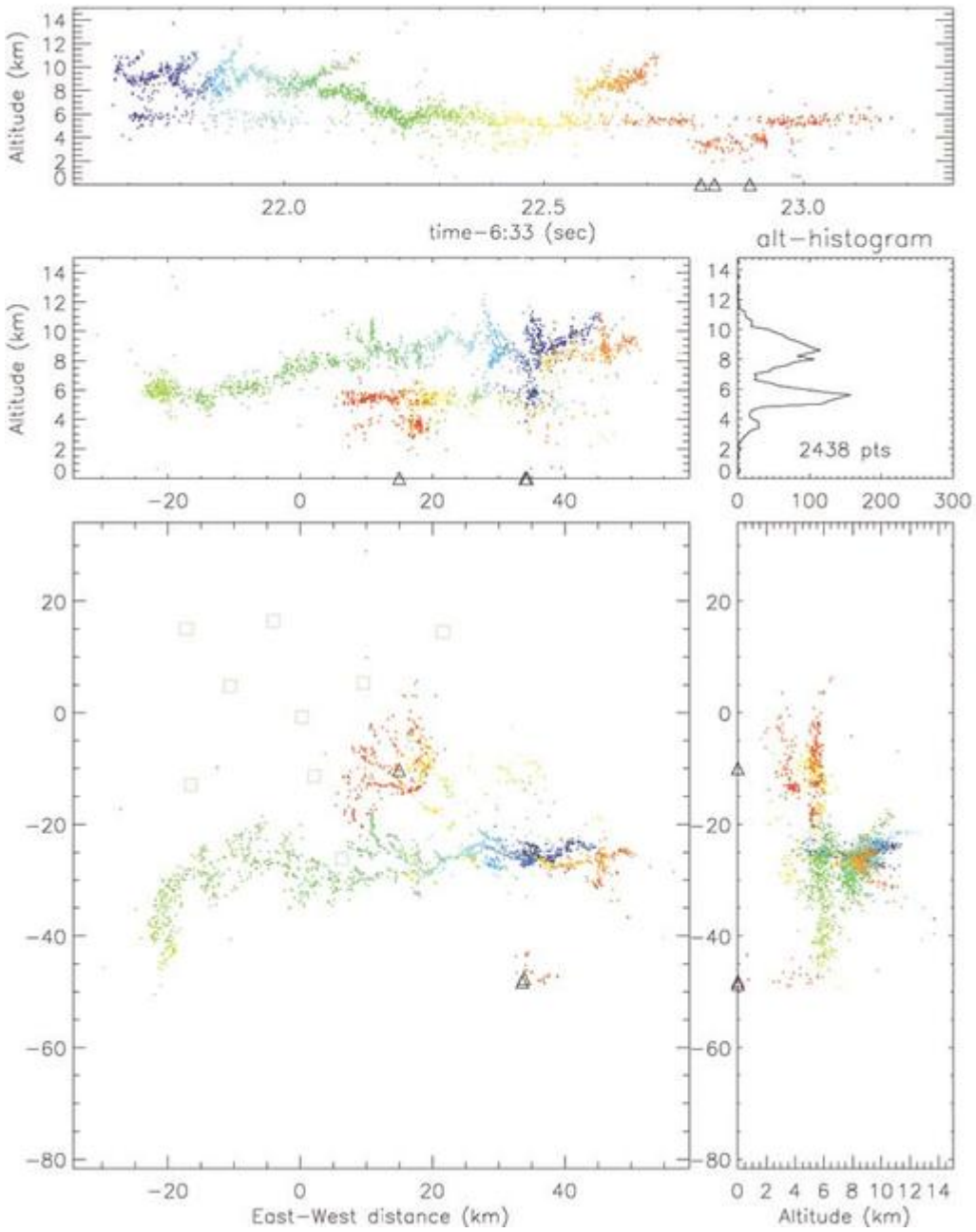


Figure 1. Lightning Discharge in Oklahoma

Figure 2 demonstrates the LMA's ability to view storms on much larger scales. This example is an integration of five minutes of data, where the color scale now represents event density, red being the highest activity and purple the lowest (logarithmic scale). Data of this type helps to determine the location of

the main conective cells in storm systems, as well as determine the overall size and structure of the storm.

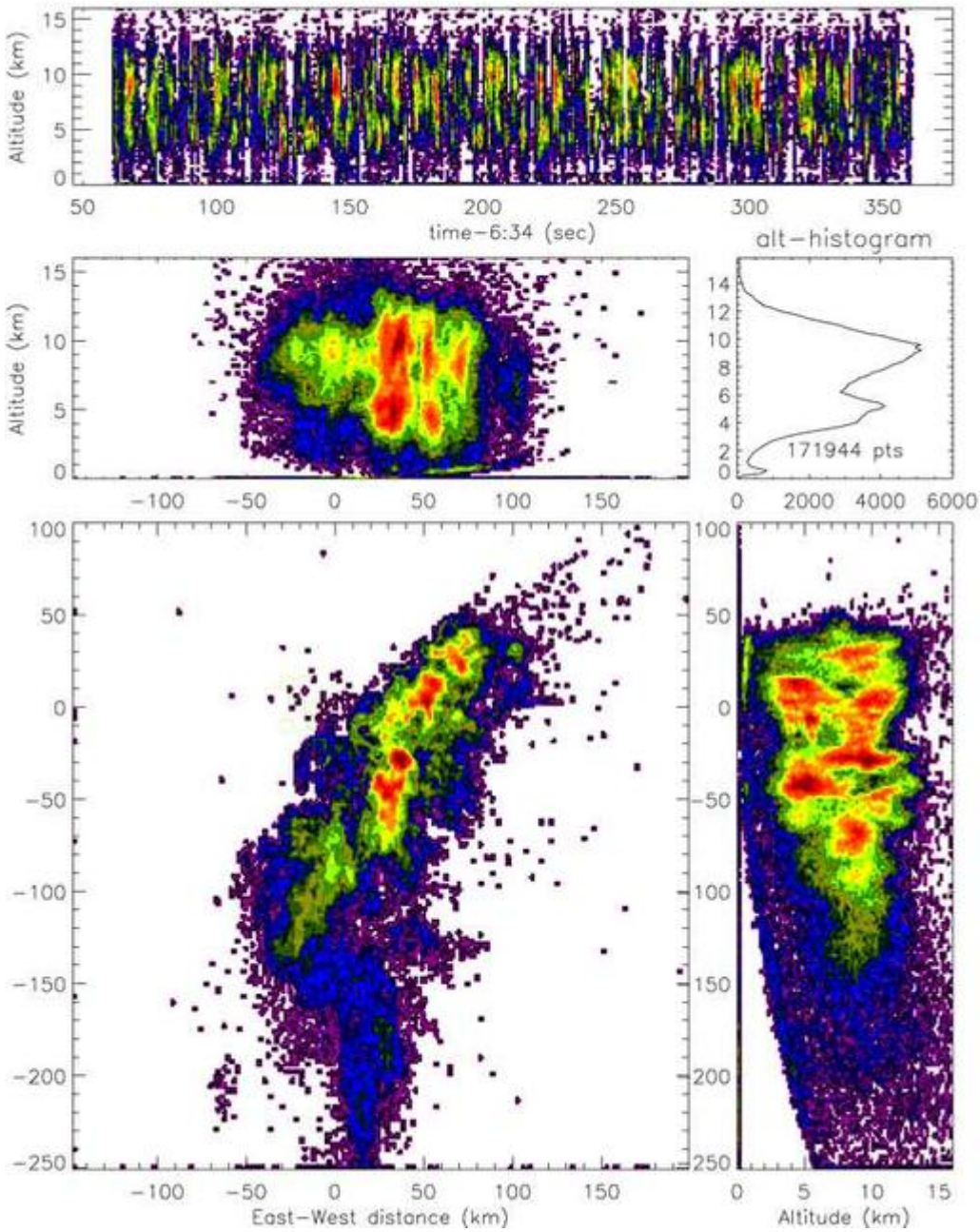


Figure 2. LMA Integration of Five Minutes of Data

What's Linux Got to Do with It?

At the very core of operation for this system is Linux. To best demonstrate the flow of data and Linux's involvement with it, we trace the stream from its source. An impulsive VHF event occurs, and the radiation from it arrives at a given remote station. Operating in each station is an inexpensive PC running Linux. Each PC has a custom-built digitizer card (built and designed here at NM Tech) which interfaces with the PC over the ISA bus. The arrival time and power of each event is recorded and stored into a 16-bit, 32KB-deep FIFO. The digitizer

board sets a flag (or generates an interrupt) noting when the FIFO becomes half-full.

On the Linux side, there is a device driver (written in C) running, which polls for the half-full flag or can be set to utilize interrupts (they happen only a few times a second, so polling works quite well). When a half-full flag is detected, the driver initiates a block transfer from the FIFO to the system, where the data is then flushed to disk.

The digitizer board itself, which utilizes a microcontroller in conjunction with a programmable logic device (PLD) to maintain phase-lock with the GPS receiver (good to about 50-100 nanoseconds), must be configured using a passive-serial mode. This is accomplished by interfacing through the parallel port on the PC and clocking in the configuration data (one bit at a time) needed to program the PLD interface. A few calls to **ioperm** make this an easy task, and the availability of excellent Linux documentation helped greatly along the way.

The collection process goes on all the time, so data starts to build up on the remote stations. The data files are stored as fixed-length files, 10MB each. Once a file reaches 10MB, Perl scripting takes charge. The scripts perform a variety of actions. First, they take a completed file and write it to the local DAT tape drive. Currently the stations use DDS-II drives, so each tape can hold about 4GB of uncompressed data, which amounts to about a week's worth of data (depending on storm activity). After a successful write to tape, the file is moved to a backup location on the local disk as a redundancy method. The scripting monitors the tape writes, logs them and monitors hard-drive space. If drive space becomes limited, the oldest files successfully written to tape are removed. Data tapes are collected on a routine basis via the famed “sneaker net” and brought back to a central location for postprocessing.

It would be nearly impossible (although we have done it) to do this all blind. We take advantage of the ease of networking in Linux to achieve real-time control over the stations. Each remote station connects back to a central site via wireless radio modems operating at 115,200bps. The remote stations establish a PPP link between themselves and the main site; the PPP links are maintained quite easily with use of more Perl scripts. Having all of the stations networked to a central site, one which has Internet access, is a real benefit. This allows for control over the stations from virtually anywhere. All one has to do is get to the central site, and access to the remote stations is only an **slogin** away. Real-time control is invaluable; there are many things which need to be monitored and adjusted at each station, such as the triggering threshold of the digitizer, monitoring signal strength, checking battery backup status and controlling the GPS receiver.

Once the data have been collected at the central processing location, the real work begins. The data must be extracted from tape and combined for each station with corresponding time intervals. A workhorse machine is needed. We currently use a dual-processor Linux box running SMP. After combining the data, various forms of filtering must be done before the solutions can be obtained. A variety of languages and services help with the task: filters written in C, Perl and shell scripts for keeping track of vitals, more C to invert the data, IDL to display the data, HTTP and FTP servers to collaborate with others and so on.

In the works is a transition to full real-time processing, which is well underway. We hope to eliminate the sneaker net and bring all data back real time, process it and archive it all in one place. This will reduce the number of man hours considerably, as well as make the system a more effective tool in collaborative field campaigns.

Linux has proven to be an important part of every aspect of this project. The ease of development on this platform is unparalleled, as is its flexibility of application. The prodigious documentation, as well as the massive proliferation of freely distributed software, has made it our OS of choice. We could not do what we do without it! It should also be mentioned that the PCs in the remote stations are up constantly—the last /proc/uptime reports over 300 days since the last reboot. Show me a Windows box that can run for nearly a year, network constantly, and archive over 100GB of data without so much as a hiccup!

Acknowledgements

email: thamlin@nmt.edu

Timothy Hamlin (thamlin@nmt.edu) has been a graduate student in physics at NM Tech for the past three years, and hopes to complete his Ph.D. sometime before the next millennium. He has been an avid UNIX user for over 10 years and a Linux convert/administrator/programmer/what-have-you since 1996.

Archive Index Issue Table of Contents

Advanced search

Using Linux in Embedded and Real-Time Systems

Rick Lehrbaum

Issue #75, July 2000

When you need an embedded operating system, Linux is a good place to start. Here's why.

Intelligent dedicated systems and appliances used in interface, monitoring, communications and control applications increasingly demand the services of a sophisticated, state-of-the-art operating system. Many such systems require advanced capabilities such as high resolution and user-friendly graphical user interfaces (GUIs), TCP/IP connectivity, substitution of reliable (and low-power) flash memory solid-state disk for conventional disk drives, support for 32-bit ultra-high-speed CPUs, the use of large memory arrays, and seemingly infinite capacity storage devices including CD-ROMs and hard disks. This is not the stuff of yesteryear's “stand-alone” code, “roll-your-own” kernels or “plain old DOS”. No, those days are gone—forever!

Then, too, consider the rapidly accelerating pace of hardware and chipset innovation, accompanied by extremely rapid obsolescence of the older devices. Combine these two, and you can see why it has become an enormous challenge for commercial RTOS (real-time operating system) vendors to keep up with the constant churning of hardware devices. Supporting the newest devices in a timely manner—even just to stay clear of the unrelenting steamroller of chipset obsolescence—takes a large and constant resource commitment. If it's a struggle for the commercial RTOS vendors to keep up, going it alone by writing stand-alone code or a roll-your-own kernel certainly makes no sense. With the options narrowing, embedded system developers find themselves faced with a dilemma:

- On one hand, today's highly sophisticated and empowered intelligent embedded systems—based on the newest chips and hardware capabilities—demand nothing less than the power, sophistication and currency of support provided by a popular high-end operating system like Windows.

- On the other hand, embedded systems demand extremely high reliability (for non-stop, unattended operation) plus the ability to customize the OS to match an application's unique requirements.

Here's the quandary: general-purpose desktop operating systems (like Windows) aren't well-suited to the unique needs of appliance-like embedded systems. However, commercial RTOS, while designed to satisfy the reliability and configuration flexibility requirements of embedded applications, are increasingly less desirable due to their lack of standardization and their inability to keep pace with the rapid evolution of technology.

The Alternative

Fortunately, an exciting alternative has emerged: open-source Linux. Linux offers powerful and sophisticated system management facilities, a rich cadre of device support, a superb reputation for reliability and robustness and extensive documentation. Best of all (say system developers), Linux is available with source code at no charge.

Unlike Windows, Linux is inherently modular and can be scaled easily into compact configurations—barely larger than DOS—that can even fit on a single floppy. What's more, since Linux source code is freely available, it's possible to customize the OS according to unique embedded system requirements. It's not surprising, then, that Linux has created a new OS development and support paradigm in which thousands of developers continually contribute to a constantly evolving Linux code base. In addition, dozens of Linux-oriented software companies have sprung up, eager to support the needs of developers building a wide range of applications, ranging from factory automation to intelligent appliances.

Which Linux?

Because Linux is openly and freely available in source form, many variations and configurations are available. There are implementations specifically for “thin server” or “firewall” applications, small footprint versions and real-time enhanced versions. There are also Linux ports for non-x86 CPUs, including PowerPC, RISC, 68*xxx* and microcontrollers.

How do you decide which distribution to use? First, realize that all Linux distributions are variations on the same theme—that is, they are collections of the same basic components, including the Linux kernel, command shells (command processors) and many common utilities. The differences tend to center around which of the many hundreds of Linux utilities have been included, what extras are included on the distribution CD and how the installation process is managed. Distributions may include either open-source

or proprietary software such as, for example, StarOffice. Even though Linux is free, purchasing a “commercial” Linux distribution can have many advantages, including development tools, useful utilities and support. The many companies which are now building businesses around distributing and supporting Linux are busy investing in developing tools and services to differentiate their Linux offerings from the rest of the pack. Some of the special capabilities being developed include:

- Installation tools to automate and simplify the process of generating a Linux configuration that is tuned to a specific target's hardware setup. These can save weeks or even months of development effort.
- A variety of Windows-like GUIs that vary in size, appearance, features and capabilities to support a wide range of embedded requirements.
- Support for the specific needs of various embedded and real-time computing platforms and environments (e.g., special CompactPCI system features).

Yet despite this great diversity, all Linux implementations share a common core, including kernel, drivers, several popular GUIs and utilities.

Small-Footprint Linux

For many embedded systems, the main challenge in embedding Linux is to minimize system resource requirements in order to fit within constraints such as RAM, solid-state disk (SSD), processor speed and power consumption. Embedded operation may require booting from (and fitting within) a DiskOnChip or CompactFlash SSD; booting and running without a display and keyboard (“headless” operation); or loading the application from a remote device via an Ethernet LAN connection. There are many sources of ready-made, small-footprint Linux. Included among these are a growing number of application-oriented Linux configurations and distributions that are tuned to specific applications. Some examples are routers, firewalls, Internet/network appliances, network servers, gateways, etc. You may also opt to create your own flavor of embedded Linux, starting from a standard distribution and leaving out modules you don't need. Even so, you should consider jump-starting your efforts by beginning with someone else's working configuration, since the source code of their version will be available for that purpose. Best of all, this sort of building on the efforts of others in the Linux community is not only completely legal—it's encouraged!

Real-Time Linux

Many embedded systems require predictable and bounded responses to real-world events. Such “real-time” systems include factory automation, data acquisition and control systems, audio/video applications and many other

computerized products and devices. The commonly accepted definition of real-time performance is that real-world events must be responded to within a defined, predictable and relatively short time interval. Although Linux is not a real-time operating system (the Linux kernel does not provide the required event prioritization and preemption functions), several add-on options are available that can bring real-time capabilities to Linux-based systems. The most common method is the dual-kernel approach. Using this approach, a general-purpose (non-real-time) OS runs as a task under a real-time kernel. The general-purpose OS provides functions such as disk read/write, LAN/communications, serial/parallel I/O, system initialization, memory management, etc., while the real-time kernel handles real-world event processing. You might think of this as a “have your cake and eat it too” strategy, because it can preserve the benefits of a popular general-purpose OS while adding the capabilities of a real-time OS. In the case of Linux, you can retain full compatibility with standard Linux while adding real-time functions in a non-interfering manner.

Of course, you could also dive in and modify Linux to convert it into a real-time operating system, since its source is openly available. But if you do this, you will be faced with the severe disadvantage of having a real-time Linux that can't keep pace, either features-wise or drivers-wise, with mainstream Linux. In short, your customized Linux won't benefit from the continual Linux evolution that results from the pooled efforts of thousands of developers worldwide.

Who Needs Real-Time?

In any case, how many applications actually require real-time enhancements to Linux? Bear in mind that “real-time” is a relative, not an absolute, expression. As mentioned before, a real-time system must handle real-world tasks within acceptable—and predictable—time windows. Although CPUs run at ever-increasing speeds (approaching 1GHz), the world around them goes on at a constant speed. Therefore, real-time performance is becoming ever easier to achieve. Back when the “traditional” RTOS were first developed, embedded systems depended on 4- and 8-bit CPUs clocked at single-digit megahertz speeds and running out of kilobytes of RAM. Now, with CPUs speeding along at up to 1GHz and with memories measured in the hundreds of megabytes, real-time performance is becoming less of a concern. The greater concern has become speed to market and sophistication of functionality. Where execution efficiency was the watchword of the CPU-bound past, protocols are the key to the Internet-centric future.

Going Soft

There's a term to describe the use of ordinary operating systems in real-world applications with acceptable results: “soft real-time”. In many systems, you can

ensure that the real-world constraints of your application can be achieved without resorting to using a specialized RTOS. However, this tends to be practical only when required response times are in milliseconds—not microseconds. Assuming that's the case, a minimally configured Linux on a reasonably fast processor (486-133 or faster) without special real-time add-ons may well suit your needs. If soft real-time sounds like what you need, you may want to check out a Linux add-on called Linux-SRT (soft real-time). On the other hand, your system may indeed require microsecond-level response times. In that case, you can either dedicate an inexpensive microcontroller or DSP to handling the time-critical events, or you can use one of several available real-time Linux add-ons (e.g., RTLinux or RTAI).

Embedded and Real-Time Linux Solution Providers

Since Linux is free, how can anyone build a profitable business based on offering commercial Linux distributions? It's a lot like bottled water—basically, what you pay for is services: packaging, delivery, quality assurance, etc. A word of caution: don't assume that every Linux-related program you download from the Web or obtain from a Linux CD can be freely reproduced and incorporated into the devices you develop. Some commercial Linux distributions that target embedded and real-time applications include proprietary third-party tools and utilities that require licensing and royalty payments if you incorporate them into multiple systems. In other words, read the fine print. For now, however, licensed Linux system software is the exception rather than the rule. With the market placing such a high value on Linux and its associated software being open source and royalty-free, most Linux software companies serving the embedded and real-time Linux market have opted to build their businesses based on selling tools, offering engineering services and providing technical support.

Given the strong position of Microsoft Windows in the end-user desktop/laptop market, it's unlikely the “average” desktop/laptop PC user will be running Linux any time soon. On the other hand, in embedded and real-time applications where the OS is an underlying and hidden technology supporting appliance-like operation of a non-computer device, several key features of Linux are making it a growing preference among system developers:

- Source is available and free.
- There are no runtime royalties.
- Linux supports a vast array of devices.
- Linux is truly a global standard.
- Linux is sophisticated, efficient, robust, reliable, modular and highly configurable.

Time will tell, but it certainly looks as though Linux has already altered the embedded and real-time operating system landscape in a fundamental and irreversible way. The result? Developers now have greater control over their embedded OS; manufacturers are spared the costs and headaches of software royalties; end users get more value. And the penguins of the South Pole are celebrating.

Resources

email: rick@linuxdevices.com

Rick Lehrbaum (rick@linuxdevices.com) co-founded Ampro Computers, Inc. in 1983. In 1992, Rick formed the PC/104 Consortium and served as its chairman through January 2000. In October 1999, Rick turned his attention to embedded software, founding LinuxDevices.com—"the Embedded Linux Portal".

Archive Index Issue Table of Contents

Advanced search

Troll Tech Announces Embedded GUI Toolkit

Craig Knudsen

Issue #75, July 2000

Troll Tech enters the embedded systems market—here's what's happening.

Troll Tech announced in March that it is developing Qt/Embedded, a new GUI toolkit for embedded Linux systems. The toolkit aims to provide embedded systems developers with the same power and cross-platform portability as the desktop versions of Qt, Qt/Windows and Qt/X11.

Linux users will recognize Qt as the GUI toolkit that powers the popular K Desktop Environment (KDE), now a standard component of most Linux distributions including Caldera OpenLinux 2.4, Linux Mandrake 7.0, Corel Linux 1.0, Red Hat Linux 6.2 and Slackware 7.0.

Unlike Troll Tech's Qt/X11 product for Linux, Qt/Embedded does not require the X Window System. Instead, Qt/Embedded applications access the Linux frame buffer directly. Removing X reduces the memory requirements of the toolkit drastically, leaving more memory available for applications.

While Troll Tech is not aiming to replace X, it may do so in some cases. Qt/Embedded includes its own windowing system, allowing multiple applications to run with overlapping windows. Qt/Embedded will not, however, provide remote display capabilities like X. Qt/Embedded also provides its own support for TrueType fonts, a function normally provided by X.

Because Qt/Embedded is source-code compatible with Qt 2.1, any application that works with Qt 2.1 can be built with Qt/Embedded. In fact, a version of KDE has already been tested with Qt/Embedded. It's unlikely that all the current KDE and Qt applications will begin showing up on embedded devices. However, certain applications, such as the Qt version of the Mozilla browser, are obvious candidates for an embedded device.

KDE's success has been an asset to Troll Tech. It certainly can't hurt the future of Qt/Embedded to have a growing number of open-source developers around the world familiar with Qt development.

Qt Screenshot

The Qt/Embedded API will be identical to the existing API used in Qt 2.1, allowing applications to be portable from UNIX and Windows desktops to embedded systems. Developers can use either Qt or X11 to develop their application, allowing them to leverage a huge base of desktop tools. The application can be deployed on the target system simply by rebuilding with Qt/Embedded.

Qt/X11's original license caused many concerns in Linux and Open Source communities due to its restrictions on commercial usage. Qt did not meet the official open-source definition, preventing KDE from being included in some Linux distributions. In response, Troll Tech developed the Q Public License (QPL), a new, less-restrictive license that qualifies as open source. The new license has been a significant factor in the success of both Qt and KDE.

Qt/Embedded application programmers will need to purchase a Development Kit license under terms similar to the Qt Professional Edition. There will also be a run-time license for each device in which Qt/Embedded is installed.

If you're an open-source developer, the licensing terms might not sound very inviting to you. Learning from past licensing problems, Troll Tech hasn't forgotten the open-source developer. Although details have not been finalized,

Troll Tech plans to make available a free edition of the Development Kit that will make it possible to write open-source applications for use on devices with Qt/Embedded already installed.

Up until now, talking about Qt in the Linux community meant Qt/X11. With Qt/Embedded on the way, Linux developers have more options. Linux has continued to garner interest in the embedded systems market, and an advanced GUI toolkit such as Qt will make Linux that much more attractive.

The general release of Qt/Embedded is targeted for the third quarter of this year, with OEMs receiving pre-releases earlier. At the time of publication, Troll Tech had not yet announced pricing details for the run-time licenses or the Developer Kit.

Resources

email: cknudsen@radix.net

Craig Knudsen (cknudsen@radix.net) lives in Fairfax, VA and telecommutes full-time as a web engineer for ePresence, Inc. of Red Bank, NJ. 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

The Montréal 2000 Linux Expo April 10-12

Marcel Gagné

Issue #75, July 2000

LJ's French chef visits Montréal April 10-12 for more than the food.

Quick—those people who have attended a science-fiction convention, put up your hand. Go ahead, admit it—this is a friendly crowd. SF convention goers have a term that describes conventions like the Montréal Linux Expo: *Relaxcon.* This term refers to a quiet convention, one where you can basically just relax and chat with people. While this is great for fans wanting to get close to their idols, businesses and exhibitors spending several thousand dollars for a booth may have other ideas—even in the world of free software. Don't get me wrong. On Tuesday, day one of the show, the place was hopping and you could feel the excitement at the Palais des Congrés. All in all, it was a fairly successful opening day, especially considering the weather. Wednesday, however, was a different matter. The general consensus from the vendors I spoke to on Wednesday was that day two might just as well have not happened, considering the number of people in attendance. For those of you who missed the show, let me try to paint you a picture.

Ah yes, the weather. We arrived in Montréal during what was probably the worst snowstorm of the winter. We left Mississauga with the sun shining and temperatures of 10 degrees Celsius. Around Brockville (about two-thirds of the way there), things got ugly as the blizzard set in. Our six-hour trip turned into the ten-hour drive from hell. Attendance may have been light due to the great dump of snow, but I certainly don't think the weather can be blamed for everything. The organizers expected 6000. One Montréal newspaper estimated slightly over 4500. Another paper, this one in Ottawa, cynically proclaimed, “barely 300 there for the key speeches”. I'll step out on a limb, double that number for the key speeches, and guesstimate 2500 to 3000 attendees.

Each of the keynotes was held on the first day of the Expo, Tuesday. The notables were Corel's Michael Cowpland, Red Hat's Bob Young, MandrakeSoft's Jacques Le Marois, SuSE's Dirk Hohndel, Oracle Corp., Jon “maddog” Hall and

Eric S. Raymond. Linuxconf author Jacques Gélinas introduced the last two speakers, while Jean-Claude Guédon, professor of comparative literature at the University of Montréal and author of the book *La Planète Cyber* (*Cyber Planet*), moderated and introduced the keynotes.

Linux aficionados have already heard much of what was said in the keynotes, but there were some comments worth noting. Professor Guédon pointed out that his part in this Expo was partly that of an interested observer, a chronicler of a new direction in the history of the high-tech world—the Linux revolution. He suggested that "a Linux company is a bit like a scientist, because a scientist is a kind of intellectual entrepreneur." This fit well with Eric Raymond's speech on the value of peer review in any intellectual enterprise. He explained to the audience that pseudo-sciences like alchemy took the next evolutionary step toward true science only when the work of the scientist (the source) was finally opened to public scrutiny and peer review (the open model of which Linux and GNU are the poster children). Based on the centuries of success of this approach, Mr. Raymond said he did not know what the future held, but he could tell us with near absolute certainty that "You ain't seen nothin' yet!"

Red Hat's Bob Young made an interesting analogy comparing closed-source software to the feudal system, where serfs paid for a plot of land for the privilege of working on it. He pointed out that with certain closed-software distributions, you might even risk going to jail for improving it, which would require that you somehow gained access to this highly guarded source or intellectual property. Whatever your motives, you still lose.

SuSE's CTO, Dirk Hohndel, likened the competition in the Linux marketplace to a friendly battle of wits, with companies smiling at each other while they try to outsmart each other. The great thing about Linux, he said, could be summed up in these points:

- It is a dream come true.
- It actually works.
- It's fun to be part of.

Jon "maddog" Hall, one of the legends of Linux and the Open Source movement, gave a talk entitled "Life in the Elevator". Unfortunately, it was delayed by hardware problems, and Mr. Hall called for recess while he hunted down his own notebook. The premise of his speech is that you enter the elevator at the bottom floor of a high-rise. Inside this elevator is a person with whom you have one minute to chat. In that one minute, they can ask any question about Linux (for instance, "What is it with the penguin, anyway?"). Mr. Hall had 15 questions in all. In that minute-long ride, you have to be able to

answer any of 15 questions he presented. After each answer, Mr. Hall would declare, “And that's another minute.”

In some ways, that forced recess was a lucky break for me. In those few minutes, I caught up with Dirk Hohndel before he moved on to his next engagement. I must tell you I was already impressed with Mr. Hohndel's presentation, professionalism and support for his colleagues (and competitors) during his keynote address. In stopping for an informal chat, he proved genuine. After a brief introduction, I asked him what he has been up to and where SuSE was going.

“We are very happy with SuSE's position in the marketplace,” he told me. “With the release of our SuSE 6.4, we've achieved a 95% first-time install success.” He explained that this was partly due to SuSE's new **YaST2** graphical installation tool. “Automatic hardware detection has become extremely simple.” For the 5% or so of installs that might miss something, SuSE still offers the old text-based install with more configuration options. Release 6.4 also includes a preview of XFree86 4.0, though it is not the default configuration. Mr. Hohndel said that, as one of the developers on the project, he feels this next step in X technology is important and should get as much exposure as possible.

When I asked about the future at SuSE, Mr. Hohndel said, “We are concentrating on two areas primarily: the desktop and enterprise computing.” SuSE feels that the desktop is extremely important to Linux's success and as such was an early supporter of the KDE project and continues to be supportive of KDE2. In fact, four of the developers on the KDE project now work at SuSE. Still, before KDE, you need X. Here's a sneak peek at the future of the desktop from SuSE. Their new graphical configuration tool for X, named **SaX2**, is the next evolutionary step in simple, dynamic X setup and configuration. With SaX2, SuSE hopes to make the dreaded task of X configuration and tweaking a thing of the past.

On the enterprise side, SuSE is increasingly providing services to Fortune 100 companies with data centers and support. (SuSE has their own support organization.) They are developing technology partnerships in a number of areas. Mr. Hohndel mentioned one such partnership with SGI, where they are building a framework dedicated to bringing high-availability clustering to the Linux world. “This is the kind of support that large companies want to see developed in Linux,” he said.

The main show floor had quite a variety of exhibitors, from Linux organizations to Linux companies and other things in between (and, of course, *Linux Journal*). There was a great concentration of knowledge (special thanks to the Red Hat folks for answering a niggling question regarding parallel-port weirdness in a

6.1 install), and one or two companies that didn't seem to understand that they were, in fact, selling Linux products. “Oh, no, this isn't Linux at all. It is a Linux server that offers .... ” Okay, whatever.

All in all, Linux Expo Montréal was great fun and I thoroughly enjoyed myself, but I think it could have been much better. Wise and infallible in the ways of marketing I am not, but I do have some ideas as to what went wrong. It wasn't such a large Expo that visitors couldn't wander through most displays in a day. Combine this with the fact that the only other free items on the program, the keynotes, were all bundled into a three-hour session on the first day, and I think we begin to see the problem. Simply put, if you were there on the first day, why bother coming back unless you had dropped the big bucks to attend?

Another problem can be best summed up anecdotally.

One of the things I did on my visit was deliver a computer to my parents, a computer running Linux with a K desktop. My parents are complete computer newbies, but I am convinced that Linux will be as easy for them to use as “that other OS”. Not only that, it will be more reliable (read *fewer support calls* for me). Anyway, after the show, I stopped at a local bookstore (Camelot.ca, which happened to have a booth at the Expo under Addison Wesley's banner). While waiting at the checkout with my father's Linux reference book in hand, a university student stood in front of me paying for a Shockwave book. He looked over at me, saw the book, and asked me about Linux (note to Jon Hall—include checkouts at malls in next talk). I explained that I had been at Linux Expo.

“I didn't know there was a Linux Expo in town. When and where is it?” he asked.

“Actually, today is the last day, at the Palais des Congrés,” I said. It was already two o'clock.

“That's too bad. I would have liked to have gone. How did you find out about it?”

This guy is working with computers in a university. It occurs to me that the university computer-science crowd would have been a perfect target audience. Microsoft and IBM know this well and do their best to make sure university computer-science graduates know about their products. Why couldn't the Sky Events people figure that one out? Where did all the advertising go? I was invited, as were others who are already among the converted, but what about the people who are still discovering Linux, who may want to learn more? Who else did they miss in this way?

Sky Events is already advertising next year's Montréal Linux Expo. Heading down the escalators as I left the show, I noticed banners proudly proclaiming

Linux Expo 2001—more or less same penguin-time, same penguin-channel. I hope Sky Events will take the lessons from this year's conference and turn next year's event into an even more exciting presentation. Linux Expo in Montréal was great fun, and there is no better place in North America to eat than Montréal (trust me, gentle readers, the food is amazing). It would have been nice, however, to see more faces there.

Marcel Gagné (mggagne@salmar.com) lives in Mississauga, Ontario. In real life, he is president of Salmar Consulting Inc., a systems integration and network consulting firm. He is also a pilot, writes science fiction and fantasy and edits TransVersions, a science fiction, fantasy and horror magazine. He loves Linux and all flavors of UNIX and will even admit it in public.

Archive Index Issue Table of Contents

Advanced search

Medusa DS9 Security System

Robert Dobozy

Issue #75, July 2000

It's not a panacea, it's not perfect, but it is a very interesting piece of software.

- Authors: Marek Zelem, Milan Pikula and Martin Ockajak
- E-mail (mailing list): medusa@medusa.fornax.sk
- URL: http://medusa.fornax.sk/
- Price: Free, released under GNU GPL
- Reviewer: Robert Dobozy

Imagine a perfectly secured Linux box. With the newest patches, with Tripwire running every night, with log analyzer connected to another computer using a serial line which will call you by phone when something suspicious occurs—sounds good? Now imagine a new bug in Sendmail or **ftpd**, hundreds of relatively unknown users using many better- or worse-written setuid programs. Hmmm. Can you sleep well? You can say, “I have monitoring scripts, so I'll be informed by phone when somebody gains root access.” But you could just cry when some stupid vandal immediately executes **rm -fr /**. It's too late, especially if you are away and cannot sit in front of your computer and have a look at what happened. Or you can say, “I'm making a regular backup of my system, so I'll restore it.”

Making backups is a laudable activity, but will you restore every week, when every week somebody finds a new bug before you can fix it? Well, it seems you have to disconnect your computer from the Internet, shut it down and pray your data will be safe. No, don't panic! I have something for you. Its name is

Medusa DS9. It's not a panacea, it's not perfect, but it is a very interesting piece of software.

What is Medusa DS9 and How Does it Work?

In the Slovak language, “medusa” means “jellyfish”. Like jellyfish, Medusa can sting an enemy with its tentacles. In Greek mythology, Medusa was one of the three Gorgon monsters. Anybody who got a glimpse of her face became petrified. Medusa is a security system which can extend the overall security of your Linux system. Medusa consists of two parts. The first is a set of small patches to the Linux kernel, and the second is a user-space security dæmon (authorization server) called Constable.

You may ask, “Why do I need a security system such as Medusa?” The answer depends on many factors. If you have a machine at home, you'll probably not need it. If you have a well-known and frequently used Internet server, you may have use for it. Why? Because the UNIX security scheme seems to be insufficient nowadays. Yes, it's really simple (like the whole UNIX principle), but it has many limitations. Just to mention two of them: you have no system rights at all as an ordinary user, and all rights to the whole system as root. So, when somebody breaks in using any network dæmon, he can do anything he likes inside, e.g., graphics subsystem or low-level disk operations.

Another problem connected with this is that any program which wants to use a socket with a number less than 1024 must have root privileges. In those ancient days of 4.1aBSD systems, this seemed to be a good idea; now, when combined with the limitations of BSD TCP/IP stack and buffer overflow problems, our computers are living in hard times. While Medusa cannot change those things, it tries to eliminate their impact.

The basic idea behind Medusa is really simple. Before execution of certain operations, the kernel asks the authorization server (Constable) for confirmation. The authorization server then permits, forbids or changes the operation. The authorization server and kernel talk to each other through the special device: /dev/medusa. In this way, an administrator can create his own security model, which can complete or override the original UNIX model. I have told you the principle is simple; however, the actual implementation is a bit complicated. If you are interested in how, see Resources.

In 1995, the authors of Medusa were administrators of university machines. Since there were many users, someone often cracked a machine and used it as a base for nasty activities. Instead of taking part in a never-ending race with crackers and their exploits, they decided to achieve better security in a different way. In 1996, an ancient preliminary version, old Medusa, was born. After more development, a new generation called Medusa DS9 appeared in July 1998.

Some Usage Examples

Constable is driven by its configuration file, which is usually /usr/local/etc/constable.conf. Here you can specify all the security settings you need. This configuration file can be considered a simple program. It has functions, conditions, events, blocks, etc., and is very similar to C. Constable is event-driven. When any event (syscall, VFS operation, signal) occurs, an appropriate action is executed. In the first example, we will protect one file against deletion. The Constable configuration is in Listing 1. Now, when someone tries to delete this file, Medusa returns the messages shown at the bottom of Listing 1. Hmmm; it seems as though the /tmp/delme file is undeletable, even by root.

Listing 1

Of course, it's not too useful to have one undeletable file. So, have a look at Listing 2, in which we will set a booby trap for every non-root user who tries to run the **ifconfig** program. This configuration is still not too useful, but does show how you can redirect execution of any program to something else. Thus, you can protect some sensitive programs against execution by some users. You can also redirect any file operation (access, unlink, read) to another file. For example, users who are coming to one machine from the network can have a different passwd and shadow file than local users.

Listing 2

In order to make Medusa really usable, we have to go deeper inside its functionality and configuration. Medusa uses the concept of virtual spaces. That means you can assign any process or file (inode) to one or more virtual spaces. Processes in one virtual space cannot see, change or influence processes or files in another virtual space. So, you can quite easily separate a few critical parts of your system from other parts. For example, you can have your Sendmail and FTP dæmons in virtual space number one and other system-critical files (such as the /etc directory, kernel and user home directories) in virtual space number two. Even if somebody exploits the FTP dæmon, he can do nothing, other than look at **ftpd** configuration files and the content of your public FTP directory if Medusa is properly configured.

In Listing 3, we will create two virtual spaces, one for the whole file system and another for the /tmp directory. Then we will protect the file /tmp/delme against deletion. When somebody tries to delete this file, he will be “fired” from the /tmp virtual space and will not be able to access any file in the /tmp directory or its subdirectories (even if they are mounted from another disk). This example can easily be modified to disallow this user from seeing the entire file system (by setting **vss** to 0). Keyword **vss** changes the virtual space a process can see. Virtual-space variables are usually modified by numbers in binary format (0b)

which sets individual bits of those variables (10 binary is 2 in decimal). As you can see from the listing, you can make comments as in C programming using // or /* */ characters.

Listing 3

Except for virtual spaces, information on operations, which have to be confirmed by the security dæmon, can be stored for each process and file. Actions for a file can be access, creation or deletion; for a process, they can be fork, exec, signals and so on. For operations you don't care about, the control system works the same as without Medusa installed. For operations you want to control (such as unlink in the first example), the system asks the security dæmon. Now we will monitor and control execution of suid programs, as shown in Listing 4. We will disallow execution of suid programs (like **su** and **ping**) for users who connect using the TELNET protocol. It can be easily modified to control **ssh** connections too, so a locally logged-in user can su to root, for example.

Listing 4

You can now ask, “What if somebody does write and compile his own TELNET dæmon?” The answer is, of course, that this configuration of Constable will allow him to run suid programs. To be 100% sure, you can monitor system calls. It can be done by keyword to **syscall**. In the variable ***action***, there is a syscall number stored (102 is socketcall), and in variables **trace1** to **trace5** there are parameters specified for this syscall. This configuration (see Listing 5) monitors every process that starts to use a network because the socketcall, **syscall 102**, must be used to do it, either with incoming or outgoing connections. The **lpeek** keyword reads data from the given address (**trace2**, the second socketcall parameter) and stores it into the variable *$x*. This example also shows the usage of user-defined variables. When a network connection is used, a function named **doit** will be called. If another syscall is called, the trace for this syscall is switched off. By default, the trace is switched on for all syscalls. To switch it off, you can use the **trace_off** keyword.

Listing 5

We are now coming to the most advanced feature of Medusa. You can force execution of any code in the context of any program. To simplify that process, there is a special library called Mlibc developed and included with Medusa. Mlibc (which stands for Medusa or Mini libc) is a small library, providing declaration of many standard functions, structures, macros and constants. When you link your “force” code with Mlibc, you'll get a program which can be executed inside a controlled process, as if it were compiled as an integral part

of it. In Listings 6 and 7, we forced execution of the **exit** function when any program tried to delete our well-known /tmp/delme file. The principle is simple; first, we will compile the “force” code (**exitme.c**) using

```
gcc -O2 -c -o exitme.o exitme.c
```

Then we will link it with Mlibc:

```
ld -r -o exitme exitme.o mlibc.o
```

After that, we will force execution of this code in the Constable configuration file in the event of unlinking of the /tmp/delme file.

Listing 6

Listing 7

Remember, *don't* use these to create your own impregnable Linux castle. They are truly just examples.

Among other nice features, Medusa knows and supports POSIX capabilities, so you can monitor and alter capabilities according to your needs.

Installation

What Next?

The Medusa development team is working heavily on it. They want to make Medusa a bit more object-oriented. That means you'll be better able to handle system objects to set their properties. They want to port it to other platforms (currently, only Linux/Intel is supported) and create a front end which will allow the administrator to create and configure security schemes more easily.

Medusa was not tested extensively on multiprocessor systems, but those who tried it didn't report any serious problems. Medusa needs a bit more detailed and better documentation. Constable configuration scripts can be truly complicated, so some form of automatic configurator will probably appear in future versions. You can write your own front end to Medusa, which can create the needed security model and implement it by using Constable configuration.

Medusa is a very interesting system, not only with security functionality. It will probably never appear in the official kernel, but can be used as an add-on package to increase the security of your Linux system.

Resources

Good/bad

Robert Dobozy (robo@idata.sk) is a SAP R/3 Technical consultant. He has worked with Linux since 1995, and is the co-founder and current president of the Slovak Linux Users' Group (SkLUG). All his free time is spent with his 20-month-old daughter and programming, mostly in Perl and PHP.

Archive Index Issue Table of Contents

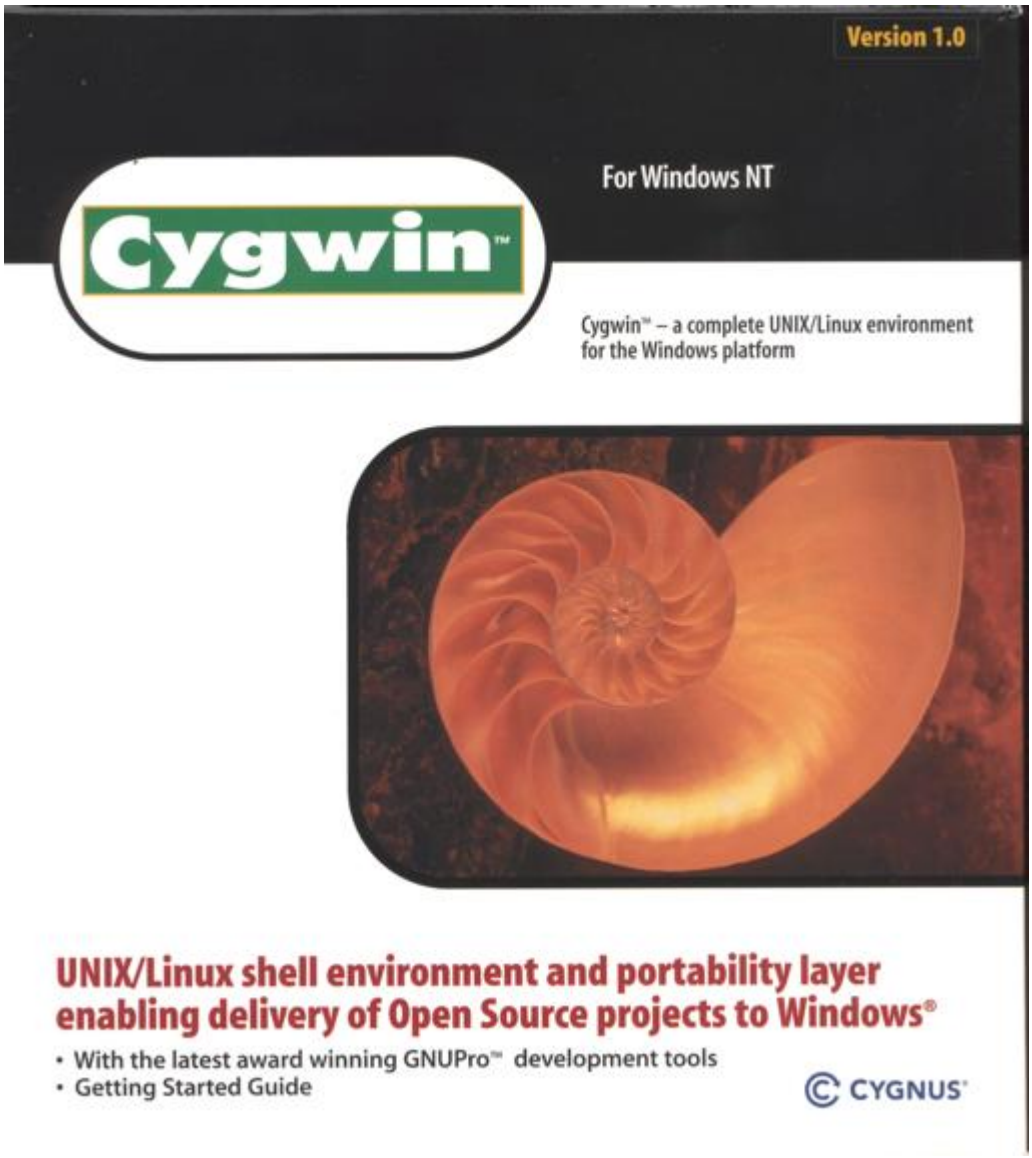
Advanced search

Cygwin: For Windows NT

Daniel Lazenby

Issue #75, July 2000

Cygwin is a port of GNU development tools to Windows NT. This port also brings a Linux/UNIX environment to the Windows platform.



- Manufacturer: Cygnus Solutions

- E-mail: info@cygnus.com
- URL: http://www.cygnus.com/
- Price: $99 US
- Reviewer: Daniel Lazenby

During 1999, Cygnus released products such as Cygnus Insight, Source-Navigator and Code Fusion. These tools focused on providing a better-integrated, Linux-based development and debugging environment. Cygnus recently added another tool to the Cygnus development tool chest, named Cygwin. With Cygwin, Cygnus takes a step in a slightly different direction. Cygwin is a port of GNU development tools to Windows NT. This port also brings a Linux/UNIX environment to the Windows platform.

You might ask, why would I want it? What does this do for me? The answer is, “it depends.” Are you a developer who wants to bring your Linux/UNIX open-source projects to Windows platforms? Do you want to develop open-source Windows applications? Or are you a Linux/UNIX administrator with the responsibility of also administering Windows NT workstations and servers? Cygwin can support all three situations.

Shells and Power Tools

Once Cygwin is installed, an administrator has the same shell interface on both Windows and Linux/UNIX platforms. Many traditional Linux/UNIX shell commands are included with the product. In the two default directories /bin and /contrib/bin, I counted about 300 commands and basic Linux/UNIX tools. (In this context, I am calling a command such as **wc** a tool.)

In addition to NT's GUI administrator interface are several console NT administrative commands. These include a series of “net...” commands that may be used to stop, start, pause and resume services. For example, these commands can be used with printers, print queues and shared items. There are also commands to change certain user account attributes. The command **net send** is used to send a message globally or to a specific user. A set of TCP/IP-related commands is also available.

The ability to intermix the Windows console commands, like those above, and Linux/UNIX commands in the same program is the result of Cygwin's integration with Windows. Jointly, these commands, tools, Linux/UNIX programs and shell scripts may be used to manage a Windows platform. I did have to play a little with escapes and quotations to get the Windows commands to execute properly from within bash scripts.

Several open-source Internet d<\#230>mons are included in the Cygwin product. **telnetd** and **ftpd** are examples of the included **inetd** d<\#230>mons. Capabilities such as **rlogin** are also provided. Combined, these tools make it possible to do system administration of Windows platforms remotely. Some inetd d<\#230>mons duplicate Windows commands. I kept getting a syntax error on a couple of commands, until I used “which” to learn the command's real location. A slight change in Windows pathing corrected my syntax problem.

Most administrators have a favorite scripting language and set of power tools. Cygwin scripting languages include bash, ksh and tcsh. Power tools such as Perl, Tcl/Tk, awk and sed are included. I also found both the vim and xemacs editors. I got a strange behavior with xemacs, though. The xemacs file-open menu option caused a dialog box to appear momentarily. All other xemacs menu options seemed to behave normally, as did the traditional Ctrl keyboard commands.

Cygwin does not automatically build a POSIX-compliant directory structure. Therefore, the library you need may be in a different location. I had to make pathing edits to a couple of the supplied awk scripts. Constructing a POSIX directory structure and properly linking to directories should reduce pathing edits.

Familiar GNU Tools

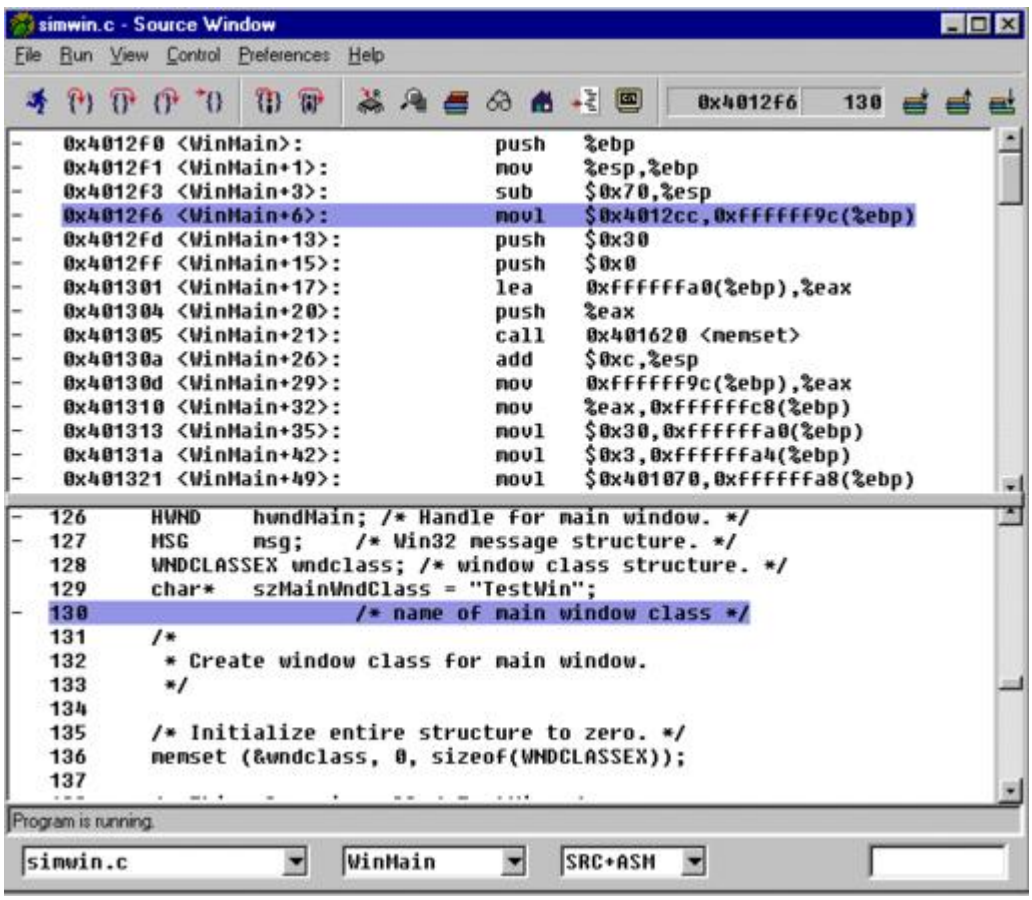


Figure 1. Source Window

Cygwin contains eleven GNUPro tools optimized for Intel Pentium processors. These tools include GNUPro compilers, linkers, assembler and comparison tools. The Cygnus Insight Visual debugging tool is also included in the package. Insight was formerly known as GDBTk. Figure 1 is a screen capture of the Insight debugger's Source Window. The October 1999 issue of *Linux Journal* contains a review of GDBTk.

Cygwin may be used to develop both console and GUI-based applications. The cygwin.dll portability library makes it possible to write and compile Windows applications with GNU tools. This library also supports the ability to port Linux/UNIX open-source projects to Windows. A sizable subset of UNIX SVR4, BSD and POSIX APIs have been integrated into Cygwin. These APIs make it possible to develop traditional Linux/UNIX programs on a Windows platform. Combined, these features and tools provide a standard Linux/UNIX GNU development environment on a Windows platform.

Using Cygwin to compile a console-mode C application on a 300MHz Pentium II with 64MB of RAM produced a timely response. A couple of Perl and awk scripts also performed well.

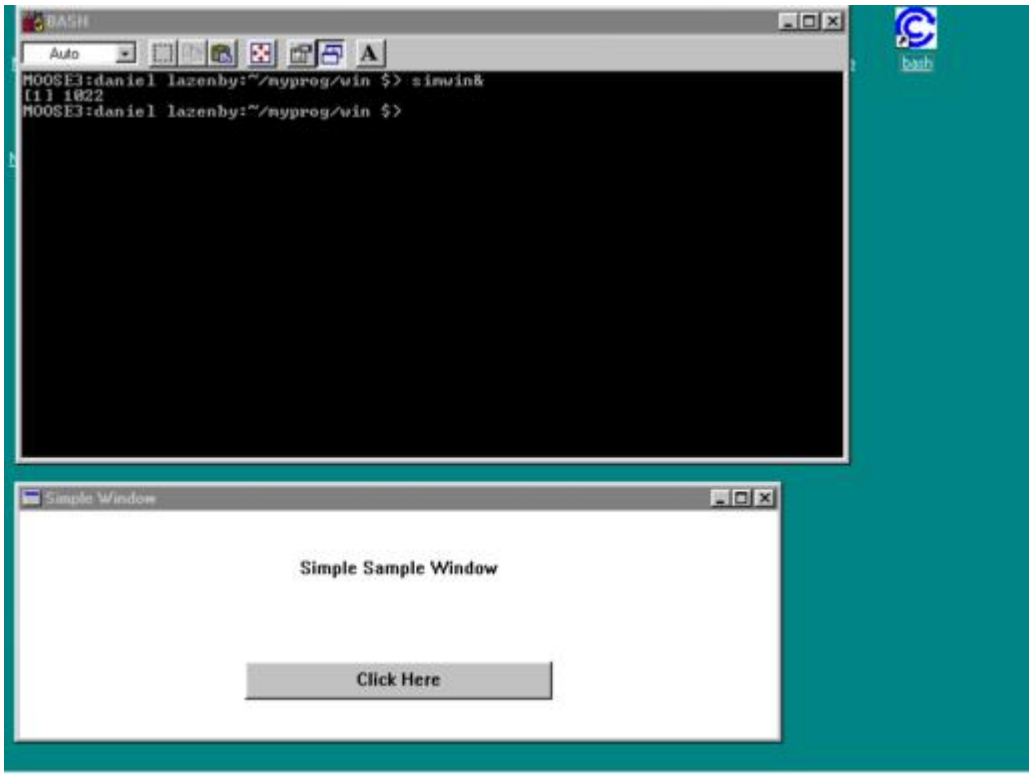


Figure 2. Sample Window

Simwin.c contains code for a simple no-frills window. Compiling from the command line performed well. Issuing the simwin.exe command at the bash

shell prompt caused the sample window to appear on the desktop as though it belonged there (see Figure 2). Resource files and DLLs are required to compile more robust Windows applications. Cygwin includes the capability to use custom resource and DLL files. The documentation briefly highlights building and linking to DLL and resource files. I am not a Windows developer, therefore, I was not able to exercise some of Cygwin's Windows compiling features seriously.

The documentation says Cygwin uses standard Windows APIs. According to the documentation, use of standard Windows APIs means Cygwin will keep functioning whenever Windows is upgraded to the next release.

As with the other Cygnus tools, Cygwin is under the GPL. Be sure to review the various licensing terms and conditions before distributing your applications. All programs or applications developed with these tools must be open source. This does not mean you can never use the tools to develop commercial applications, however. Contact Cygnus for information regarding licensing if you want to do commercial work.

Installation and Setup

Cygwin requires at least a Pentium processor with 32MB of RAM and 100MB free disk space. The supported operating system is NT 4.0 with at least Service Pack 3. Plan to allot more than 100MB for this product. Install Shield suggested 300MB would be required for the basic product and contributed files. When source code was also selected, the storage requirement went up into the 1GB+ range. On my system, I selected the basic product and contributed files. They appear to use about 140MB of disk space.

Cygnus uses the very reliable Install Shield to install Cygwin. Install Shield properly installed and registered Cygwin with my Windows system. Pay heed to installation instructions. Cygwin should not be installed in any directory with spaces in the directory or path names. Once installed, traditional Linux/UNIX escaping, quoting or tabbing is necessary to get to directories with spaces in their names.

After installation, several steps are required to set up the Cygwin environment. A few of the tasks that need to be done include creating mount points for existing disk drives and directories; establishing the desired POSIX Cygwin directory structure; establishing environment variables; establishing and verifying file and path naming conventions; and customizing the shell.

Security

Platform security is an area requiring a few moments of attention. Maintaining platform and file system security is important, and maintaining security on platforms containing business data is critical.

Cygwin has two security-related options: NT Extended Attributes (ntea) and NT Security (ntsec). Ntea can work with both FAT and NTFS file systems. The NTFS file system is required by the ntsec option. Enabling the ntea option permits NT file permissions to behave like UNIX file permissions. This option works best with NTFS file systems. It does work with FAT file systems. After using ntea a while with the FAT file system, one can expect a major performance hit.

NT can reproduce the POSIX security model. Enabling the ntsec option will cause file ownership and permissions, and process privileges to be treated in a UNIX-like manner. For processes, this means one can start a process with the group owner being the administrator. Anyone who is a member of the administrator group may send signals to the process. Normally, only the ID creating the process has permission to send signals to the process. Proper function of ntsec requires both /etc/passwd and /etc/group files. Using supplied tools, these files may be generated from the NT security files.

I am not sure how adding Cygwin to an NT platform impacts maintenance of a desired security posture. Once fully integrated, altering an established NT security posture with Cygwin seems possible. I do have one general question: are two sets of security reviews and corrective actions now required, one for NT and one for Linux/UNIX (Cygwin)? I believe the Cygnus documentation should include some discussion or clarification related to which NT/Cygwin security attributes take precedence.

Documentation and Support

A couple of sections in the manual provide concise information and guidance on Cygwin functions. Much of the rest is a collection of facts about the product. These facts have been organized to appear as though they are providing guidance. One example is the “Setting up Cygwin” topic. This topic presents eight bullet items, pointing to various pages. Each set of pages discusses aspects of setting up Cygwin, so one must go to eight different places in the manual to read about the Cygwin environment. This technique is used several times throughout the manual. I found the process of piecing information together tiresome. I never got the feeling I had seen all the information I needed to configure Cygwin properly.

On-line documentation includes man pages, Texinfo and HTML pages. Some man pages are no longer actively maintained. Instead, one should use the

Texinfo source. Texinfo is the authoritative information source for many traditional man pages. Cygwin has complete control over the HTML files distributed with the product. Cygnus suggested the HTML files are the more accurate source of information at the time the product is shipped. Therefore, the HTML documentation should be consulted first.

The web-based FAQ is not very big, however, the few minutes it takes to browse this FAQ are worth the time. It provides an itemized list of implemented ANSI C and POSIX.1 API calls. All of the compatible APIs are also listed in the local on-line and printed documentation.

The Cygnus documentation is considerate of individuals who have Windows experience. References to Linux/UNIX and Linux/UNIX programming for the non-Linux/UNIX-initiated are provided.

Product installation support is available. Once the installation support ends, assistance may be obtained from the Cygwin mailing list, cygwin@sourceware.cygnus.com. Another source of information is the sourceware.cygnus.com/cygwin web site.

The Good and the Bad

Daniel Lazenby (d.lazenby@worldnet.att.net) first encountered UNIX in 1983 and discovered Linux in 1994.

Archive Index Issue Table of Contents

Advanced search

Understudy

Daniel Allen

Issue #75, July 2000

Understudy is a software-based server clustering utility that implements load balancing and failover protection for Linux (Red Hat, Debian and Slackware), Solaris, Cobalt, FreeBSD and Windows NT.

- Manufacturer: Polyserve Software
- E-mail: info@polyserve.com
- URL: http://www.polyserve.com/
- Price: under $1000 US, including support
- Reviewer: Daniel Allen

Every host on a network has down time, from the coolest RaQ to the lowest NT 486. The job of keeping down time to a minimum falls to the system administrator. Various solutions are available, spanning the range of needs and budgets. One way is to use high-availability servers with Fiber-Channel RAID arrays, multiple redundant CPUs and power supplies and a transaction-oriented file system. The servers can be arranged behind $50,000 load-balancing and failover systems to swap out servers automatically upon failure.

A solution at the other end of the cost spectrum is running a backup server which is manually switched with the primary server when necessary. In this scenario, if a server fails unexpectedly, it can be many minutes or hours before the poor system administrator can make the switch. This solution is both inelegant and widely used in company networks.

A third way is “server clustering”, or making multiple servers appear to users as if they were the same server, for fault-tolerance and load-balancing purposes. Very interesting efforts are underway to offer completely Linux-based server clustering solutions. These include the open-source Linux Virtual Server, and other work being done by the High Availability Linux Project. These projects show great promise, and they may be the right answer for sites wishing to be

close to the bleeding edge. However, small businesses need fully supported solutions that do not require substantial modification to their existing, possibly heterogeneous, networks. This is the gap which Polyserve hopes to fill with Understudy. As you will see below, I think it does the job nicely.

Understudy is a software-based server clustering utility that implements load balancing and failover protection for Linux (Red Hat, Debian and Slackware), Solaris, Cobalt, FreeBSD and Windows NT. It supports between two and ten heterogeneous servers in a cluster, all of which must be located on the same DNS subnet. Polyserve hopes to release a newer version soon that circumvents the single subnet requirement. A cluster of servers can provide any service, including web, mail, news or file sharing.

When a server goes down, it is marked inactive within the cluster and another server takes its place in seconds. When the server comes back up, it is immediately reintegrated into the cluster. By using Understudy in conjunction with a load-rotation scheme called “round robin DNS”, a site can also provide simple load balancing. Load balancing requires one additional IP address for each server in the cluster. Simple failover requires only one IP address for a “virtual host”, which is how users see the cluster.

Installation

Installation of the Red Hat Linux version was simple. After reading the release notes, I wouldn't expect major difficulties on other platforms. Understudy provides a “quick-start” white paper on their web site which is recommended reading, along with the white papers on web server specifics and on round robin DNS. They are easily understood if you have ever configured a web server or changed your DNS configuration.

I downloaded the RPM for the free 30-day trial and ran **rpm** as root to install it. After installing files, Understudy started its dæmon and reminded me to assign a password for the administration tool, which I did. I repeated this process on each of the four servers that would make up the cluster.

Configuration: A Tour

The four servers were administered remotely, so I could not run the graphical local administration tool, which requires X on the server. However, Polyserve also offers a graphical remote administration tool, available for either Red Hat or Windows 98/NT. I downloaded and installed the RPM on my local Debian system using Alien, the Debian RPM manager. There were no serious problems, although I needed to modify the startup script it created to properly point to the copy of the Java Runtime Environment (1.1.7) and the libraries it also

installed. It filed everything away in /usr/local/polyserve with a startup script in /usr/bin.

Figure 1.

Next, I set up my first cluster with failover protection using a pair of servers. This requires a single “virtual host”, which is simply an unattached IP address in the same subnet as the real hosts. This was a straightforward process, following the instructions in the Quick Start Guide. Firing up the graphical administration tool, it prompted me for a cluster IP and password. I supplied the IP of the first server and was presented with the main window (Figure 1). The main box, titled “Cluster status”, listed the name of the server I supplied, with the reassuring status of “OK”. The menus include “File”, “Cluster”, “Server Log” and “Help”. “Cluster” has the most interesting choices: “Add Server”, “Add Virtual Host”, “Add Service Monitor to Selected Host”, “Delete Selected Item” and “Update Selected Virtual Host”. I chose “Cluster **-->** Add Server” and was prompted for a server name or IP. I filled in my second server. Voilà: the “Cluster status” told me both servers were okay. So far, so good.

Now, to add my first “virtual host”. This requires adding a new host in your DNS tables (such as in /var/named on your DNS server):

```
  virtual1  60   IN       A  150.1.1.1
```

This simply adds a new host name with a Time To Live (TTL) of 60 seconds with its Address.

I added this new line with an appropriate IP address for my subnet, and restarted the named dæmon. Back in the administration tool, I selected “Add Virtual Host”. It prompted me for the name or IP of the virtual host, and listed selection boxes to determine which real server was to be the primary server and which was to be the backup. I entered my information.

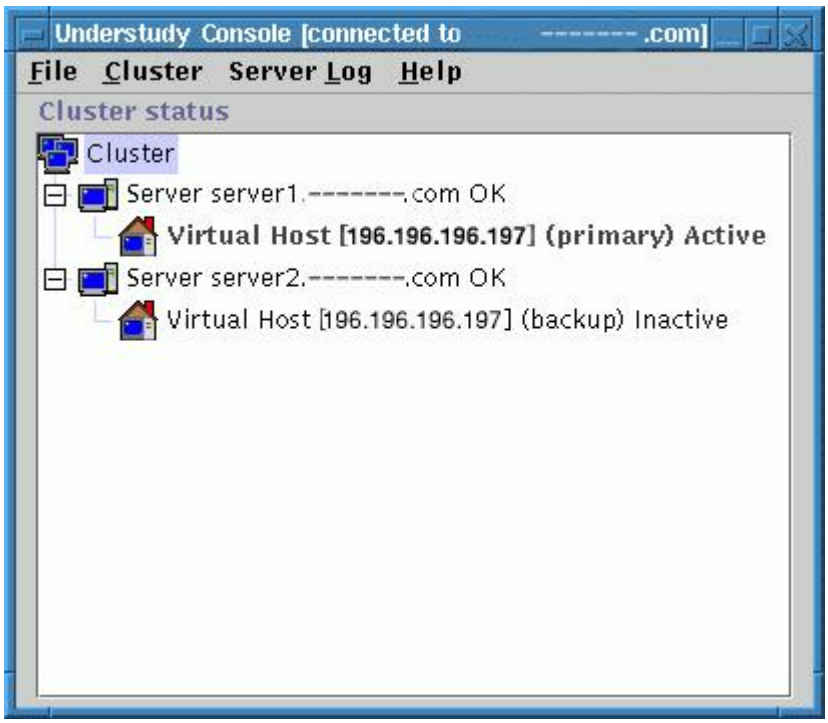


Figure 2.

At this point, the Cluster status looked a bit more interesting (Figure 2). It listed both real servers, and subheadings described that the first server was Active for the virtual host, and the second server was Inactive. I tried to telnet to the virtual host. It connected me to the first server. I went back to the administration tool and deleted the virtual host. I re-added it, but this time, decided that the second server would be the primary server for this virtual host. The display reflected the change immediately. I telnetted to the virtual host. Sure enough, it connected me to the second server.

What's happening behind the scenes is something like this: Understudy runs as a dæmon on each server. The IP address of the virtual host is automatically aliased to the primary server. A small amount of traffic is constantly passed between the real hosts, via broadcast ARP messages. Through the dæmon, each host knows which is acting as primary. When the primary server goes down, the backup immediately reassigns the virtual host's IP address to itself. It continues to listen, so it can release the IP address when the primary server comes back up.

Note that Understudy will not allow you to use an IP address already assigned or aliased to a real host as a virtual host. I imagine that otherwise it would be easy to “hijack” the IP address of someone else's host in your subnet.

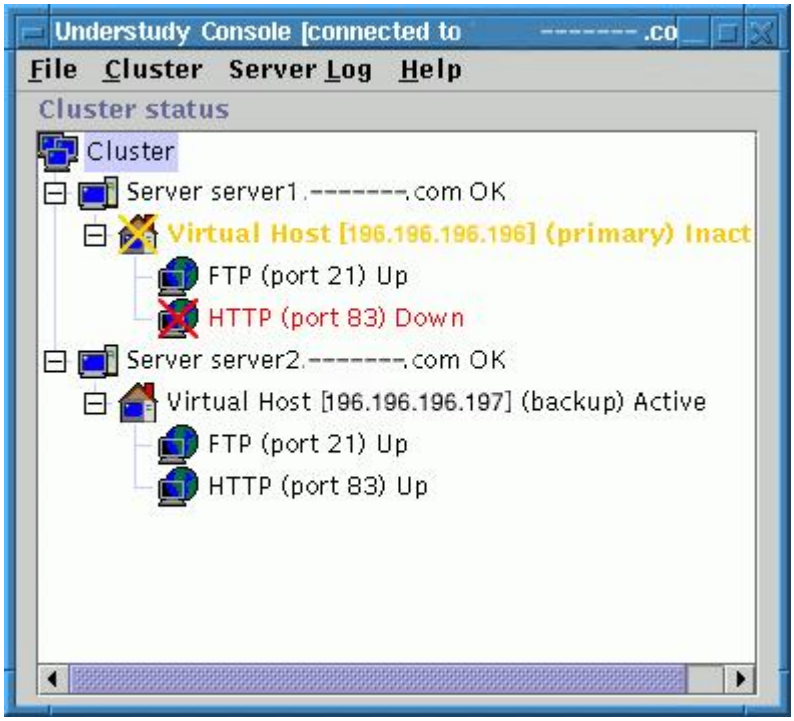


Figure 3.

Next, I set up a “service monitor” (Figure 3). This allowed me to choose particular ports to monitor, such as for mail, web, FTP or TELNET. If the active server does not respond at that port, the inactive server will step in. I selected HTTP, and the Cluster Status reported that the web server was up on both real servers. I verified, using Lynx, that requests to the virtual host went to the primary server, unless a service it was monitoring was down, in which case requests went to the backup server. In all cases, Lynx showed the URL of the virtual host name, as expected.

For the next test, I set up round robin DNS. Round robin DNS is a feature built in to name servers such as BIND (versions 4.9 and up). Round robin allows servers to share loads transparently by rotating between any number of IPs for a given host name. The only problem is that no correction is made if one or more servers go down, so out of every cycle of requests, some are sent to a dud server. With Understudy, this is no longer a problem. You can set up round robin DNS for a number of virtual hosts, where each virtual host has a different primary. If any server is down, its requests are sent to the next secondary. Full examples for doing this are available in the Understudy documentation. These instructions were reasonably clear and easy to follow. At the conclusion of a couple hours of work, I had a fully redundant set of servers with no interruption to existing services on the servers.

One final function of the administration tool is a server log, which accesses dæmon messages for each server in a cluster. This brings me to a minor complaint: the logs are somewhat difficult to parse. It would be nice to see an integrated cluster log, providing a summary of the server logs.

Security

The instruction manual states that all messages between the remote console and the servers are signed for security. The remote console will work with a firewall, and the servers will record a message to the log if somebody tries to "replay" internal UDP messages to try to confuse the servers.

Use in the Wild

To use Understudy in a production environment, you will want to configure any services (such as web, FTP, mail, TELNET) to respond to the virtual IP addresses (as well as the real IP addresses). There are complete instructions on adding the Virtual Hosts to your Apache or Microsoft IIS web servers.

Understudy does not automatically mirror information from one server to another, although Polyserve has stated that is a goal for a future version. You should think about whether servers will need to have up-to-the-second data copies, and plan accordingly. Some database applications might require extra hardware, such as a RAID array connected to multiple servers. I would visit the Linux High Availability web site, at http://linux-ha.org/, for LAN mirroring ideas.

Pricing and Support

Understudy can be downloaded and demoed free for 30 days, during which time technical support via e-mail is also free. It is trial-ware; during the trial period, the dæmon will turn itself off after two hours of use, requiring you to restart the dæmon. A permanent license with a service contract costs a little under $1000. Without the service contract, the price is roughly half as much. Polyserve has various support options, so you should contact them for a complete listing. I have good things to say about their customer service. Owing to network problems related to the Understudy software (but not the fault of the software), I spent a fair amount of time talking with Polyserve's support staff on the phone. They were technically competent and very helpful in pointing me toward a good solution to my problems.

The documentation is downloaded from the site in PDF form. It is complete and useful, although the product manual shows a few signs of poor editing. Unlike the white papers, the manual incorrectly states that you can use it with only two servers, not ten.

For further help, there is also a six-page help facility which describes the program's operation. For some reason, on my computer the help pages kept throwing Java exceptions. However, the information was still accessible, and they were minor distractions. These were the only bugs I found in the program.

Conclusion

Understudy should be a godsend for the beleaguered system administrator, server farm or ISP that needs to have services up 24/7 through reboots, failures and planned outages. One strong point in favor of this software is its ability to work in any network with all kinds of hosts. Even if your backup server is a 33MHz 486, Understudy can keep your network limping along until you can fix the primary server. It seems to be a good solution for those who cannot afford $10,000 to $50,000 for a dedicated failover and load-balancing server, or simply are not willing to pay a $2000 license for each server in the cluster.

The Good; The Bad

Daniel Allen (da@coder.com) discovered UNIX courtesy of a 1200-baud modem, a free local dialup and a guest account at MIT, back when those things existed. He has been an enthusiastic Linux user since 1995. He is President and co-founder of Prescient Code Solutions, a software consulting company located in Ithaca, New York.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

The System Logging Dæmons, syslogd and klog

Michael A. Schwarz

Issue #75, July 2000

Take command of your log files by learning to handle those pesky logging daemons.

Most UNIX-like systems since the early days of BSD (and Linux certainly falls in this category) have provided an API for application programs to send log messages to the system, where they can be centrally handled at the discretion of the system operator. Prior to the creation of this facility, each application program would handle log messages in its own way. Some would write to STDERR, some would write to a file, some would write to a pipe, and some would offer all these options or more.

As the number and complexity of applications on a system grows, so, too, does the complexity of the system administrator's job. Applications and their messages vary widely in their significance to certain audiences. If a number of applications are considered “critical” and their status is the system administrator's responsibility, he does not want to search to find out where and how every critical application logs its status. That's where **syslogd** comes in.

syslogd

BSD added an API for logging to the standard library. Linux also offers it. This API consists of three function calls:

```
#include <syslog.h>
void openlog( char *ident, int option, int facility)
void syslog ( int priority, char *format, ...)
void closelog( void )
```

An application that wishes to use syslogd for logging uses these calls. A brief introduction to the use of this API can be found in the sidebar “Using the syslog API in Applications”. The one critical thing to know is that all messages from this API have, at a minimum, a “facility” and a “priority”.

Using the syslog API in Applications

Facilities include **LOG_AUTHPRIV**, **LOG_CRON**, **LOG_KERN**, **LOG_DAEMON** and so forth. These serve to identify the “system” of origin. Note that it is not the “program” of origin. For example, many different programs make up UUCP, but they all log as **LOG_UUCP**. The program name can be a component of a log message, but this has nothing to do with the facility. Some programs will log as more than one facility. For example, **telnetd** might log failed logins as **LOG_AUTHPRIV**, but it might log other messages as **LOG_DAEMON**. “Priorities” specify the “severity level” or “level of attention” the message merits. We'll discuss these concepts at greater length throughout this piece.

Our primary focus here is the tool that handles messages sent via this API. While syslogd was originally developed for 4.2BSD, we are going to cover the version that ships with most Linux distributions today, specifically syslogd version 1.3-3. The syslogd utility normally reads a configuration file at startup to determine how messages are to be handled. This file, normally /etc/syslog.conf, tells syslogd what to do with messages. Much of the rest of this document will describe how to use syslog.conf to customize logging on your system.

Listing 1

The syslog.conf file follows the more or less ubiquitous UNIX convention of using the pound sign (#) as a comment character. We'll use the sample syslog.conf file in Listing 1 for the rest of our discussion. This is the “out-of-the-box” syslog.conf from my Red Hat 6.1 laptop. I use other distributions personally, primarily Debian and SuSE, but Red Hat seems to be the most popular. Let's understand what this file is doing.

The Selector

“Rules” in syslog.conf are a single line which consists of two parts. The first is a “selector”, which specifies the set of messages on which the rule is to act. The second is an action, which specifies what is to be done with messages that match the selector.

The selector is further divided into a “facility” and a “priority”. Yes, these match the terms mentioned above in the brief description of the syslog API. The facilities and levels have numeric values and you can use them in the syslog.conf file, but it is strongly advised that you do not. Symbolic values are supported, and if the syslog API is ever changed, the numeric values might change, whereas one would expect that the symbolic names would be kept in alignment with any such change. So, be safe and use the symbolic names.

The symbolic facility names are **auth**, **authpriv**, **cron**, **daemon**, **kern**, **lpr**, **mail**, **mark**, **news**, **security** (same as **auth**), **syslog**, **user**, **uucp** and **local0** through **local7**. The **security** keyword is deprecated in favor of the **auth** keyword. The **mark** keyword is internal and should not be used by applications; the syslogd program can be set to produce a **mark** periodically, which provides a means to tell if you aren't getting messages, because none are being generated or because syslogd has died. The rest of them correspond to the major subsystems on your Linux box.

The priority keywords are **debug**, **info**, **notice**, **warning**, **warn** (same as **warning**), **err**, **error** (same as **err)**, **crit**, **alert**, **emerg** and **panic** (same as **emerg**). The keywords **error**, **warn** and **panic** are deprecated and should no longer be used.

A selector consists of a facility and a priority separated by a period (.) character. Thus, **mail.crit** would select all critical messages from the mail facility.

The default behavior of the BSD syslog system is for all messages of the specified or higher priority to be handled by the action. The Linux syslogd does the same by default. It does have a number of extensions, however.

You may use the asterisk (*) character to indicate all facilities or all priorities (depending on whether it appears before or after the period). Thus, the **authpriv.*** line in the example sends all messages from the authentication facility, no matter what priority, to /var/log/secure. You may use the special priority **none** to indicate that *no* messages from a given facility rule are to be acted upon by the action.

You may specify multiple facilities with the same priority in a single rule by listing the facilities separated by commas (,) before the period. Thus, the **uucp,news.crit** line sends all critical and above priority messages from the mail and news facilities to /var/log/spooler.

You may specify multiple selectors for a single action by listing them separated by the semicolon character. Each subsequent selector may override the previous. Thus, the ***.info;mail.none;news.none;authpriv.none** rule would send all messages above **info** priority from all facilities (because of the *) to /var/log/ messages, *except* messages from mail, news or authentication facilities (because of the **none** keyword and because rules are applied in order, left to right).

More than one rule may apply to a message! It is important to understand that a message will be sent to *all* actions with matching selectors. It is not as if a message, once matched, is gobbled up. That means you can store a single message to multiple actions if it is matched by multiple selectors.

There are more priority selection extensions. First, remember that the default is to select messages of the stated or higher priority. You may also reverse the sense with the exclamation mark. So, for example, a rule such as

```
*.!err   /var/log/routine
```

would send all messages *not* at **err** or above to /var/log/routine (a file meant for "routine" messages, apparently).

You can also restrict the selection to an exact priority instead of to a given priority and higher with the equals sign (=). Thus, the **news.=crit** rule out of our example would send only critical messages from the news facility to /var/log/news.crit.

Table 1

At this point, you may be a little foggy on what, precisely, these various priorities are meant to denote. Let's shed a little light on that issue by looking at the "Priorities" table.

A classic problem in designing software is trying to figure out an empirical way to tell the difference between a condition you would report as **crit** vs. **alert**. In fact, it is sometimes even more difficult to decide when you should use **notice** vs. **warn**. You won't find total agreement between packages on what level of message falls where. One difficulty lies in trying to decide who will read the log. An **emerg** to a business unit might be only a **warn** to a network administrator.

There's no one good answer to this problem. At least, by creating a uniform method for handling program messages, we avoid a proliferation of different reporting systems, and some conventions have emerged with time. Because we are able to match a message to more than one action, we can output messages to targeted audiences. For example, we could report all **auth** messages to the security department's home directory, but the system administrator might choose to receive only those of "crit" or above. The syslogd lets us do this.

The Actions

There are some drawbacks to using the "out-of-the-box" Red Hat syslog.conf file. Notably, all "actions" are basically to write to local files. The syslogd dæmon can do much more than that. Let's take a look at actions next.

Actions may send messages to any of these destinations:

- A regular file: this is what you see in our example. This is simply the name of a file to which the message is appended.

- A named pipe: named pipes, or FIFOs (First-In, First-Out), are a simple form of inter-process communication supported by Linux and many other operating systems. You create a named pipe with the **mkfifo** command; a FIFO appears in the file system. You tell syslogd it is writing to a FIFO instead of a file by putting a pipe character (|) in front of the FIFO name. Take a look at the man pages for mkfifo, both the command and the system call, and the man page for "fifo", which is a description of the special file. You read and write FIFOs with the normal file system calls. A description of FIFO programming is beyond our scope, although I can highly recommend the excellent book *UNIX Network Programming* by W. Richard Stevens, from Prentice-Hall.
- A terminal or console: if you specify a tty device (such as /dev/console), syslogd is smart enough to figure out that it is a device, not a file, and treat it accordingly. This can be fun if you have a dumb terminal—you can send all your messages to /dev/ttyS1 (for example) and get all your messages on the terminal screen while you work on your console. This is state-of-the-art 1970 technology—I love glass teletypes!
- A remote machine: now *this* is the true power. Let's assume you have many Linux boxes on a network. Do you want to log in to each to check their logs for certain conditions? Of course you don't, and you don't have to. Optionally, the syslogd listens on the network for messages as well. Just put an at sign (@) followed by the host name:

```
*.crit   @sol.n0zes.ampr.org
```

This last will send all critical and above messages from all facilities to sol.n0zes.ampr.org, which will then apply its own syslog.conf file to save them. Syslogd will not forward a message received from the network to another host: in other words, you get one and only one hop. This may be overridden with switches when syslogd is invoked. It seems like a reasonable thing to do, since even the possibility of circular message routing would be enough to scare the dickens out of any network administrator.

This capability has obvious advantages for centralized logging and log scanning for security violations and so forth. It also has obvious deficiencies. It is hard to maintain a complete log when your network is down. Take advantage of the fact that you can route messages to more than one action by making sure every message finds its way to a file before you send it to remote logger.

A List of Users

One feature that newer users of Linux may not be aware of is console messaging. This isn't used very much any more, thanks to **talk** and **irc** and other much more interactive "chat" mechanisms with much cleaner user interfaces. You can, however, send a text message to any user logged in to your system

with the “write” command. This is an unpopular facility for several reasons. First, in today's windowed environments, a user probably has many “terminals” active and it is hard to know which one to write to. Second, if they are in the middle of some intense full-screen activity (such as editing a large file with **vi**) and you blast a bunch of text at them that confuses their editor and screws up their screen, they will not like you very much. Most Linuxes I have seen default their users to messaging being off. This facility uses the same ability to write to a user's console to send messages directly to their screen. Just put a comma-separated list of user names as the action. Save this for truly critical stuff. You might turn this on to try it, but I bet you will turn it off again before too long.

Everyone Logged In

There is a similar method, called **wall** or write-to-all. This lets you send a text message to every user logged in to the system. The superuser can do this whether you choose to accept messages or not. This is how **shutdown** sends its warning messages. You can have syslogd send a message to everyone by this mechanism by specifying an asterisk (*) as the action. Save this one for the most dire of dire messages, if you must use it. This should be used for warning of an impending crash—anything less is probably overkill.

klogd

At this point, you may be asking what **klogd** has to do with any of this. The answer to that is simple. The kernel can't call the syslog API. There are many reasons for this. The core reason, and the simplest to understand, is that Linux actually provides two completely separate APIs. The more familiar one is the “standard library” used by user-space applications; this is the one that uses syslog. The other API is normally not used by applications: this is the kernel API code that runs as part of the kernel. This code needs services similar to those offered by the applications programming interface, but for numerous technical (and a few aesthetic) reasons, it is not possible for kernel code to use the application's API. For this reason, the kernel has its own entirely separate mechanism for generating messages. The klog dæmon, klogd, is an application process that ties the kernel messaging system to syslogd. Actually, it can also dispatch kernel messages to files, but most configurations use the klogd default, which is to prepare kernel messages, and in essence, resubmit them through syslog.

There is quite a bit more to klogd if you wish to delve into the depths, but for the purposes of this article, it is sufficient to know that klogd feeds kernel messages to syslogd, where they appear to be coming from the **kern** facility.

syslogd provides a powerful and simple mechanism for managing messages from multiple applications in a highly configurable manner. Its ability to

“demultiplex” the message stream makes using the syslog API an appealing option for applications developers, and I would encourage you to consider using that API in your own programs.

syslogd Switches

Signals

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

Archive Index Issue Table of Contents

Advanced search

Using Linux at Left Field Productions

David Ashley

Issue #75, July 2000

One programmer's experiece developing a Gameboy emulator on Linux.

Left Field Productions, Inc. is a game developer in Westlake Village, California, which concentrates on the console market, specifically the Nintendo N64 and Gameboy systems. Their web page is at http://www.left.com/. I joined Left Field in September of 1998 to do Gameboy programming. Since that time, Linux has been used on the company-wide network offering various services, all running on a lowly Pentium 90 with 16MB of RAM and a single 6.4GB hard drive. The machine is known by all as “the Linux box”.

The company uses 10BaseT for all networking. The Linux box also provides the main gateway to the Internet. It is running the **named** name server, and the Apache web server for company internal web pages. The Linux box has dual mechanisms for connecting to the Internet: one through ISDN, the other through a conventional 56K modem. ISDN is the preferred method, but during occasional outages, it is necessary to downgrade to the modem.

The Linux box provides public Samba services, making storage area available for file backups and exchange among the artists and programmers. The **fetchpop** program was used to pull mail off our ISP and forward it to individual accounts on the Linux box. When some problems with fetchpop arose, mostly due to the lack of a timeout feature during mail retrieval, I replaced it with **fetchmail**, which we are still using. Periodically, employees' preferred desktop mail client connects to the Linux box to retrieve their mail and send any outgoing mail off to the Internet. Sendmail also provides outgoing mail service.

The Linux box has enjoyed up times of more than 60 days. The few times it has been rebooted were due to planned outages for power utility service, and upgrades such as adding a new hard drive. As far as I know, there has never been a system crash. I think it is sometimes rebooted unnecessarily by some of the other Linux-literate employees, as it can be more convenient than

specifically restarting a single task after modifying a configuration file. Employees with lots of DOS/Windows experience have a kind of “when in doubt, reboot” philosophy.

When I first started at Left Field, we used a free assembler/linker combination for generating the code for the Gameboy ROM images, and we used commercial painting and graphic arts programs for art generation. The tools for converting and compressing graphics files were all developed in-house. We used Win32-based machines for everything. On my Pentium II 400 machine, a complete rebuild, where all source files are assembled and then linked, would take from 30 seconds to 2 minutes, depending on the project. We were working on a basketball game called *Kobe Bryant 3 on 3* and a Disney title called *Beauty and the Beast, A Boardgame Adventure.*

Once the ROM image was generated, we could download it to the Nintendo debugger system for testing, or more frequently, we would run it on a Gameboy emulator. The emulator was very usable, but it had quite a personality and was temperamental at best. I found if I exited the emulator and reloaded it a few times, it would invariably crash the system. Also, it was unusable for testing the sound aspects of the games—the quality of its sound emulation was painful to hear. The emulator was actually a DOS program working with an extender, and knew nothing about the Windows 95 windowing environment. I think legacy is the most likely cause for its instability.

After finishing up these first two projects, I found myself with some free time before the next project began in earnest. I had long been thinking of writing our own emulator because of my dissatisfaction with the emulator we were using—the frequent crashes and inability to get the source in order to repair it ourselves was quite frustrating. So, within a couple of months of starting at Left Field, I began some minor work writing an emulator for the Gameboy CPU. All my programming was done in C. I installed the public-domain DJGPP compiler and used it under DOS. In fact, I was able to significantly speed up build times by using the MAKE utility from DJGPP, replacing the Watcom MAKE we had been using.

Unfortunately, the demands of the project caused me to stop work on the emulator before I could even test it. Then, about a year later, I had some free time again and was able to get back to the code. This time, I chose to move away from Win32/DOS and switch to **gcc** under Linux—a move that made the programming infinitely more enjoyable. Surprisingly, there were only a few mistakes in the emulation code, and in a short time, I had the CPU “working”. It appeared to be behaving correctly. The next step was to emulate the video hardware of the Gameboy. For display output, I chose to use the SDL library, which is a multi-platform gaming library. One of the supported platforms is

Win32, so the benefit there was that any code I wrote could be used by other employees who were still running Windows. I was using SDL under an X Window System environment. After some solid work, I had the video emulation working quite well, and it was a joy to see ROM images actually work.

Finally, I had to add the sound-emulation code. This proved to be the easiest task, and after a short time, the emulator was producing quite accurate and acceptable sound, again using the SDL library. With a simple recompile using a cross compiler, an .EXE executable could be built. The emulator worked under Win32 as well. There were a few quirks related to sound under Windows 95 that had to be worked out. Windows proved incapable of servicing the audio interrupts at the 64Hz rate I had been using without problems under Linux. I had to compromise and lower the rate to 32Hz so Windows could keep up. I never determined whether the problem was in SDL's Win32 code or in Windows itself.

The assembler/linker we had been using offered a version for Linux, but I wasn't happy with it. The Linux source wasn't as up to date as the DOS version—the two versions were based off different source trees, and it was clear the DOS version had priority. My options were to use the older Linux version or port the DOS code to Linux. I chose instead to abandon the assembler and write my own. Using core code that originated from my own ACC C-like compiler, I managed to create an assembler with a syntax similar enough to the assembler we had been using. I took the opportunity to make changes in syntax when it was convenient. I knew how the assembler was going to be used, and some features weren't important, so I never implemented them.

In the end, my own assembler reported lines-per-minute assembly rates over 30 times faster than the old assembler. With the small source files, assembling each was practically instantaneous. The next part I needed was the linker. Again, I began with the ACC code and modified it to suit. Linking was also much faster than before.

Now, to test the assembler/linker combination, I took our game source trees and made the necessary syntax modifications in the source files. I used the *Beauty and the Beast* code, and spent about four hours going through all the files to get something without linker errors. Naturally, the resulting ROM image didn't work, but after a day of hunting for bugs in all parts of the system, I got a ROM image that actually came up on the emulator looking like the real game—very encouraging.

In a project like this, debugging problems can be tricky. When no single part has really been tested, a bug can be anywhere; thus, I found myself frequently hunting in the wrong places for bugs. Sometimes, I was surprised to find the

assembler actually did something right, and the emulator was to blame—and vice versa.

In testing the ROM images with the emulator, it became apparent I needed some debugging functions built into the emulator. Even before beginning work on the assembler, I had added disassembly capability to the emulator. Doing so had been very helpful in finding bugs in the emulator. After I had the assembler working, I added some nice features like symbolic debugging, break points, expression evaluation, memory viewing and instruction execution history. For text display and entry, I added a scrollback buffer, name completion and **tcsh**-style line editing.

Bugs became more and more rare, and were easier and easier to find. It was clear the new system was completely viable for developing, and an in-house suite of tools offered very strong advantages that we would never get by using outside software. For example, any desired feature could be added easily, since the source was ours. For portability, I had written everything in standard C. One thing I kept in mind was that at any moment I might have to retreat from Linux and switch back to DOS, and I wanted the tools to work there as well. The assembler and linker compiled perfectly with DJGPP.

I was pleased to note build times were reduced to almost nothing. A complete rebuild that had taken 30 seconds before took three seconds now, on the same machine. It must be noted that those times reflect two different operating systems as well as two different assembler/linker pairs, so an actual breakdown of how the speedup was occurring can't be made. I never bothered to do a detailed analysis; I was happy just to be using the new system.

On the other hand, my emulator, written in pure C, placed significantly more demands on the CPU than the DOS emulator we had been using. The author probably had hand-coded x86 code sprinkled liberally throughout. I also assumed this was the source of most of the crashes caused by that emulator.

To complete the Linux Gameboy development environment, I had to port the various tools used for converting graphics files and dealing with data files in general. Some I had created myself, and these ported almost without modification because I had used DJGPP as the C compiler. The only change required was related to the DOS custom of having CR/LF (carriage return/line feed) as the end of a line, rather than the UNIX-style LF only. DJGPP header files define the flag **O_BINARY** which must be used when opening a file, to specify that CR/LF should not be converted. Under Linux, **O_BINARY** is not defined, so compiler errors would result. The solution was to place the three lines

```
#ifndef O_BINARY
#define O_BINARY 0
#endif
```

early in the source files, so the source would work without change under both DJGPP and gcc/Linux.

The main graphics manipulation tools had been written by someone else at the company, and I had to make more extensive changes to get them to compile under gcc/Linux. There was the **O_BINARY** problem as before, but in addition, some of the programs did wild-card expansion in the program itself. The DOS shell doesn't do wild-card expansion, so DOS programs must provide that service for themselves. Unfortunately, the functions for performing this were nonstandard C and so wouldn't carry over. In the end, I hacked out those sections of the code and relied on standard UNIX shell wild-card expansion to do the job. Also, the header file “windows.h” was included frequently, and that dependence had to be removed as well as the structures and system calls in the code that required it.

Another problem which came up is the DOS/Win32 standard practice of file names being case insensitive. Under UNIX, case is significant, so errors popped up in source files where a file such as “Elmer.h” was being requested, when the file in the directory was actually elmer.h (or even worse, ELMER.H). Also throughout the code, programs would create an output file with an extension and the extension would be in upper case, so I'd generate files with names such as “cpaused.CHR”. To me, this was jarring and unattractive, so I modified all the files to produce lower case extensions. This required another round of changes when the wrong file name was being referred to, and error messages were appearing.

In the end, I managed to mirror on Linux every tool we had under DOS/Win32. Developing code under Linux was vastly more enjoyable than under DOS/Win32, mostly because it was faster and more stable. There is also something very rewarding about using your own tools in any work. In total, from start to finish, the time to bring everything up on Linux, as well as writing the new emulator, took about three to four weeks.

Shortly after the next project began taking form, one of the artists asked if he could do builds himself, as he wanted to tinker with animation frames and see how they looked in actual use. Rather than go to the trouble of getting all the tools set up on his machine, as a quick solution I copied the source tree to a directory path that my machine had been making public with Samba. The directory /d on my machine appeared as //dave/d on the network. I set up a simple script to check for the existence of the ROM image file. If the file wasn't there, it would do a rebuild. So, to get a new ROM image, the artist would copy over the modified data files, then delete the ROM image and wait a moment for

the new build to appear magically. With Linux's excellent disk caching, I was usually unaware of when this happened.

One final problem caused me some worry. I was currently the only person actually using Linux as a development platform. What would happen if one of the other programmers wanted to contribute to the project? I didn't want to force my Linux preference onto anyone else, so I had to be able to deal with this. Already, all our tools had equivalent Linux/DOS versions, so that wasn't a problem. The problem would be from multiple programmers merging code changes to the same source tree. Under Win32, we had been using Microsoft Visual Source Safe for that. Although I suspected there must be a Linux client for talking to Source Safe, I never looked. Instead, I studied how to use **CVS** (Concurrent Versions System), the traditional source-code revision-control system used in a vast array of open-source projects. After experimenting with CVS and learning enough about its quirks, I found it performed at least as well as Source Safe. It conveniently handles the end-of-line problem by storing the source files in the repository in UNIX format and adding/removing the CR as needed when dealing with a Win32/DOS client. I'm using CVS on the command line, and I prefer that to mouse clicks; CVS seems faster at updating the modified files.

Even though I'm the only programmer on the current project, I am checking my code changes into the network CVS server, just to get into the habit of doing so, as well as to provide an additional level of backup and modification history. As you might have guessed, the CVS server is running on the Linux box.

Related URLs

email: dash@xdr.com

David Ashley (dash@xdr.com) has been working with computers and electronics for over 24 years. He has written several free games for Linux, including Scavenger, XBomber and Sdlshanghai. He pays his bills by working at Left Field Productions, Inc. of Westlake Village, California, doing game programming for Nintendo consoles. Having recently become a father, he is finding himself with less and less time for his addiction—programming in Linux. He tells us, “All opinions expressed in this article are my own and do not represent the opinions or official policy of Left Field Productions, Inc.”

Archive Index Issue Table of Contents

Advanced search

Getting the NT Out—And the Linux In

David C. Smith

Issue #75, July 2000

An overview of configuring Linux using Samba to replace the services provided from Windows NT servers.

You have probably been hearing many rumors lately about Linux in the corporate environment. I've been hearing “Linux is great, but it's not ready for production” and “I wouldn't trust my business to Linux.” Lately, with all the press Linux has been getting, it's time to set the record straight. Being a longtime UNIX user, I jumped on the Linux bandwagon several years ago. I have used Linux in a production environment and know plenty of people who are doing the same.

There are many web, mail and database servers currently used in production systems, with more being added all the time. Linux success stories range from Linux being used at NASA, to being used for creating movie effects. So, is Linux ready for a prime time production environment? You bet! Is Linux ready to replace Windows NT Servers for your corporate LANs? Yep! I'll walk you through building a Linux server that is going to be more stable, faster, easier to maintain and costs less.

In setting up a Linux file and print server, you will find more configuration and customization than I will be using in this simple scenario. To learn more about the different options and configurations, see the Resources section at the end of this article.

SMB Background

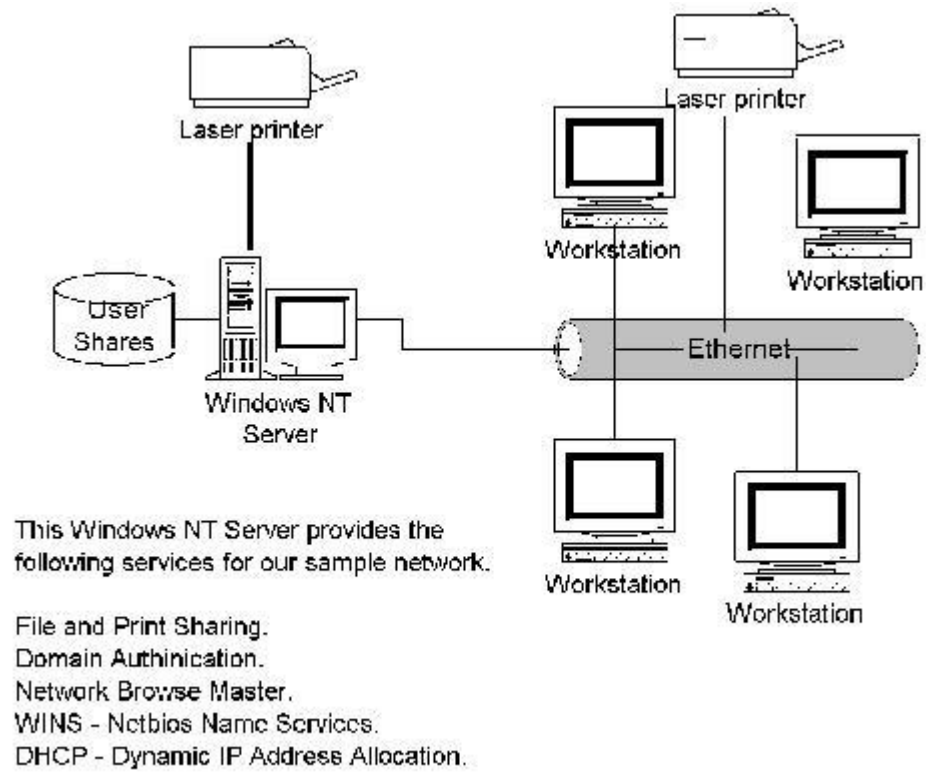
Windows machines use a protocol called Server Message Block (SMB) to perform file and print sharing as a network service. The SMB protocol defines how clients talk to servers to request printers, files, security validation and more. SMB has been around for a long time, and has some limitations that require a bit of thought. SMB requests and responses are based on local

broadcasts for a NetBIOS name, which is usually the server name. This presents a problem to (routed) environments in which routers separate networks, like the Internet, because broadcasts do not pass through routers. This created a need for translation from NetBIOS names to IP addresses. Microsoft implemented this solution as the Windows Internet Name Service (WINS).

SMB is also used for directory services. Most users think of the directory services as the “Network Neighborhood” feature on their desktops. It's a bit more than that, but enough to start. It's important to keep track of which machines are on a network and the services they provide. Nodes do this by electing a “Browse Master” that keeps track of which computers are on the network. When SMB machines boot, they broadcast their name and service information for all to hear. The elected browse master keeps a database of these names and will respond to requests from local machines. This browse master can be updated from other browse masters on different networks and can share its own information.

Windows NT Services

First, let's take a look at a sample Windows NT network and see what services are being provided (Figure 1). A Windows NT server has been configured as a file and print server. Users log in to the Windows NT server, using their Client for Microsoft Networks service with their network credentials. Once the user has been validated, a logon batch file is executed that assigns a user's home directory, various network drives and printers. The NT server also keeps track of which computers are on the network and the services they provide; clients can use this information in the Network Neighborhood.



The Linux Side of the House

Linux can use SMB to communicate with Windows and DOS-based clients using a package named Samba. The Samba suite was originally created by Andrew Tridgell, and is now developed by the Samba team. The Samba suite is currently running on somewhere around forty different platforms spanning the globe. Samba's main server dæmons are **smbd** and **nmbd**, which are pronounced “SMB-Dee” and “NMB-Dee”. **smbd** provides file, print and authentication services to Windows and DOS clients, and nmbd provides NetBIOS name resolution and browsing services (rfc1001/1002). Using these packages, Linux can easily provide the same services as our NT scenario.

Building a SAMBA Server

Get the samba-latest.tar.gz file from the SAMBA site and unpack it to a temporary directory using

```
tar -xvzf samba-latest.tar.gz
```

Change to this directory, and review the README files for any special information. After familiarizing yourself with the documentation, begin the install with the following commands:

```
cd source
 /configure
make
make install
```

Once the **make install** is complete, smbd and nmbd should be ready for configuration.

In reading the Samba documentation, you will find many different ways to configure smbd and nmbd. The Samba suite has extensive features that allow Linux to integrate and complement NT servers and services, but we are going to configure our Linux server to replace the NT server shown in Figure 1. Specifically, we are going to configure Samba to validate users and run our login batch file, provide file and print shares, and provide network-browsing services.

Begin by editing the smbd initialization file, smb.conf. By default, it is located in /usr/local/samba/lib/smb.conf, but is sometimes found at /etc/smb.conf. I would like to stress that there are many features which can be configured in the smb.conf file, and I am starting with only the basics.

Global Parameters

security = user is the default security setting for Samba 2.*x*. This configures Samba to require a user to provide authentication to access the server. To understand how Samba works with NT domains and servers, see "Security = Domain" in the Samba documentation.

iworkgroup = MyGroup controls which workgroup your server will appear to be in when queried by clients.

encrypt passwords = Yes controls whether encrypted passwords will be negotiated with the client. Windows NT 4SP3+ and Windows 98 will expect an encrypted password by default.

min passwd length = 6 sets the minimum length in characters of a plaintext password that smbd will accept when performing UNIX password changing.

smb passwd file = /etc/smbpasswd sets the path to the encrypted smbpasswd file. By default, the path to the smbpasswd file is compiled into Samba. I always add this to reduce confusion.

logon script = STARTUP.BAT specifies the batch file (.bat) or NT command file (.cmd) to be downloaded and run on a machine when a user successfully logs in. The file must contain the DOS-style cr/lf (carriage return/line feed) line endings.

If **domain logons = Yes** is set to true, the Samba server will serve Windows 95/98 domain logons for the workgroup it is in. For more details on setting up this feature, see the file DOMAINS.txt in the Samba documentation.

domain master = Yes enables WAN-wide (wide area network) browse list collation. Setting this option causes nmbd to claim a special domain-specific NetBIOS name that identifies it as a domain master browser for its given workgroup. Local master browsers in the same workgroup on broadcast-isolated subnets will give this nmbd their local browse lists, and will then ask smbd for a complete copy of the browse list for the entire WAN. Browser clients will then contact their local master browser and will receive the domain-wide browse list, instead of just the list for their broadcast-isolated subnet.

preferred master = Yes is a Boolean parameter which controls whether nmbd is a preferred master browser for its workgroup.

Setting Up Network Shares

That's it for our global parameters. We can now move on to creating network shares. By setting up a **[homes]** section, our server can create home-directory mappings on the fly:

```
[homes]
comment = Home Directories
read only = No
create mask = 0750
browseable = No
```

Now let's create some shares for users to access. The share definition should include the path, who can access the share (valid or invalid) and whether the share is writeable. By default, if no valid user or group is defined, the share is open to any client, so keep this in mind when creating your shares. In the apps share, I chose to create the UNIX group **all_users** containing just my local users.

```
[apps]
comment = Apps Directory
path = /shares/apps
valid users = @all_users
read only = No
[project1]
comment = Project 1 Directory
path = /shares/proj1
valid users = dcsmith kholmes joe katie redpup
read only = No
```

Last, I set up my netlogon home. This will be set to the relative path for my netlogon scripts. In this example, my login script is located at /etc/netlogon/STARTUP.BAT.

```
[netlogon]
path = /etc/netlogon
```

The full Samba configuration script is shown in Listing 1.

Listing 1

Samba Dæmons

The next step is to start the Samba dæmons. After checking everything out, you will probably want to add this to your system startup procedures.

```
/usr/local/samba/bin/smbd -D -s
/usr/local/samba/lib/smb.conf
/usr/local/samba/bin/nmbd -D
```

Troubleshooting

If everything went well, both smbd and nmbd were started successfully. If not, start troubleshooting by reading the log files at /var/adm/logs and review the FAQs from the Samba site.

Troubleshooting utilities, located in the Samba bin directory, are **testparm**, which will parse your smb.conf for errors, **smbstatus**, and **nmblookup** for NetBIOS name issues.

Setting Up the Samba Password File

Now it's time to add your users and passwords to your smbpassword file. One item to note is that users must have a UNIX account password as well. There are many options regarding passwords, such as remote password sync and NT domain and pass-through authentication, to help you with larger administration issues. In our case, user accounts are on our local Linux box. This command will create a SMB account and then prompt you to change your password.

```
/usr/local/samba/bin/smbpasswd -a testuser
```

You should now be able to log in as testuser and get authenticated from your Windows machines and access network shares. Great fun, eh? Once you get up and running, you will want to use some of the tools and utilities that Samba provides. One of the more useful utilities available is SWAT, a web-based administration tool that helps monitor and configure almost all Samba configurations. If SWAT is not available on your system, you can find it and more on the Samba home page.

Wrapping it Up

Hopefully, I provided you with enough information and inspiration to build Linux file and print servers. While I am not recommending that you dash out and replace your production NT servers, give Linux servers a chance. I'll bet you'll find them more stable and reliable, and they make remote administration much easier. The Samba team is constantly making Samba a better product with more and better features and utilities. As Linux solutions become more

and more of a reality, I believe you will find that Linux file and print servers are an efficient, cost-saving tool—that will make both you and your department budget happy.

Resources

email: dcsmith@duderman.com

David Smith (dcsmith@duderman.com) lives in Springfield, VA. He works at TimeBridge Technologies, where he manages customer networks as the Engineering Manager. When not working, he is either at a baseball game or waiting for the baseball season to start.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Linux System Calls

Moshe Bar

Issue #75, July 2000

How to use the mechanism provided by the IA32 architecture for handling system calls.

This article aims to give the reader, either a kernel novice or a seasoned programmer, a better understanding of the dynamics of system calls in Linux. Wherever code sections are mentioned, I refer to the 2.3.52 (soon to be 2.4) series of kernels unless otherwise noted.

The Linux Kernel in Brief

The most widespread CPU architecture is the IA32, a.k.a. x86, which is the architecture of the 386, 486, the Pentiums I, Pro, II and III, AMD's competing K6 and Athlon lines, plus CPUs from others such as VIA/Cyrix and Integrated Device Technologies. Because it is the most widespread, it will be taken as the illustrative example here. First, I will cover the mechanisms provided by the IA32 type of CPU for handling system calls, and then show how Linux uses those mechanisms. To review a few broad terms:

- A **kernel** is the operating system software running in protected mode and having access to the hardware's privileged registers. The kernel is not a separate process running on the system. It is the guts of the operating system, which controls the scheduling of processes to achieve multitasking, and provides a set of routines, constantly in memory, to which every user-space process has access.
- Some operating systems employ a **microkernel architecture**, wherein device drivers and other code are loaded and executed on demand and are not necessarily always in memory.
- A **monolithic architecture** is more common among UNIX implementations; it is the design employed by classic designs such as BSD.

The Linux kernel is mostly a monolithic kernel: i.e., all device drivers are part of the kernel proper. Unlike BSD, a Linux kernel's device drivers can be “loadable”, i.e., they can be loaded and unloaded from memory through user commands.

Basically, multitasking is accomplished in this way: the kernel switches control between processes rapidly, using the clock interrupt (and other means) to trigger a switch from one process to another. When a hardware device issues an interrupt, the interrupt handler is found within the kernel. When a process takes an action that requires it to wait for results, the kernel steps in and puts the process into an appropriate sleeping or waiting state and schedules another process in its place.

Besides multitasking, the kernel also contains the routines which implement the interface between user programs and hardware devices, virtual memory, file management and many other aspects of the system.

Kernel routines to achieve all of the above can be called from user-space code in a number of ways. One direct method to utilize the kernel is for a process to execute a system call. There are 116 system calls; documentation for these can be found in the man pages.

System Calls and Event Classes on the IA32

A system call is a request by a running task to the kernel to provide some sort of service on its behalf. In general, the kernel services invoked by system calls comprise an abstraction layer between hardware and user-space programs, allowing a programmer to implement an operating environment without having to tailor his program(s) too specifically to one single brand or precise specific combination of system hardware components. System calls also serve this generalization function across programming languages; e.g., the **read** system call will read data from a file descriptor. To the programmer, this looks like another C function, but in actuality, the code for read is contained within the kernel.

The IA32 CPU recognizes two classes of events needing special processor attention: interrupts and exceptions. Both cause a forced context switch to a new procedure or task.

Interrupts

Interrupts can occur at unexpected times during the execution of a program and are used to respond to signals; they are signals that processor attention is needed from hardware. When a hardware device issues an interrupt, the interrupt handler is found within the kernel. Next month, we will discuss interrupts in more detail.

Two sources of interrupts are recognized by the IA32: maskable interrupts, for which vectors are determined by the hardware, and non-maskable interrupts (NMI Interrupts, or NMIs).

Exceptions

Exceptions are either processor-detected or issued (thrown) from software. When a procedure or method encounters an abnormal condition (an exception condition) it can't handle, it may throw an exception. Exceptions of either type are caught by handler routines (**_exception handlers_**) positioned along the thread's procedure or method invocation stack. This may be the calling procedure or method, or if that doesn't include code to handle the exception condition, its calling procedure or method and so on. If one of the threads of your program throws an exception that isn't caught by any procedure (or method), then that thread will expire.

An exception tells a calling procedure that an abnormal (though not necessarily rare) condition has occurred, e.g., a method was invoked with an invalid argument. When you throw an exception, you are performing a kind of structured “go to” from the place in your program where the abnormal condition was detected to a place where it can be handled. Exception handlers should be stationed at program-module levels in accordance with how general a range of errors each is capable of handling in such a way that as few exception handlers as possible will cover as wide a variety of exceptions as are going to be encountered in field application of your programs.

An Example of Exceptions as Objects from Java

In Java, exceptions are objects. In addition to throwing objects whose class is declared in java.lang, you can throw objects of your own design. To create your own class of throwable objects, you need to declare it as a subclass of some member of the **Throwable** family. In general, however, the throwable classes you define should extend class **Exception**--they should be “exceptions”. Usually, the class of the exception object indicates the type of abnormal condition encountered. For example, if a thrown exception object has class **illegalArgumentException**, that indicates someone passed an illegal argument to a method.

When you throw an exception, you instantiate and throw an object whose class, declared in java.lang, descends from **Throwable**, which has two direct subclasses: **Exception** and **Error**. Errors (members of the **Error** family) are usually thrown for more serious problems, such as **OutOfMemoryError**, that may not be easy to handle. Errors are usually thrown by the methods of the Java API or the Java Virtual Machine. In general, code you write should throw only exceptions, not errors.

The Java Virtual Machine uses the class of the exception object to decide which catch clause, if any, should be allowed to handle the exception. The catch clause can also get information on the abnormal condition by querying the exception object directly for information you embedded in it during instantiation (before throwing it). The Exception class allows you to specify a detailed message as a string that can be retrieved by invoking **getMessage** on the exception object.

Vectors

Each IA32 interrupt or exception has a number, which is referred to in the IA32 literature as its *vector*. The NMI interrupt and the processor-detected exceptions have been assigned vectors in the range 0 through 31, inclusive. The vectors for maskable interrupts are determined by the hardware. External interrupt controllers put the vector on the bus during the interrupt-acknowledge cycle. Any vector in the range 32 through 255, inclusive, can be used for maskable interrupts or programmed exceptions.

The **startup_32** code found in /usr/src/linux/boot/head.S starts everything off at boot time by calling **setup_idt**. This routine sets up an IDT (Interrupt Descriptor Table) with 256 entries, each four bytes long, total 1024 bytes, offsets 0-255. It should be noted that the IDT contains vectors to both interrupt handlers and exception handlers, so “IDT” is something of a misnomer, but that's the way it is.

No interrupt entry points are actually loaded by startup_32, as that is done only after paging has been enabled and the kernel has been relocated to 0xC000000. At times, mostly during boot, the kernel must be loaded into certain addresses, because the underlying BIOS architecture demands it. After control is passed to the kernel exclusively, the Linux kernel can put itself wherever it wants. Usually this is very high up in memory, but below the 2GB limit.

When **start_kernel** (found in /usr/src/linux/init/main.c) is called, it invokes **trap_init** (found in /usr/src/linux/kernel/traps.c). **trap_init** sets up the IDT via the macro **set_trap_gate** (found in /usr/include/asm/system.h) and initializes the interrupt descriptor table as shown in the “Offset Descriptionis” table.

Offset Descriptions

Table 1

At this point, the interrupt vector for the system calls is not set up. It is initialized by **sched_init** (found in /usr/src/linux/kernel/sched.c). To set interrupt 0x80 to be a vector to the **_system_call** entry point, call:

```
set_system_gate (0x80, &system_call)
```

The priority of simultaneously seen interrupts and exceptions is shown in the sidebar “Runtime Priority of Interrupts”.

Runtime Priority of Interrupts

The System Call Interface

The Linux system call interface is vectored through a stub in libc (often glibc) and is exclusively “register-parametered”, i.e., the stack is not used for parameter passing. Each call within the libc library is generally a syscall*X* macro, where *X* is the number of parameters used by the actual routine. Under Linux, the execution of a system call is invoked by a maskable interrupt or exception class transfer (e.g., “throwing” an exception object), caused by the instruction in 0x80. Vector 0x80 is used to transfer control to the kernel. This interrupt vector is initialized during system startup, along with other important vectors such as the system clock vector. On the assembly level (in user space), it looks like Listing 1. Nowadays, this code is contained in the glibc2.1 library. 0x80 is hardcoded into both Linux and glibc, to be the system call number which transfers control to the kernel. At bootup, the kernel has set up the IDT vector 0x80 to be a “call gate” (see arch/i386/kernel/traps.c:trap_init):

Listing 1

```
set_system_gate(SYSCALL_VECTOR,&system_call)
```

The vector layout is defined in include/asm-i386/hw_irq.h.

Not until the int **$0x80** is executed does the call transfer to the kernel entry point _system_call. This entry point is the same for all system calls. It is responsible for saving all registers, checking to make sure a valid system call was invoked, then ultimately transferring control to the actual system call code via the offsets in the **_sys_call_table**. It is also responsible for calling **_ret_from_sys_call** when the system call has been completed, but before returning to user space.

Actual code for the system_call entry point can be found in /usr/src/linux/kernel/sys_call.S and the code for many of the system calls can be found in /usr/src/linux/kernel/sys.c. Code for the rest is distributed throughout the source files. Some system calls, like **fork**, have their own source file (e.g., kernel/fork.c).

The next instruction the CPU executes after the int **$0x80** is the **pushl %eax** in **entry.S:system_call**. There, we first save all user-space registers, then we range-check **%eax** and call **sys_call_table[%eax]**, which is the actual system call.

Since the system call interface is exclusively register-parametered, six parameters at most can be used with a single system call. **%eax** is the syscall number; **%ebx**, **%ecx**, **%edx**, **%esi**, **%edi** and **%ebp** are the six generic registers used as param0-5; and **%esp** cannot be used because it's overwritten by the kernel when it enters ring 0 (i.e., kernel mode).

In case more parameters are needed, some structure can be placed wherever you want within your address space and pointed to from a register (not the instruction pointer, nor the stack pointer; the kernel-space functions use the stack for parameters and local variables). This case is extremely rare, though; most system calls have either no parameters or only one.

Once the system call returns, we check one or more status flags in the process structure; the exact number will depend on the system call. **creat** might leave a dozen flags (existing, created, locked, etc.), whereas a **sync** might return only one.

If no work is pending, we restore user-space registers and return to user space via **iret**. The next instruction after the **iret** is the user-space **popl %ebx** instruction shown in Listing 1.

More Complex System Calls

Some system calls are more complex then others because of variable-length argument lists. Examples of a complex system call include **open** and **ioctl**. However, even complex system calls must use the same entry point; they just have more overhead for parameter setup. Each syscall macro expands to an assembly routine which sets up the calling stack frame and calls _system_call through an interrupt, via the instruction int $0x80. For example, the setuid system call is coded as

```
_syscall1(int,setuid,uid_t,uid)
```

which expands to the assembly code shown in Listing 2.

Listing 2

The User-Space System Call Code Library

The user-space call code library can be found in /usr/src/libc/syscall. The hard-coding of the parameter layout and actual system call numbers is not a

problem, because system calls are never really changed; they are only "introduced" and "obsoleted". An obsoleted system call is marked with the **old_** prefix in the system call table for entry.S, and reference to it is removed from the next glibc. Once no application uses that system call anymore, its slot is marked "unused" and is potentially reusable for a newly introduced system call.

Tracing System Calls

If a user wishes to trace a program, it is equally important to know what happens during system calls. Thus, the trace of a program usually includes a trace through the system calls as well. This is done through **SIGSTOP** and **SIGCHLD** ping-ponging between parent (tracing process) and child (traced process). When a traced process is executed, every system call is preceded by a **sys_ptrace** call. This makes the traced process send a SIGCHILD to the tracing process each time a system call is made. The traced process immediately enters the **TASK_STOPPED** state (a flag is set in the task_struct structure). The tracing process can then examine the entire address space of the traced process through the use of **_ptrace**, which is a multi-purpose system call. The tracing process sends a SIGSTOP to allow execution again.

How to Add Your Own System Calls

Adding your own system calls is actually quite easy. Follow this list of steps to do so. Remember, if you do not make these system calls available on all the machines you want your program to run on, the result will be non-portable code.

- Create a directory under the /usr/src/linux/ directory to hold your code.
- Put any include files in /usr/include/sys/ and /usr/include/linux/.
- Add the relocatable module produced by the link of your new kernel code to the **ARCHIVES** and the subdirectory to the **SUBDIRS** lines of the top-level Makefile. See fs/Makefile, target fs.o for an example.
- Add a **#define __NR_*xx*** to unistd.h to assign a call number for your system call, where *xx*, the index, is something descriptive relating to your system call. It will be used to set up the vector through sys_call_table to invoke your code.
- Add an entry point for your system call to the sys_call_table in sys.h. It should match the index (*xx*) you assigned in the previous step.
- The NR_syscalls variable will be recalculated automatically.
- Modify any kernel code in kernel/fs/mm/, etc. to take into account the environment needed to support your new code.
- Run **make** from the top source code directory level to produce the new kernel incorporating your new code.

At this point, you must either add a syscall to your libraries, or use the proper **_syscalln** macro in your user program in order for your programs to access the new system call. The *386DX Microprocessor Programmer's Reference Manual* is a helpful reference, as is James Turley's *Advanced 80386 Programming Techniques*.

Linux/IA32 Kernel System Calls

A list of Linux/IA32 kernel system calls can be found, with the listings, in the archive file ftp.linuxjournal.com/pub/lj/listings/issue75/4048.tgz. Note: these are not libc “user-space system calls”, but real kernel system calls provided by the Linux kernel. Information source is GNU libc project, http://www.gnu.org/.

email: moshe@moelabs.com

Moshe Bar (moshe@moelabs.com) is an Israeli system administrator and OS researcher who started learning UNIX on a PDP-11 with AT&T UNIX Release 6 back in 1981. He holds an M.Sc. in computer science. His new book Linux Kernel Internals will be published this year. You may visit Moshe's web site at http://www.moelabs.com/.

Archive Index Issue Table of Contents

Advanced search

Voice Recognition Ready for Consumer Devices

Linley Gwennap

Issue #75, July 2000

This looks like the year that voice recognition finally reaches the mainstream.

This looks like the year that voice recognition finally reaches the mainstream. Motorola unveiled “Mya, the 24-hour talking Internet” at the Oscars. Tellme.com and other startups are deploying voice portals that accept speech commands and read web content over a standard telephone. The latest Jaguar allows drivers to adjust the climate and sound systems using their voice.

Most of these services run on remote servers or PCs where plenty of processing power is available. But the Jaguar example is telling: CPU performance has reached the point that even an inexpensive embedded processor can perform useful voice recognition. Over the next few years, voice will become a common interface in a variety of non-PC devices, many of which will be running Linux.

Until recently, voice recognition required each user to train the system to recognize his or her particular speech patterns. Like most other software, however, voice recognition improves given faster processors and more memory. Recent products reduce training time dramatically. Speaker-independent software eliminates training entirely. To achieve highly accurate speaker-independent recognition with moderate processing requirements, designers must limit the context and vocabulary of the application. For example, a car needs to recognize only a few dozen words, including “temperature”, “radio”, and the numbers needed to select a station.

Lernout & Hauspie (http://www.lhsl.com/), a leading supplier of voice software, supplies speech engines for applications as simple as these, as well as far more complex ones. According to Klaus Schleicher, a director of product management at L&H, the simplest speech engine provides speaker-independent recognition of up to 100 words, but requires less than 200K of memory. L&H offers a more-powerful speech engine that can recognize up to

1,000 words, again without training. This engine requires 2MB of memory and can run on a 200MHz processor. This hardware costs a bit more, but is still easily obtainable for $30 today, and that price will drop over time. The larger vocabulary is suitable for applications such as a TV set-top box that can be programmed by speaking the name of a show or a hand-held PDA that can manage calendars and address books via voice.

Composing arbitrary text, such as an e-mail message, requires a much larger vocabulary. For this purpose, L&H has a speech engine with a 20,000-word vocabulary—twice as large as the average adult's. This engine requires some training, but only about five minutes per user. Even this large vocabulary doesn't require a full-blown PC or server; the company has demonstrated it using a 200MHz StrongArm processor and 32MB of memory. This speech engine could be incorporated into a webpad, allowing users to compose e-mail and other documents without using a keyboard.

One problem is that these speech engines are still not 100% reliable. The smaller the vocabulary, the smaller the error rate—after all, there are fewer words to confuse. In addition, a “command and control” application has natural opportunities to seek clarification. For example, if the user says “Turn off the TV” in a noisy room, the system might respond “I didn't understand that; please try again” or “Do you want the TV off?” In these limited-domain applications, the software actually interprets the voice input to determine its meaning, in this case, to turn off the TV. One possible interpretation of the input phonemes might be “turnips are meaty”, but the software would quickly discard this possibility as irrelevant in the context of controlling the television. This intelligent interpretation is called natural language processing (NLP). The combination of good voice recognition and a well-programmed NLP back end can produce a reliable system.

A working example is MIT's Jupiter system, a conversation interface for weather information built by the university's Spoken Language Systems group. You can call it (1-888-573-8255, but it is often busy) and ask about the weather anywhere in the U.S. or around the world. It uses a 500MHz Pentium III PC running Linux, but it hasn't been optimized to reduce CPU overhead. Jupiter has a vocabulary of about 2,000 words and is very usable. Text dictation, however, has a much larger vocabulary and an unbounded content domain: an e-mail message could have any subject matter, even turnips. NLP for this application is much harder and generally limited to putting nouns and verbs in the right places. After dictating a few hundred words into even the best speech engine, a user is likely to have to go back and correct at least a dozen errors.

Thus, for applications where a keyboard is available and the user can type reasonably well, typing is likely to be the most efficient interface for the

foreseeable future. But L&H's Schleicher says, “the human voice is the most natural user interface for communication and computing on a variety of devices.” For command and control applications in cars, information appliances, set-top boxes and even PCs, voice recognition is an excellent interface. The hardware just needs the right programming—and the sound of your voice.

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

Archive Index Issue Table of Contents

Advanced search

An Appetite for Discovery

Marcel Gagné

Issue #75, July 2000

Looking at the skies for stars and aliens can both be done on Linux systems.

Ah, it is a beautiful night, non? Welcome, mes amis. Sit down and François will get you a glass of wine. I must tell you, mes amis, that this month's special issue on Science and Engineering has your chef feeling rather pensive this evening.

Some say that science has become rather dry lately. I find this idea difficult to understand as we stand on the threshold of unraveling the human genetic code, as new extra-solar planets are discovered on an almost daily basis, as human spacecraft orbit distant asteroids or gather cometary dust. If science has become boring, then I would suggest it is the presentation of science that has become dry. It's time to put some excitement back into the study of life, the universe and everything else for that matter. You are running Linux on your computer, non? Then it is time for you, mes amis, to join the search for knowledge.

Linux's scalable nature and multi-user UNIX roots make it an ideal platform for scientific work. Now everyone can have a workstation for data collection and analysis, mathematical and statistical modeling, and magnifique visualizations and simulations. Even better, Linux's open source means that scientists and engineers don't have to wait for some company to create the tools they need. Add to that Linux's network-ready and network friendly design, and you have the best environment not only for research, but for collaboration with your peers. Scientists have been using UNIX machines for years precisely for these reasons. With Linux, things have gotten even better. When the going gets tough and more horsepower is needed than can be generated by a single machine, Linux is still the answer with low-cost Beowulf clusters.

No longer do we, the general public, have to watch science take place in the shadowy temples of distant laboratories. With the power of Linux, everyone can be a participant. Proof of Linux's prowess in the scientific world is as easy

to find as a trip to the **SAL** web site (Scientific Applications on Linux) at sal.kachinatech.com/index.shtml.

For instance, you could start a collection of meteorological data for your area. Using a program like **Kweather**, you can monitor daily weather events in your area such as temperature highs and lows or precipitation. Kweather lets you graph the results so you can track patterns over time. Are we truly experiencing a warming trend? Memory is fragile, as I am often reminded when I comment on the 14-foot-high snowbanks of my childhood. With Kweather and disciplined observation, you can know for sure. For your copy of Kweather, pay a visit to J<\#252>rgen Hochwald's web site at www.privat.kkf.net/~juergen.hochwald/linux/kweather/e_index.html.

But why, mes amis, would we want to concern ourselves merely with the world around us when an infinite number of *other worlds* seek to capture our curious imaginations, yet another click of the mouse away? Besides being an aficionado of wine, good food and heavenly sauces, your humble chef is also an amateur astronomer. Ah, the mysteries of the universe! François! More wine, please.

And so, the next item on our menu comes to us courtesy of Aaron Worley, who brings us **Hitchhiker 2000** (HH2000). This wonderful program is a solar system simulator, a digital orrery (Marcel's Collins defines it as a "mechanical model of the solar system in which the planets can be moved at the correct relative velocities around the sun"). Aaron calls his site "The Hitchhiker's Guide to the Solar System".

HH2000 is educational, surely; but mostly, the program is *just plain fun*. The idea is that you have a camera mounted on your celestial body of choice, any planet or any moon in the solar system. You can choose comets and asteroids as well. HH2000 comes with a healthy database of objects and the means to add more by way of .CSV-format files (comma-separated values). The effects are great, too. Depending on your angle of view, which you can change by dragging your mouse around the view screen, you'll notice the planets' night side is realistically in shadow. Using the center mouse button (with my two-button mouse, the right button did the job), you can zoom in or out by dragging the mouse pointer up or down. With a single click, you can flip the viewing from local orbit to deep space to an orbital view.

This eye on the sky uses **OpenGL** or **Mesa** libraries (I used Mesa) for 3-D effects, both of which can be a little resource-hungry at times. For maximum effect, you might want something punchier than my 150MHz Pentium notebook. Still, even under this environment, I found Hitchhiker to be plenty acceptable, which brings us to requirements. HH2000 is built on GNOME libraries, so you'll either need to be running GNOME or have the libraries loaded. If you are like François

and me, you probably have both desktop environments loaded already. Before you can start exploring, you will likely need one other piece of software (aside from OpenGL or Mesa). Download the **gtkglarea** libraries as well, and compile them. You will find links to all these pieces on the Hitchhiker 2000 web site (see Resources). Finally, pick up the HH2000 code. Binaries for glibc2.0 systems are available, as is source code. With my Red Hat 6.2 system, I compiled the program from source. The steps are simple:

```
tar -xvfz hh2000-0.3-0.tar.gz
cd hh2000-0.3-0
 ./configure
make
make install
```

Running the program is done by typing **hh2000**. Now, sit back and enjoy the ride. Careful on that gas pedal.

For the truly serious astronomer, Elwood Downey brings us our next item, and what a masterpiece this is. **XEphem** is a star-charting package that pretty much does it all. You start by identifying your location (in my case, Toronto, Ontario) and clicking “Update”. XEphem loads the appropriate latitude and longitude information for your chosen city. If you need to be more accurate than “next door is okay”, it is possible to enter that information manually as well. Want to see what portion of the moon will be visible April 30, 2007? You can change the date through the calendar interface, click “Update” once again, then select “View” and “Moon”. The major planets are available at a click, as is a solar system view, which can be animated and its angle of viewing changed.

Every program like XEphem needs a star chart. For many amateur astronomers like myself, the planets are there as a warmup to the real meat of observing, namely the stars and deep sky objects. XEphem comes through with a fantastic star chart that allows you to define many viewing options. For instance, you can activate constellation lines, labels, define the types of objects you would like to see (galaxies, open clusters, double stars, etc.) or the minimum display brightness for these objects. XEphem also lets you center in on an object and zoom in with a simple slide control.

Exploring with XEphem is almost too much fun (as your humble chef discovered while trying to finish this article). Mais, qu'est-ce que c'est? That open cluster just below Cepheus looks interesting. Would you like to see what it looks like through a very powerful telescope? Center the object, click on “View Image” and let the Space Telescope Science Institute and the Hubble Space Telescope give you a close-up! Quite honestly, mes amis, the features are just too numerous to mention. Here's another bonus. Next time your friends who run that other operating system tell you about all the great software you can't run on Linux,

show them XEphem. Then watch their faces when you tell them they can't have it for the other OS.

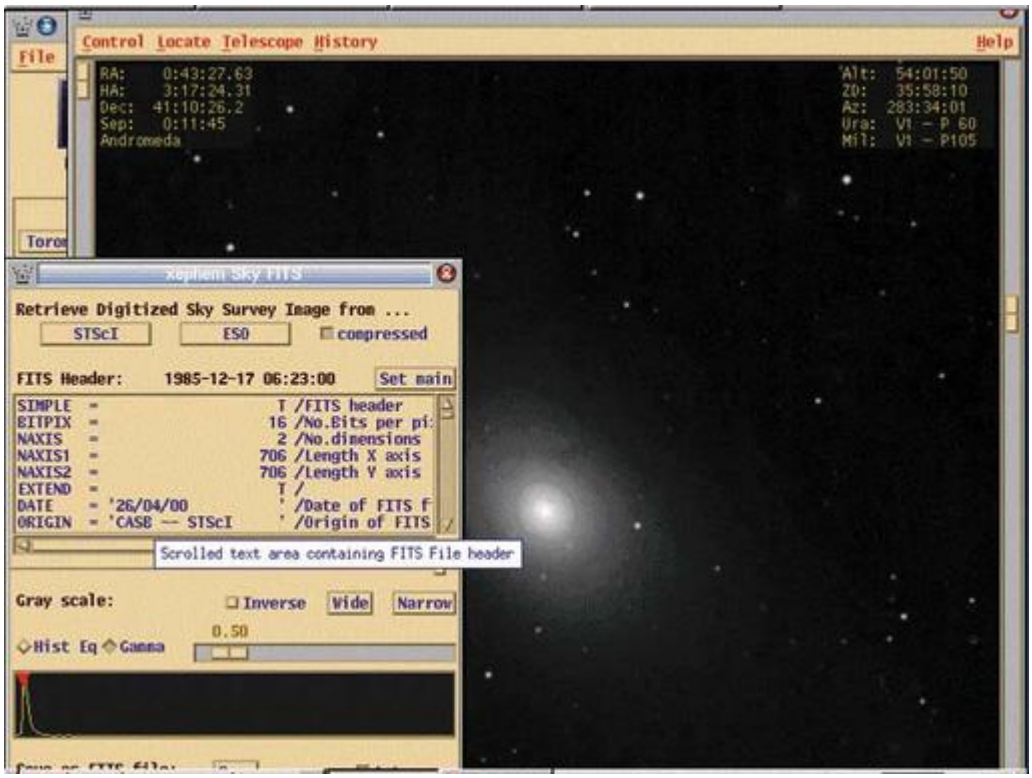


Figure 1. M31—A XEphem Celestial Close-up

XEphem is a commercial product, but it is also available as a free download. I will tell you that the download version is still fairly spectacular. The CD-ROM (commercial version), however, comes with much more, including the full Hubble Guide Star Catalog (not included in the download), a huge number of additional deep sky objects, a full printed manual, and pre-built binaries for various platforms. While you can connect to the Internet when you want additional information such as a close-up (as in the previous paragraph), you may find it well worth the price.

If you want to build XEphem yourself, you will also need either Motif or the freeware LessTif from the Hungry Programmers (do they know Chez Marcel is open, I wonder?). It is available at http://www.lesstif.org/ for download. Building LessTif is a matter of downloading lesstif-current.tar.gz from the web site and following these now-familiar steps:

```
tar -xvfz lesstif-current.tar.gz
cd lesstif-current
 ./configure
make
make install
```

Next, you need to download your copy of XEphem. At the time of this writing, the latest version was **xephem-3.2.3.tar.gz**. Building XEphem is a little different than the usual “configure” and “make”. Here are the steps:

```
tar -xzvf xephem-3.2.3.tar.gz
cd xephem-3.2.3/libastro
 ./xmkmf
make
cd ../GUI/xephem
 ./xmkmf
make
make install
```

Happy Stargazing

Finally, why not join in the greatest adventure of them all: SETI, the Search for Extraterrestrial Intelligence? Your Linux system is ideally suited to this task. Since you are running a true multi-user system, it is possible to have a **setiathome** process running in the background (reniced so it doesn't draw too heavily on your system resources). Who knows? You may be the one who decodes the first signal from a distant civilization, like California—I joke, seulement, non? Nevertheless, visit the SETI@home pages, register as a SETI explorer, download your client and do your part in exploring what may be humankind's most exciting new frontier. [You can join the *Linux Journal* Reader Group too.]

There is no compiling or linking to do. You simply download the client and untar it. For instance, I downloaded the 2.4 version of the client. After untarring the bundle, I renamed the directory (you will see why) and ran the client from there.

```
tar -xvf setiathome-2.4.i386-pc-linux-gnu-glibc2.1.tar
mv setiathome-2.4.i386-pc-linux-gnu-glibc2.1
setiathome
cd setiathome
 ./setiathome
```

You can also launch the program from a crontab (as I do) and just leave it running in the background. Here is my crontab entry:

```
0 * * * * cd /root/setiathome; ./setiathome\
-graphics<\n> -nice 19 > /dev/null 2> /dev/null
```

Currently, SETI@home runs with a text-only client, but the latest version also comes with an experimental “screensaver mode” program called **xsetiathome**. The **-graphics** option in the crontab above is required if you wish to use this experimental “xsetiathome” GUI front end. Even before this GUI feature was added, Linux SETI users were creating their own GUI clients to give setiathome a friendlier face. One of my favorites is still **TkSETI** from Rick Macdonald. You can download TkSETI from www.cuug.ab.ca/~macdonal/tkseti/tkseti.html. One of the things I like about it is the ability to check my progress against my other friends who run SETI@home. At this moment, Chef Marcel's lovely wife, Sally, is way ahead, but François is way behind.

Figure 2. Keeping Tabs on SETI@home with TkSETI

Oui, mes amis, it is that time again. I hope you enjoyed the items on today's menu and that you will find yourselves exploring other tasty avenues on your own. In the meantime, it is a clear night and Chef Marcel has the telescope set up out back. If you wish to join me, François will pour you a final glass of wine and we will savor the Chablis while we search the heavens. Join me again next time at *Chez Marcel*. Bon Appétit!

Resources

Marcel Gagné (mggagne@salmar.com) lives in Mississauga, Ontario. In real life, he is president of Salmar Consulting Inc., a systems integration and network consulting firm. He is also a pilot, writes science fiction and fantasy and edits TransVersions, a science fiction, fantasy and horror magazine. He loves Linux

and all flavors of UNIX and will even admit it in public. You can discover many things from his web site at http://www.salmar.com/.

Archive Index Issue Table of Contents

Advanced search

Press Releases with Mason

Reuven M. Lerner

Issue #75, July 2000

To learn more about building dynamic web sites, Mr. Lerner presents an application for reading the news using Mason and MySQL.

Last month, we took an initial look at Mason, a template system that sits on top of **mod_perl** and allows us to create fast-executing dynamic web sites built out of small components.

This month, we will look at a simple application built in Mason—a system to display the latest press releases on a corporate site. Of course, such a system could be tailored in a number of ways, including an on-line newspaper or other publication in which information changes on a regular basis. In creating this small site, we will see some of the steps involved in working with Mason.

Creating the Database

The core element of our news system will be a relational database. I will use MySQL in these examples, although any relational database system should work fine.

I created a new MySQL database called “atfnews” on my MySQL server and assigned privileges so that the user atfnews can connect using the password “atfpass”. I then created the following two tables:

```
CREATE TABLE Categories (
     category_id SMALLINT UNSIGNED AUTO_INCREMENT,
     category_name VARCHAR(25) NOT NULL,
     PRIMARY KEY(category_id),
     UNIQUE(category_name)
);
CREATE TABLE Articles (
     article_id MEDIUMINT UNSIGNED AUTO_INCREMENT,
     category_id SMALLINT UNSIGNED NOT NULL,
     posting_date TIMESTAMP NOT NULL,
     headline VARCHAR(30) NOT NULL,
     body TEXT NOT NULL,
     PRIMARY KEY(article_id),
```

```
        UNIQUE(category_id, headline)
);
```

As you can probably tell from their names, the Categories table contains a list of category ID numbers and names. The Articles table contains several more pieces of information, including an article ID, the category ID into which an article should be placed, the date and time at which the article was posted, the article's headline and its body. We ensure no two articles in a given category have the same headline with a UNIQUE clause at the end of our CREATE TABLE statement.

The posting_date column takes advantage of MySQL's **TIMESTAMP** data type. This type automatically inserts the time and date of the latest **INSERT** or **UPDATE** to a given row. In this way, we can easily determine when news stories were added to the database, without having to enter or keep track of the information ourselves.

In order for our news system to work, we will need to create at least two different sets of components. One set will allow users to enter news items into the database (i.e., perform INSERTs), and the second will make it possible to retrieve items from the database (i.e., perform SELECTs). In a production setting, we would probably want to restrict posting access to a selected number of users. This would be possible with a standard .htaccess file, which allows users to restrict access to individual files or directories, or with a more sophisticated system that stores user information in a database.

Structuring the Components

One of Mason's strong points is its use of components. Components are actually Perl subroutines, cleverly disguised in the form of HTML files with some Perl thrown in. (Mason's parser performs the underlying magic that turns components into subroutines.) This structure means that repeated functionality can be packaged into one component, then invoked from within other components.

Listing 1

For example, Listing 1 contains a component called “database-connect.comp”. This component returns a value, rather than producing HTML output. Its purpose is to connect to a database server and return a database handle, typically called **$dbh**. By centralizing this connection code, we can easily move our site from one server to another, changing only the relevant **$host**, **$user**, **$password** and **$database** variables as necessary.

Once database-connect.comp has been configured, any component on our system can receive a valid database handle with the following code:

```
<%init>
my $dbh = $m->comp(database-connect.comp);
</%init>
```

The above code takes advantage of Mason's object-oriented interface, using the predefined **$m** object to invoke another component.

By placing the assignment inside of **<%init>**, we ensure that the component will connect to the database before anything else occurs within the component. However, this also means we are creating a new lexical variable (**$dbh**) with each invocation of the component.

It would be slightly more elegant to perform the above assignment within a **<%once>** section, creating **$dbh** a single time and keeping the value around. However, **<%once>** sections are executed outside of the Mason component context, meaning they cannot invoke methods on **$m**. Moreover, **<%once>** sections are invoked before new Apache child processes are created, which a **$dbh** object might not like. Thus, it is common to define **$dbh** in a **<%once>** section, but to perform the assignment in **<%init>**:

```
<%once>
my $dbh;
</%once>
<%init>
$dbh = $m->comp(database-connect.comp);
</%init>
```

The plain-vanilla mason.pl (or “handler.pl”, as the Mason documentation describes it) configuration file that comes with the Mason distribution is almost good enough for this system to work. We need to load only **Apache::DBI**, a wrapper module that works with DBI within the mod_perl environment, ensuring that database connections are created and dropped only as necessary.

In order to load Apache::DBI, we need to put a **use Apache::DBI** statement in mason.pl, which is loaded with a **PerlRequire** statement in httpd.conf. In order to save some memory, we insert a **PerlModule Apache::DBI** line into httpd.conf. This ensures the module is loaded into memory before Apache splits into numerous child processes. The module might still require a fair amount of memory, but at least that memory will be shared among all Apache processes rather than requiring each one to have its own copy.

Adding Categories

The first step toward making our news system work is to allow users to create new categories. Each news story in our simple system will be placed in precisely one category, much as each story in a newspaper is printed in only one section.

If we were to use the CGI model for creating a database editor, we would need to create an HTML form, pointing its **<Form>** action to the URL of a CGI program. That CGI program would then need to retrieve the HTML form elements, connect to the database and perform an INSERT.

Listing 2

With Mason, all this becomes much easier because of the relationship between HTML form elements and variables. We will still need two different components, one that presents the form and another that acts on the form's contents. The first component, add-category-form.html (see Listing 2), is a normal HTML form, with a single text field and a “submit” button. The only difference between this form and its non-Mason counterpart is the **action** attribute in the **<Form>** tag. In Mason, even a file with an .html suffix is a program and can thus receive input from an HTML form.

Listing 3

The component that handles the input and inserts a new row into the Categories table is called add-category.html (see Listing 3). As is often the case with Mason components, you must first look at the component's final sections (**<%once>**, **<%init>** and **<%args>**) in order to understand what is happening.

In the case of add-category.html, our **<%once>** section merely defines **$dbh**, as described above. The **<%init>** section performs two actions. First, it defines **$dbh** based on the returned value from “database-connect.comp”. Once the database connection has been established, the **<%init>** section goes on to INSERT the user's input into the database. Notice how we use DBI's placeholders, shown here as a question mark in the list of **VALUES**, to avoid potential problems with quoted strings within our SQL query.

The placeholder is filled in with the value of **$new_category_name**, a scalar variable defined in **<%args>**:

```
<%args>
$new_category_name
</%args>
```

By defining it there, we indicate that add-category.html must receive an HTML form element **new_category_name** when it is invoked. We could have given **new_category_name** a default value; however, this value is crucial to the functioning of add-category.html and must be mandatory.

Depending on whether the SQL INSERT succeeds, the scalar variable **$successful_insert** is set to true or false. This value is then used in the large if-else statement, to produce HTML that reflects the success or failure of the

INSERT shown at the beginning of Listing 3. Notice how **$DBI::errstr**, the standard DBI error message, is available from within our component.

Adding News

Listing 4

Once we have added one or more categories, we can begin to insert news items into the system. Unlike add-category-form.html, add-news-form.html (Listing 4) will need to connect to the database and cannot be a simple HTML form. This is because we want to present the user with a **<select>** list of current categories. In order to create this list dynamically, we will need to connect to the database and perform a simple SELECT. Other than that, the HTML form is relatively straightforward. We will use a table to organize the titles and form elements, but it consists of three basic elements: a headline, the body text and a category **<select>** list.

I decided to do this in a relatively inefficient (but easy to understand) way, using an SQL **ORDER BY** clause to retrieve names in alphabetical order. In order to keep track of the two different values (ID and name), I put them into the **@categories** array:

```
while ($row_ref = $sth->fetchrow_arrayref)
{
     my ($id, $name) = @$row_ref;
     push @categories, {id => $id,
                        name => $name};
}
```

We can then iterate through **@categories**, placing the category ID as the “value” attribute (which will be submitted to the add-news.html component), but displaying the name of the category:

```
<select name="category_id">
% foreach my $category (@categories) {
<option value="<% $category->{id} %>">
<% $category->{name} %>
% }
</select>
```

Listing 5

The component that adds news, add-news.html (Listing 5), is almost identical to add-category.html, except it inserts three values rather than just one: the category ID, the headline and the body of the article. If the submission is successful, we tell the user that the article has now been placed in the database.

Retrieving News

While we could retrieve the news directly into a top-level component, it is easier for us to create a generic component that retrieves any number of articles from any category. In this way, we can use this “get-news.comp” component in a number of different high-level components, retrieving the number and type of articles that interest us.

Listing 6

Listing 6, get-news.comp, is fairly straightforward, returning a list of articles to the caller. It builds the article list much as we built the category list in add-news-form.html, retrieving each of the articles:

```
while ($row_ref = $sth->fetchrow_arrayref)
{
     my ($headline, $body, $posting_date) =
           @$row_ref;
     push @articles, {headline => $headline,
           body => $body,
           posting_date => $posting_date};
}
return @articles;
```

We take advantage of MySQL's **LIMIT** clause to restrict the retrieval to only as many articles as the user is interested in receiving. Also, we retrieve the articles in reverse order of their arrival, so that the article with the latest timestamp will come first. This ensures whenever we retrieve the latest five articles, they will indeed be the newest:

```
my $sql  = "SELECT headline, body, posting_date ";
   $sql .= "FROM Articles ";
   $sql .= "WHERE category_id = ?";
   $sql .= "ORDER BY posting_date DESC ";
   $sql .= "LIMIT ?";
```

Printing the News

get-news.comp returns the latest news into an array. But, of course, users are interested in reading news, not looking at a Perl array. For that reason, we'll define two more top-level components: one to choose the category and number of articles we wish to read and one to display them.

Listing 7

First, we will create a component view-stories-form.html (Listing 7), which lets us choose a category and maximum number of stories to display. This component repeats the paradigm of creating a **<select>** list from a Perl array. It then invokes view-stories.html (Listing 8), a simple component which does nothing but iterate through the stories returned by get-news.comp, placing them in a nicely formatted page of HTML.

Listing 8

Conclusion

As you can see, the amount of effort and code necessary to create this site was fairly modest. And while this is a relatively simple site, it does work—and it represents one way in which Mason and databases can be used together to create a dynamic site in a minimum amount of time. True, we ended up writing a number of components; but at least two of them are reusable if we decide to expand the site in the future, and thus will reduce the amount of work and debugging we'll have to do at that time.

With a bit more work, we could add personalization to this site, allowing users to read only news that is new to them and in only the categories that interest them.

As I indicated last month, Mason has increasingly become my tool of choice for producing these sorts of web sites, because of the speed and ease with which I can do so. The fact that I can separate tasks into reusable components and the high-speed gains from working within mod_perl are bonuses that make Mason an extremely attractive web-development environment.

4066s1

email: reuven@lerner.co.il

Reuven M. Lerner , an Internet and Web consultant, moved to Modi'in, Israel following his November marriage to Shira Friedman-Lerner. He has written a book, Core Perl published by Prentice-Hall. Reuven can be reached at reuven@lerner.co.il. The ATF home page, including archives, source code and discussion forums, may be found at http://www.lerner.co.il/atf/.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

Focus on Software

David A. Bandel

Issue #75, July 2000

AirTraffic, xscorch, tt and more.

By the time you read this column, several distributions will be offering the new 2.4.*x* Linux kernel. Much work has gone into it, and a new facet has been added: a /devfs file system. It will be interesting to see if distribution makers use this particular facet. I've seen more debate on this particular feature than on any other—the camps remain divided.

The arguments don't collide directly. The arguments for devfs run along these lines: the /dev file system is populated with hundreds of entries for all possible devices. The number and type of devices is about to explode with the use of USB and IR. Each entry requires one inode, which results in the use of 4096 bytes of disk space for each entry. Because these are node entries (i.e., device files), that's basically wasted space; the inode contains all required information. The arguments against devfs are primarily implementation-based—this isn't how it should be done, and it's kludgy. This argument could easily be used with many new implementations, which is why some kernel developers oppose its inclusion in the kernel: it needs more time to mature. I won't take sides, but I will tell you I've been using it since before mid-April and it's working the way it should, for me. If necessary, you can still create entries, but they should be created by the modules as they load. I expect many improvements will be made in the months ahead. I think this is an exciting idea even if the implementation needs work.

AirTraffic: http://airtraffic.sourceforge.net/

For those of you who remember a game called Air Traffic Controller or ATC, this game should bring back memories. What I can remember of ATC was that it was not easy to keep all those little planes that came on and off the screen too fast, apart. Well, I still can't. Guess my decision not to become a real air traffic controller was a good one—at least based on this game. It requires libgnomeui,

libart_lgpl, libgdk_imlib, libSM, libICE, libgtk, libgdk, libgmodule, libXext, libX11, libgnome, libgnomesupport, libesd, libaudiofile, libm, libdb, libglib, libdl, glibc and libz.

xscorch: http://chaos2.org/xscorch/

This particular game goes way back. I remember playing this one on a dumb DEC terminal connected to a mainframe somewhere; I just can't remember what it was called. Then came a qbasic clone that used monkeys throwing explosive bananas at each other—same game. Aim your sights on the mountains or valleys where the other cannons are, and try to blow them up before they get you. Between this game and AirTraffic above, I've obviously had too much fun this month (or too much spare time on my hands—not). It requires libgtk, libgdk, libgmodule, libglib, libdl, libXext, libX11, libm, libXpm and glibc.

tt: http://awacs.dhs.org/software/tt/

tt is a time tracker that is very simple to use. In fact, the easiest way to use it is as part of a shell script that turns it on and off. You define the projects you're working on, and tell tt to start timing that project. When you've finished working, tell tt to stop. The data from tt can be exported in several formats into a MySQL database, an ASCII file, etc. While not included, I'm sure a small button with project names could be put on your X desktop. Then you could just click a project on and off as you work on it. It requires Perl 5 and the following Perl modules: POSIX, Time::Local, Sys::Hostname and Fcntl.

phpSched: sourceforge.net/project/?group_id=3034

Do you have to keep track of employee schedules? Perhaps you have a dozen or so folks working for you who need schedules. **phpSched** might be able to help you. Once you set it up, employees can even request when they want to be on. You (or whoever controls the schedule) get final say. You can see who's scheduled next week, last week or even last month. Everything is saved in a MySQL database. It requires a web server with PHP and MySQL, and a frames-capable browser.

pmc: http://www.diablonet.net/~ishamael/

Perl Mail Client (pmc) is very basic, but gets the job done. It uses the mbox format, which most other mail clients don't seem to want to do. This habit of graphical e-mail clients emptying your mailbox can be annoying if you're temporarily stuck on a non-graphical terminal (and non-graphical clients look ugly in an xterm box on your pretty graphical screen). **pmc** gets around the

problem. It's not elegant, but it is functional. It requires Perl 5 and the following Perl modules: Gtk and Net::SMTP.

unicount: www.terahertz.net/~macabre/files/unicount-1.3.4.tar.gz

For web sites, there are counters and there are counters. I've seen gaudy counters, broken counters, graphics-only counters (usually on Lynx-unfriendly, completely graphical pages) and more. This counter is different. It can be graphical if you want, or text. It's easily changed in a config file. The text is unobtrusive. I put a sample on the home page on my system, and I hardly noticed it. And it works. It does require that the server parse the page, but I'm not convinced that's a particular problem. It requires glibc and a web server that parses HTML.

blackbook: http://blackbook.sourceforge.net/

This address book is not fancy, just a place for names and numbers and some text where you can put an address. It keeps a file in your home directory with the entries, so those of you who don't like SQL databases can use it. The data file is readable by only the application, and no import or export button is available yet. It requires libgtk, libgdk, libgmodule, libglib, libdl, libXext, libX11, libstdc++, libm and glibc.

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

Archive Index Issue Table of Contents

Advanced search

Embedded Systems News Briefs

Rick Lehrbaum

Issue #75, July 2000

The latest new from Embedded Linux Consortium, Protectix, Inc., ZDNet and more.

GoAhead Software announced a “high availability” (HA) Linux solution for telecommunications and Internet infrastructure equipment. The goal is the reliability “holy grail” of HA: 99.999% (“5NINES”) up time. (www.linuxdevices.com/news/NS8938608255)

The **Embedded Linux Consortium** (ELC) unveiled its shiny new web site at http://www.embedded-linux.org/. See which companies are members, become a member and learn more about the organization. (www.linuxdevices.com/news/NS5228213053)

Protectix, Inc. and **Lynx Real-Time Systems** are collaborating on Linux network security enhancements. Protectix had previously developed ApplianceOS, a security-focused embedded Linux that fits within 20MB of Flash memory. (www.linuxdevices.com/news/NS7042363749)

ZDNet announced the acquisition of LinuxDevices.com. This embedded Linux portal site is being incorporated “as is” into ZDNet's Linux Resource Center. (www.linuxdevices.com/news/NS7268795633)

DIPARM (Italy) released version 1.3 of RTAI, the popular open-source real-time Linux add-on. The new version adds dynamic memory allocation, /proc interface, enhanced LXRT-Informed module and Perl bindings to simplify “soft real-time” programming. (www.linuxdevices.com/news/NS7090394754)

QNX Software Systems is making its QNX real-time OS more “Linux-like”. QNX development software is now free to developers, as is source to most QNX drivers and utilities. Also, a new Linux compatibility layer lets you run Linux

binaries on QNX. QNX kernel source remains secret. (www.linuxdevices.com/news/NS3407445841)

Before building that Linux supercomputer, visit www.PileofPCs.org/—the new web site devoted to proliferating cluster-based supercomputing. (www.linuxdevices.com/news/NS7489101114)

Adomo Inc. says it is developing a Linux-based server, thin-client terminals, information applications and web-based services for the home. The products will provide fast and easy access to information by all family members from anywhere in the home. (www.linuxdevices.com/news/NS3866190958)

Century Software unveiled a duo of technologies that capture data from large systems, translate it into XML format, then display it on embedded devices with limited display capabilities. (www.linuxdevices.com/news/NS2915863199)

The list of post-PC processors supported by Linux lengthened, as Rt-Control released the latest version of **uClinux**. **uClinux** now supports ARM7TDMI and i960 in addition to Motorola 68K. (www.linuxdevices.com/news/NS8108603067)

10,000? That's how many Linux-based Webplayer Internet Appliances will be shipped by **Virginconnect**, according to Merinta—the company that supplies the Internet Appliance software suite used within the device. (www.linuxdevices.com/news/NS8068168802)

K-Team (Switzerland) is releasing GPL "Video for Linux" (V4L) drivers for video capture in robot vision applications. K-Team builds miniature mobile robots. Robotic penguins? (www.linuxdevices.com/news/NS6964926004)

MINIX author Andy Tanenbaum reports release of MINIX under terms equivalent to the BSD license. Andy claims MINIX is much smaller than Linux, and might be a suitable OS for tiny embedded systems like watches, cameras or transistor radios. (www.linuxdevices.com/news/NS2755841532)

MontaVista Software announced Hard Hat Linux for the IBM PowerPC 405GP. The 405GP is a 200MHz 32-bit RISC-based "system-on-chip" with built-in 100MBps Ethernet, serial/parallel ports, memory controllers and more. (www.linuxdevices.com/news/NS2088063517)

Software tools vendor **VioSoft** announced Linux support for NEC's VR4121, VR5432 and VR5000 64-bit MIPS RISC system-on-chip processors. The VR4141, together with the VRC4171A companion chip, implement all the functions of a typical hand-held PC. (www.linuxdevices.com/news/NS2020182962)

Get a free design package for an MC68EZ328-based SIMM-sized SBC that runs uClinux. To manufacture it in quantity, you'll need a license. (www.linuxdevices.com/news/NS7598641214)

Applied Data Systems unveiled a 4x6-inch single-board computer for Linux-based embedded control and graphical user interface. The board contains an onboard StrongARM CPU, plus a hearty complement of I/O. (www.linuxdevices.com/news/NS7736362772)

Coollogic launched the e-Pilot 7000, a Linux-based Internet appliance for ISPs and Vertical Market Integrators. e-Pilot comes with Coollinux, Coollogic's embedded Linux OS. (www.linuxdevices.com/news/NS3600550089)

A new open-source programmable logic controller (PLC) project plans to create PLCs based on Linux and its real-time variants. (www.linuxdevices.com/news/NS8228680582)

Axis Communications (in Sweden) announced an open-source Journaling Flash File System (JFFS) that provides a crash/powerdown-safe file system for Linux-based diskless embedded devices. (www.linuxdevices.com/news/NS3252080240)

ISDCorp joined the MIPS Alliance Program and plans to port its Royal Linux distribution to MIPS32 and other MIPS RISC processors. (www.linuxdevices.com/news/NS3859329386)

KURT (KU Real-Time Linux) is a real-time version of Linux with real-time event resolutions in the tens of microseconds. It's from the University of Kansas. (www.linuxdevices.com/news/NS9486308756)

A new Linux-based entertainment console from **Indrema** integrates the new Gecko browser, 3-D games, MP3 storage and playback, personal TV, HDTV and Internet video. Expect it in stores by the holiday season. (www.linuxdevices.com/news/NS4185499125)

M-Systems reports that the DiskOnChip flash disk is designed into a new Universal Internet Box from 3iLinux. The 3iLinux device contains a 386 CPU, 8MB RAM, 8MB DiskOnChip, 2x16 LCD, pushbuttons and a modem or LAN connection. (www.linuxdevices.com/news/NS6480449991)

Rick Lehrbaum (rick@linuxdevices.com) founded his second startup in October 1999: LinuxDevices.com—"the Embedded Linux Portal". Rick received his BS and MS degrees in physics from NYU and Northeast Louisiana University, respectively.

Letters

Various

Issue #75, July 2000

Readers sound off.

Corrections and Apologies *I was a bit surprised by the response we got to our sendup of the "Monty Python" nude organist routine. This column is dedicated to those responses. Those for which there was no room can be found in "More Letters" on the LJ web site. I was particularly amazed that some people took this as "sexual" and therefore obscene. The Post Office and our newsstand distributors all cleared this cover as non-offensive. Our only intent was to have a bit of fun, and we are sorry to have offended anyone. By the way, guys, what do you think of the May 1 cover of VARBusiness?*

Python Supplement Responses

I loved the Python supplement to *LJ*. We should do something like this again soon. I just got an e-mail from Mike O'Dell, senior VP and chief scientist at UUNET, who told me, "your article restores my hope for humanity" and "this almost makes me feel good enough to write a program!"

I mentioned this to Andrew Kuchling, and he mentioned some negative feedback you've received. I wonder what that was about? The python.org web site received a flame from someone offended by the naked man on the cover, and in the edu-sig, there was a discussion (*not* a flamewar!) of gender stereotypes, but in general I've seen mostly positive responses and encouraging words.

—Guido van Rossumguido@python.org

I just want to congratulate you on the excellent Python supplement included with the May edition of the best Linux magazine around!

I am not a professional programmer, but after reading the excellent articles included in said supplement, I was buzzing to get down to some serious programming using Python! I do not know what, but there was something in the way the articles written that just excited me!

Everything that comes out of the Linux community seems to always be bursting with energy and excitement. I looove it!! Keep up the excellent work!

—Rui Pinhor.pinho@ic.ac.uk

I've been very pleased with my subscription to *Linux Journal* over the past three-plus years, but I was particularly unimpressed with the cover of the Special Supplement included with the May 2000 issue.

Not only did I find it unprofessional and distasteful, but it was also somewhat embarrassing to receive a magazine at my office with male nudity on the cover.

I hope you'll accept this as constructive criticism and will carefully consider future covers. There are many other tasteful alternatives that would have better served the reputation of your otherwise fine publication.

—Doug Ledbetterdougl@mnu.edu

I just got my new *Linux Journal*, and was shocked at what I found on the cover of the Special Supplement. I hope the *Linux Journal* isn't going to start doing things for shock value. Please don't. (The secretary got a thrill out of it, though.) I actually took a marker and drew in some clothes.

—Walter Williamswwilliams@mountain-cad.com

What were you thinking displaying a picture of a naked person on the cover of your supplement? A lot of people (like me) receive this magazine at work. It's hard enough for me to get people in this company to take Linux seriously (especially with the ridiculous penguin logo). Now I have to get funny looks and explain to people (including my *boss*) why I get magazines with pictures of naked men on them sent to me at work.

It's a poor attempt at humor. Some of us have to work in a professional environment, and this damn near cost me my job.

—Phil Garrettpgarrett@hillmgt.com

Nice cover on the May 2000 issue! Love those penguinistas raining down on Redmond. And keep those Python articles coming!

—Darryldarryl@igor.penguinpowered.com

The cover is repulsive and offensive. It is very inappropriate in the business environment. If you wanted to get our attention, you did! We will not renew our subscription and I sure don't understand the logic behind this.

—Ed Hutchinsonehutch@raex.com

Hi there. I don't currently subscribe to *Linux Journal*, but I do use computers and the UNIX OS quite a bit. Anyway, I went to your web site and saw the cover of the Python supplement. I just wanted to say, good for you, guys! Geeks have long been bombarded with images of naked women, often anatomically impossible images at that, and it's about time we had some naked men for nerd girls to look at. :)

—Jina Chanjina@speakeasy.org

Must say, this is the first issue of *Linux Journal* I've ever read, despite being a Linux fan and having Tux propaganda in my cubicle at work. It was really nice to see the running Python gag throughout ... technical journals can be so dull! Thank you for having a sense of humor, at least once in a while.

As for the cover ... if hot-rod magazines can have barely clad women on the cover, I say “Yay!” to cute naked men on the cover of computer journals! About time. If more computer journals had naked guys in them, I might read more of them. ;) Oh wait, I'm supposed to say, “I only read it for the articles.” Yeah, that's it. But really, the cover is very nicely done ... dare I say tastefully silly with a large dash of cute?

—Lady Lilithlilith@serv.net

Thanks for all of your good work.

I'm a student assistant at SVC Mt.Vernon in electronics—we've been doing a pilot Linux project since summer. Part of my assisting deals with cookbook programming for hardware types.

We'd like to order 12 reprints of the Python add-on from this month's issue. Thanks.

—Kip Johnsonkiptech7@netscape.net

I'm no homophobe or fundamentalist wacko, but the supplement cover is too much! The first thing I saw was some naked guy, and I really thought it was someone's gay porn magazine put in my box by mistake.

Seeing that the readership of the magazine is probably 90% male, I would have preferred to see a naked female or at least one in a thong bikini. :)

I'm not sure what to do with the supplement, other than put a sticker on it so that others, who see it lying on my desk at work, don't also think it's a gay porn magazine.

You crazy Linux guys!

Great magazine by the way, keep up the good work. And remember—90% male readership.

—Neal Richternrichter@bridgernet.com

This is regarding your answers to letters published in the May 2000 *Linux Journal* defending the BSD Daemon babes. In your response to Jordan Hubbard, you stated regarding your past attendance at oil industry conventions:

“I was offended that women were being used as sex objects and that men were considered stupid enough to fall for such tactics.”

I am *offended* by people so full of themselves and their elitist, high-minded social agenda that they actually believe that those poor slobs at trade shows need someone to be offended *for* them! I looked at the pictures Jason posted on the web. What the hell is the problem? Get over yourself!

Oh, but that's not all. When one of the actual models writes in support of her role, you scold her for actually having *fun* at the show:

“Perhaps when you start thinking of yourself as a woman instead of a girl, you will understand...”

But wait, there's more! What's on the cover of the Special Supplement included with the issue? A completely naked man sitting in a field. Oops, he's wearing a bowtie. The hypocrisy and irony here is stifling.

Am I offended? Of course not! Worried about what my postal carrier might think, but not offended; for me, for him or for anyone, because I have a *life*.

We have a God-given and constitutionally protected right to free speech in the U.S., though it's dwindling daily, a luxury some of the readers of your magazine don't enjoy. Well, along with free speech goes the right to offend and be offended, if you so choose. Isn't America great? Go burn a bra or something...

Does this, like, totally squash my chances of ever getting published in *LJ*?

—Keith Brownbahalana@wt.net

I find the cover of the May 2000 Special Supplement to be in excruciatingly poor taste, bordering on the obscene. It is not what I expect of a semi-technical publication.

Cancel my subscription and refund the balance of the current subscription payment.

—Stephen P. Molnar, Ph.D.smolnar@jadeinc.com

I must say that what arrived in my mail with the latest issue of your magazine was quite a shock! To see a naked person on the front of the mailing is very offensive to say the least. I have a young daughter, and I wouldn't care for her to see such a sight. Can you imagine if I took a copy of this to my workplace and left it in plain view on my desk? I would be looking for another job, because of sexual harassment! If your magazine must resort to these methods to sell copies, then I will choose not to renew my subscription, and I will move to a more professional publication.

—Wesley Harmonagharmon@bellsouth.net

It's a bit difficult to take your shrill comments on sexism seriously, since you are the editor-in-chief of a magazine that just published an issue with a nude man on the cover.

—David Rodvolddrodvold@xaim.com

Loved the Python Special Issue cover. Keep up the good work.

—Stephen Bachseb2t@jm.acs.virginia.edu

Archive Index Issue Table of Contents

Advanced search

upFRONT

Various

Issue #75, July 2000

Stop the Presses, *LJ* Index and more.

THEY SAID IT

“Information wants to be $6.95.”

—Don Marti, VA Linux Systems

“Dot-coms are falling all around us like the frog plague at the end of the movie *Magnolia*.”

—Richard Thieme, in an “Islands in the Clickstream” essay

“For a list of the ways which technology has failed to improve our quality of life, press 3.”

—Phil Reed, on Slashdot

“People never grow up, they only learn how to act in public.”

—Tina Kimbley, Mudrealms.com

“Men are just teenage boys with credit cards.”

—Cindy Crawford

THE BUZZ

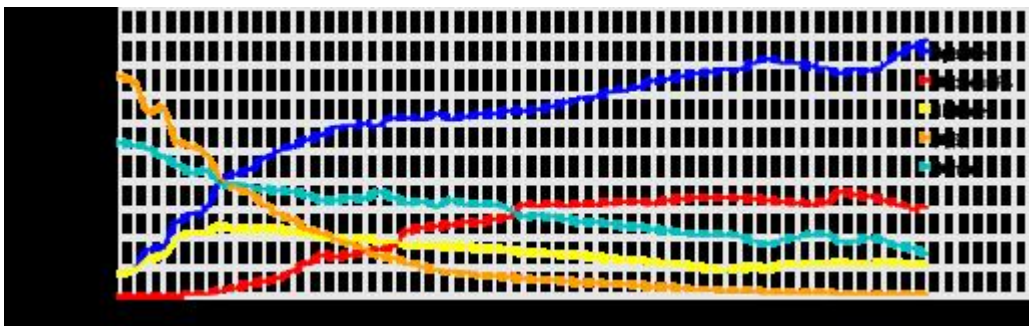
What were Linux people talking about in April and early May? Below is a sampling of some of the hotter news stories over the past few weeks, as reported in “The Rookery”, *Linux Journal's* on-line source for news, notes,

quotes and reports from the field (updated daily and found at our web site, http://www.linuxjournal.com/):

- Yopy's palm-sized Linux device set to debut this summer. Developers wanted!
- A $17 million Linux supercomputer being used at NOAA's Forecast Systems Lab to improve weather forecasting.
- Applixware's Linux division spinning off into its own company, VistaSource.
- Linuxcare canceling its IPO plans.
- Lineo *accepting* $37 million in funding.
- New York and Northern Virginia geeks (including ESR) protest outside the Library of Congress. Their target: the infamous Digital Millennium Copyright Act.

APACHE SURGE

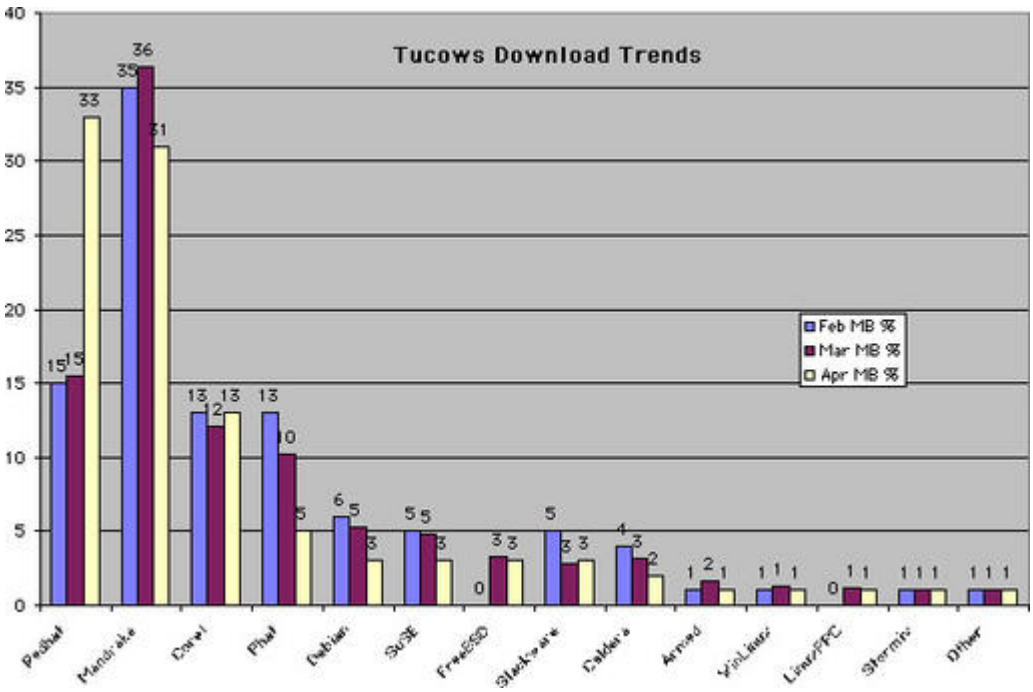
After years of steady climbing, Apache's share of the web server market peaked, according to Netcraft. Through the second half of 1999, its shares slipped. So did Microsoft IIS, but as the end of the year came around, Microsoft suddenly surged up and Apache began to reciprocate in the downward direction.



But that was for just one month. Since the first of the millennium, Apache has been steadily going up to reach record shares, and IIS has mostly gone down, relatively speaking (although in absolute numbers, it has been going up). In April, Apache tallied 8,812,960 web servers for a 1.48% increase to 61.53%. Microsoft was second, with 1,047,890 servers and a .16 percent increase to 21.09% of the total. Third was iPlanet, the family of Sun Solaris and Netscape servers sold by Sun. iPlanet servers may trail the leaders, but not among the top-traffic sites. “The Solaris/Netscape combination does particularly well amongst high-transaction SSL sites such as the leading retail brokerages, Charles Schwab, E*Trade, and Fidelity,” Netcraft says.

TUCOWS DOWNLOADS

After two months at the number-one position in Tucows downloads (measured in MB), Mandrake yielded the lead to Red Hat which more than doubled, from 15 to 33% of the total share, edging out Mandrake's 31%. Number three Corel returned to its February level at 13%. Number four Phat Linux lost half its share in one month, from 10% to 5%. Debian also continued to drop, from 6% in February to 3% in April. SUSE, FreeBSD and Slackware were each tied and holding about even at 3%. Caldera continued dropping, to just 2%. Everybody else held even at 1% or less. (Note: While February and March were full months, April was tabulated through the first 25 days.)



NATURAL-LANGUAGE MARKET

The natural-language market is under attack by several competitors. The most successful of these so far is “Ask Jeeves” (ASKJ), Ask.com. Ask Jeeves is a proprietary, closed-source natural-language understanding program for the Web. ASKJ attracted much attention in 1999 with its wildly successful IPO. In addition to AskJ, there are a number of smaller, pre-IPO companies entering the highly valued natural-language market. Forecasters see fantastic growth in applications such as intelligent customer service agents, web-based help desks and customer support.

In October 1999, AskJ and Microsoft announced a partnership to provide natural-language-based help desk support for Windows 2000. ASKJ is to ALICE what MS is to Linux. Although the markets are much smaller, the stage is set for a classic open-source/closed-source battle.

Ask Jeeves has been a magnet for lawsuits. In 1999, the company was sued by MIT professors Boris Katz and Patrick Winston, among others, who claim they have the patent on web-based natural-language transactions. The ALICE project has been operating “under the radar” in stealth mode, under the GPL.

ALICE

I am the leader of a group of Linux-like contributors to a gigantic open-source software project. Our goal is the creation of artificial intelligence using the natural language program ALICE. I developed AIML (Artificial Intelligence Markup Language) along with a crude interpreter, and released both under the GNU GPL. The result was the now-predictable, open-source magic: a team of developers from around the world began improving the code, creating content and providing an immense user base.

ALICE won the Loebner Prize, an annual “Turing Test”, in 2000, for being the computer program ranked closest to a human. The ALICE and AIML developers have created a library of content, numerous on-line documentation sites and versions of the ALICE server in both Java and C/C++. AIML is “platform-independent, language-independent” in the sense that ALICE robot scripts run on different interpreters on different operating systems. There are interfaces for HTTP, CGI, IRC, plain text, GUIs and even voice I/O (input/output).

There are at present four companies in various stages of starting up around ALICE and AIML technology. The developers have moved on to embedded system applications, intelligent customer service agents and entertainment applications of the technology. We are trying to stay “above the fray” and “not pick winners and losers”, as this technology begins to attract the attention of “suits and other real-world investors.” Does this story sound familiar?

—Dr. Richard Wallace, dr.wallace@mindspring.com

STRICTLY ON-LINE, http://www.linuxjournal.com/

Low-Bandwidth Communication Tools for Science, by Enrique Canessa and Clement Onime, tells us how in Trieste they are building prototype on-line scientific tools to further enhance electronic collaboration and support the use of web navigation and database search by e-mail. ScientificTalk and www4mail are two such tools that are based on Linux and discussed here.

AIPS: A Historical Reminiscence by Patrick P. Murphy takes a look at the astronomical image processing system and how it is being used on Linux by the National Radio Astronomy Headquarters and astronomers the world over.

Four book reviews to help you decide if these books are worthwhile:

- *Linux Administration, A Beginner's Guide*, review by Harvey Friedman.
- *Red Hat Linux 6 for Small Business*, review by Paul Dunne.
- *Security Technologies for the World Wide Web*, review by Wael A. Hassan.
- *Getting Started in Computer Consulting*, review by Ralph Krause.

CONFIGURING APACHE FOR WIRELESS BROWSERS

The use of cellular phones and other wireless devices has been rising exponentially. Recently, many of these devices have contained Internet-enabled functionality and even web-browsing software. If you run a popular Internet site, it's possible someone has tried to visit it with such a device and seen nothing! In this column, I'll show you how to configure the Apache web server to handle these requests successfully.

WAP/WML

The leading Internet implementations on hand-held devices have used the Wireless Application Protocol, WAP. The idea comes from the wireless industry and is based on existing Internet technologies such as IP. Just as I use HTML for my site, mikal.org, I'll now use WML for people visiting my site over WAP.

WML stands for Wireless Markup Language and is based on XML. Similar to HTML, WML is read and interpreted by a browser built into a WAP-enabled device.

Apache WML Setup

The first thing I need to do to handle wireless visitors is inform Apache about the MIME type I'll be using. I add MIME support for the WML file extension to the default MIME type configuration file in this way:

```
text/vnd.wap.wml      wml
```

This file includes a definition of the most commonly known MIME types. On my file system, it's located in the /etc/mime.types directory.

Catching Wireless Visitors

I want to catch anyone visiting my web site with a wireless browser and send them to my WML page, welcome_wap_user.wml (listing below). For this purpose, I'll use Apache's powerful **mod_rewrite** module, available in version 1.2 or later. By using this, I can rewrite requested URLs on the fly based upon rule conditions. It's possible that mod_rewrite isn't already compiled into the

server; check the Apache documentation for instructions on doing this. Specifically, I'll be looking at the **HTTP_USER_AGENT** and **HTTP_ACCEPT** environment variables to check for known WAP browsers.

```
<?xml version="1.0"?>
<!DOCTYPE wml PUBLIC "-//WAPFORUM//DTD WML 1.1//EN"
     "http://www.wapforum.org/DTD/wml1_1.1.xml">
<wml>
     <card>
     <p> Welcome to mikal.org! </p>
     </card>
</wml>
```

The mod_rewrite module can be used by placing the appropriate directives, shown below, in the httpd.conf Apache configuration file. Line 1 turns on the **RewriteEngine**. Jorrit Waalboer, from the WML programming list at eGroups.com, provided me with the **RewriteCond** statements in lines 2-7 to determine if the client is a WAP browser. If RewriteCond matches one of these browsers, **RewriteRule** tells Apache to serve welcome_wap_user.wml.

```
RewriteEngine on
# Catch most WAP browsers
RewriteCond %{HTTP_ACCEPT} text/vnd\.wap\.wml [OR]
# WinWAP, WAPjag
RewriteCond %{HTTP_USER_AGENT} wap [OR]
# Nokia emulators (sdk)
RewriteCond %{HTTP_USER_AGENT} 7110
# Rewrite!!
RewriteRule ^[\./](.*)$  /welcome_wap_user.wml [L]
```

To activate these changes, I'll need to restart Apache.

I can now check my site, but I'll need a WAP-compatible device. For this purpose, I use the UP.Simulator, part of UP.SDK, the Phone.com software development kit. The UP.SDK supports development of WAP services written in WML. After a quick check with the UP.Simulator, I see my site is now WAP-aware and ready for the future of wireless Internet.

For additional information on WAP, WML or building a WAP service, see Resources.

—Philip Mikal (philip_mikal@yahoo.com) is an Internet technology consultant based in Silicon Valley. He can be reached at mikal.org.

Resources

http://updev.phone.com/

http://wap.colorline.no/wap-faq/

http://www.wapforum.org/

http://www.waplinks.com/

http://www.egroups.com/group/wmlprogramming/

EVENTS

- USENIX; June 19-23, 2000; San Diego, CA; http://www.usenix.org/events/usenix2000/
- LinuxFest; June 20-24, 2000; Kansas City, KS: http://www.linuxfest.com/
- PC Expo; June 27-29, 2000; New York, NY; http://www.pcexpo.com/
- LinuxConference; June 27-28, 2000; Zürich, Switzerland; http://www.linux-conference.ch/
- Libre Software Meeting #1, July 5-9, 2000; Bordeaux, France, www.abul.org/rmll1-fr.html and rmll1-uk.html
- Summer COMDEX; July 12-14, 2000; Toronto, Canada; http://www.zdevents.com/comdex/.
- O'Reilly/2000 Open Source Software Convention; July 17-20, 2000; Monterey, CA; conferences.oreilly.com/convention2000.html

LJ INDEX—JULY 2000

1. Linux share of web servers in the domains .td (Chad), .ne (Niger), .lr (Liberia), .gq (Equatorial Guinea), .cf (Central African Republic) and .dj (Djibouti): **100%**
2. Total number of Linux web servers in the .td, .ne, .lr, .gp, .cf and .dj domains: **32**
3. Linux share of web servers in the .gg (Guernsey, Alderney and Sark) domain: **67.6%**
4. Total number of Linux web servers in the .gg domain: **97**
5. Linux share of web servers in the .md (Republic of Moldova) domain: **67.5%**
6. Total number of Linux web servers in the .md domain: **564**
7. Linux share of web servers in the .ro (Romania) domain: **59.7%**
8. Total number of Linux web servers in the .ro domain: **1,645**
9. Linux share of web servers in the .de (Germany) domain: **42.7%**
10. Total number of Linux web servers in the .de domain: **197,670**
11. Linux share of web servers in the .ru (Russian Federation) domain: **15.1%**
12. Total number of Linux web servers in the .ru domain: **3,498**
13. BSD family share of web servers in the .ru domain: **52.6%**
14. Total number of BSD web servers in the .ru domain: **12,211**
15. Total number of Google users early in its development: **10,000**

16. Total number of current Google users: **10,000,000**
17. Number of new domains registered during a 10-day period in March, 2000: **1,000,000**
18. Registration rate of new domains, per second, during the same period: **1**
19. Number of gallons of fresh water required to produce one pat of butter: **100**
20. Number of gallons of fresh water required to produce a chicken egg: **120**
21. Number of gallons required to produce a loaf of bread: **300**
22. Number of gallons required to produce a pound of beef: **3500**

Sources

- 1-14: The Internet Operating System Counter (www.leb.net/hzo/ioscount)
- 15-16: New Media
- 17-18: Netcraft
- 19-22: David Siegel

STOP THE PRESSES: Big Money Moves into Embedded Linux

On May Day (May 1, 2000), a point when financial markets seemed to have lost faith in Linux as a "Big Trend" (many Linux stocks lost most of their value in the first third of the year), a big chunk of change—$37 million, to be exact—was invested in Lineo, Inc., which is emerging as the leading embedded Linux software company.

In an interesting twist, the list of sources for that money includes only three venture capital firms. Another fourteen investors are actual or potential Lineo customers, including familiar names like Motorola, Samsung, Mitsubishi, Compaq, Citrix and Acer—plus a raft of motherboard, laptop and component manufacturers in Taiwan, Japan and Korea. These include DaiShin Information and Communications, First International Computer, Global Alliance, Hikari Tsushin, Arima and Mitac International. The VCs are Egan Managed Capital, J&W Seligman and Astoria Capital Partners.

These manufacturers are *players*. "There is a substantial interest on the part of major manufacturers in embedded Linux. And we include in that category a variety of software, hardware, components and solutions for embedded systems. These are smart companies that got to where they are by knowing how both to predict market trends and sense what's happening right now," says Lyle Ball, Vice President of Communications and co-founder of Lineo.

The most interesting aspect of this news is that it appears to be something unusual: a very traditional "Old Economy" play. These manufacturers want to

put Linux in their products, not just score a big run-up off a Lineo IPO (which, of course, they certainly wouldn't mind).

To get the significance of this, consider the little-discussed fact that every company has two markets: one for its goods and services and another for itself. Before the New Economy showed up, the latter market was extremely secondary, even for publicly traded companies. Value was *all*. Growth mattered, but there was no prevailing imperative to take a new company public or to run its value up to the sky overnight. But the get-big-quick imperative of the New Economy led to biased business conversations over the last several years, so that talk about investment has drowned out talk about the fundamentals of business. And this kind of talk has been endemic to the commercial Linux market ever since Red Hat went public last August and instantly branded Linux as the hot "growth topic" of 1999.

But this investment appears to be operating on the Old Economy imperative, which is to support product and service innovation. At least, this is what a long and manufacturer-heavy list of investors suggests.

What this also suggests is that Linux will probably expand from servers to appliances and other embedded devices more quickly than it will spread to clients—although there is no shortage of desktop and laptop manufacturers on this list of investors.

As Lineo sees it, the embedded Linux market shapes up this way:

Internet Infrastructure

- routers
- modems
- switches
- gateways
- convergence devices

Retail Business Systems

- credit-card readers
- point-of-sale systems
- hand-held scanners

Transportation Systems

- automotive systems

- radar controls
- global positioning systems
- anti-lock brakes and car infrastructure

Home Automation

- Internet set-top boxes
- smart appliances
- security
- environmental controls (thermostats, light and water irrigation systems)

Consumer Devices

- PDAs
- mobile phones
- entertainment systems
- scanners
- printers

Industrial Controls

- software controllers
- process automation
- manufacturing equipment
- industrial (e.g., assembly-line) automation

“These are categories where we already have customers or expect to have them as demand spreads,” Ball says. He expects demand to spread quickly because Linux's advantages lower the threshold of adoption well below competing operating systems, including familiar embedded operating systems such as Wind River, ISI and QNX. These virtues include open OS code source, an in-place worldwide support infrastructure, a huge developer community and base of knowledge, and relatively easy code adaptation across different processor types. “Linux is the Switzerland of Internet connectivity and infrastructure,” Ball adds. “You're not making NetWare Internet-savvy here. You're taking something that's extremely native to the Net and adapting it to a whole class of new Net-native devices. And you're doing it with the support of many thousands of Linux developers, all over the world, vs. relatively few working for proprietary embedded OS companies. That's why our community is bypassing the development schedules that have determined embedded growth in the past, and accelerating a whole new schedule. We're putting the embedded Linux growth schedule on *Internet time*.”

Linus Torvalds seems to agree. He has been talking up embedded Linux for a while now, and his employer, Transmeta, has more or less the same intentions as Lineo.

“Now you can imagine a Coke machine as a Net-native connected device,” Ball says. This might have been thinkable in the old embedded processing world, but it wasn't do-able because there wasn't a Net-native embedded OS that was familiar to the very people who are doing the most to deploy the Net itself. Now that it's do-able, it will be very interesting to watch the progress across Lineo's list of market categories.

—Doc Searls

Archive Index Issue Table of Contents

Advanced search

Collecting RFCs

Peter H. Salus

Issue #75, July 2000

For over a decade, I have been acquiring RFCs.

Requests for Comment (RFCs) are the standards of the Internet. They began in the spring of 1969, when the ARPANET was under construction. RFC 1 (“Host Software” by Steve Crocker) is dated 04/07/1969.

For over a decade, I have been acquiring RFCs. I own several boxes and file drawers full of them. For the most part, they are one-sided xeroxes or laser prints. The former were copied from friends' copies; the latter, sucked from the IETF's web site and printed out.

This is actually quite unsatisfactory, largely because I tend to drop pages on the floor or shuffle them about and eventually end up with hundreds and hundreds of loose sheets that require many hours of resorting and replacing in folders.

Thus, when I heard about a plan to put a number of the RFCs relevant to IPv6 into paper-bound volumes, I was quite excited. I've now seen nearly a half dozen of the books, and I'm still excited. However, I have a problem with the series, so I'll come clean on things before going any further.

I wrote forewords to two of the volumes, and wrote both the introduction and made the selection where a third is concerned. So I am not exactly pure as the driven snow where the books are concerned. But in my defense, I must point out that I did these reprehensible things because of the perceived value of the collections.

The series is edited by Pete Loshin, and he deserves lots of gratitude for executing the project. The books are published by Morgan Kaufmann. Individual details follow.

The first volume I saw was the *Big Book of IPsec RFCs* (ISBN 0-12-455839-9). IPsec is the Internet Protocol Security Architecture; the book is made up of 23 RFCs which relate to it. In fact, if you are interested in Internet security *or* security in VPNs (virtual private networks), this book will be indispensable: it is the ultimate reference on the subject. The RFCs contained are:

- 1320. MD4 Message-Digest Algorithm
- 1321. MD5 Message-Digest Algorithm
- 1828. IP Authentication using Keyed MD5
- 1829. ESP DES-CBC Transform
- 2040. RC5, RC5-CBC, RC5-CBC-Pad and RC5-CTS Algorithms
- 2085. HMAC-MD5 IP Authentication with Replay Prevention
- 2104. HMAC
- 2144. CAST-128 Encryption Algorithm
- 2202. Test Cases for HMAC
- 2268. Description of RC2(r) Encryption Algorithm
- 2401. Security Architecture for the Internet Protocol
- 2402. IP Authentication Header

and a dozen more (2403-2412, 2451 and 2631).

Loshin might have written a bit more himself, rather than just compiling the material. But the material is there. And there is an extremely dense index, which means implementors will easily locate what they need.

The second volume, the *Big Book of World Wide Web RFCs* (ISBN 0-12-455841-0), contains 19 RFCs ranging from 1630 (which defines URLs) to 2718 (which defines new URL schemes). They literally cover everything that has been standardized for the Web.

Volumes three, *Big Book of Internet Host Standards* (ISBN 0-12-455844-5), and four, *Big Book of Internet File Transfer RFCs* (ISBN 0-12-455845-3), are the two for which I wrote the forewords. The former contains 11 RFCs (ranging from Jon Postel's 768, “UDP”, to 1127, “A Perspective on the Host Requirements RFCs”, Bob Braden's succinct and insightful “informational” document), but not:

> A host is a host from coast to coast<\n> and no one
> will talk to a host that's closeUnless the host (that isn't
> close)is busy, hung or dead.

FTP is one of the two “original” protocols. The first mail programs were “saddlebags” on FTP. The latter book contains 21 RFCs, running the gamut from 906 (“Bootstrap Loading using TFTP”) to 2640 (“Internationalization of the FTP”).

The volume I did is *Big Book of IPv6 Addressing RFCs* (ISBN 0-12-616770-2).

Future volumes will concern LDAP, BGP and Terminal Emulation RFCs. It looks like a fine series to me. But, as I said, I'm prejudiced.

email: peter@usenix.org

Peter H. Salus , the author of *A Quarter Century of UNIX* and *Casting the Net*, is an *LJ* contributing editor. He can be reached at peter@usenix.org.

Archive Index Issue Table of Contents

Advanced search

The Message

Doc Searls

Issue #75, July 2000

Doc discusses instant messaging.

AWZ.com is a teen site running on Linux that uses what may be the most popular teen application of our time: instant messaging. Lately, IM has become a hot topic, not so much because it's such a hot application (which it is), but because AOL seems hell-bent on dominating the category. For years, the company has been bundling AIM (AOL Instant Messenger) with not only their own proprietary on-line software, but with every copy of the Netscape browser. AOL also owns ICQ, the popular IM system developed by Mirabalis in Israel a few years back.

According to Tapan Joshi, the president of AWZ.com, AOL is trying to dominate IM the old-fashioned way: by working (both in software and the courts) to keep anyone else's instant-messaging software from interoperating with AOL's, regardless of the inconvenience this causes to AOL's own customers. In an April 27 op-ed piece in the *San Jose Mercury News*, Joshi lamented it isn't just Microsoft and Yahoo! that AOL wants to keep away from AIM and ICQ. It's little guys like AWZ.com.

Reading the piece brought to mind the last conversation I had with another IM pioneer: Udi Shapiro. It was in early 1998, not long after Udi had bought his company, Ubique, back from AOL. The on-line giant originally acquired Ubique for its technology, which subsumed and transcended “chat”, “buddy lists”, “instant messaging” and the other stuff AOL was also busy building into a Ubique competitor, the now-hyperfamiliar AIM. When it became clear that Ubique was getting the short shrift at AOL, Udi quietly repurchased the company, searched around for a proper benefactor, and found one in IBM's Lotus Development Corp. LDC purchased Ubique in May 1998 and now sells its technology as Lotus Sametime (which *PC Magazine* rates ahead of AIM, by the way). Meanwhile, Udi is back where he started, at the Weissman Institute in

Israel, working on computational and biochemical theories of the origin of life and stuff like that.

I'm glad I had that one lunch with Udi, back there between AOL and Lotus. During that lunch, Udi said the main lesson he learned at AOL was that instant messaging “changed the sociology of telephony” at any company that used it widely. Thanks to the buddy list in AIM, it became bad form at AOL corporate headquarters to bother a co-worker with a trivial call when a trivial instant message would do a better job. “Got a minute to talk?” and “Where are you going for lunch?” were better asked on the instant messenger than on the phone. New manners developed for both phone and IM. It was a remarkable development, Udi reported. Typically, this realization was mostly irrelevant to AOL's own concept for AIM, which was to serve as a frame for advertising messages delivered to the service's heaviest users: teenage girls chatting about boys and homework.

The next visionary to deliver insights to me about instant messaging was my friend Perry Evans, best known in our industry as the creator of Mapquest and to the rest of the world as the guy whose idea it was to create flavored bite-sized shredded wheat (or something like that), when he was a hungry kid working for Nabisco. Now the president and CEO of Webb.net, Perry let me know over drinks in Denver last year that he and Webb were doing for next-generation instant messaging roughly what Transmeta did for Linux, which was to give its progenitor a day job.

The Linus Torvalds of next-generation instant messaging—and a low-profile employee of Webb.net—is Jeremie Miller, a quiet, modest and brilliant programmer who lives in the farm country of Iowa. Jeremie's Linux is Jabber, “an open-source, cross-platform, completely extensible, end-all to Instant Messaging as we know it”, or so says the FAQ at the Jabber.org web site. But it's more than that. To understand how much more, you have to know two things.

First, instant messaging as we know it—almost entirely through AIM and ICQ—is a client/mainframe affair. When you're on an AIM client, you're hooked into exactly one server, in Vienna, Virginia. When you're on an ICQ client, you have the same relationship to another singular server. Same with Excite and Yahoo! That's four services, four giant servers and nearly zero interoperability between them—just like it was in the good old days of dial-up bulletin board services and on-line services. To be fair, Microsoft's MSN Messenger and Lotus's Sametime are both client/server systems. You can buy and deploy them on a server-by-server basis. But every one of these systems offers nothing in the way of real Internet Infrastructure, since they're as proprietary as a company boardroom and as open as a brick.

Second, about all most of us know so far about instant messaging are the three features we associate with AIM and ICQ: buddy lists, chat and instant messages. We'd also rather ignore AIM's and ICQ's other agendas, which are to serve as consumer marketing vehicles for branding, advertising or both.

Jabber is a totally different animal. Here's the way Perry put it when he blew my mind back in Denver: “Jabber puts a living XML document in the middle of a real-time conversation.” That means the content of an instant dialogue can be much more than typed text. For example, it can be a legal brief, collaboratively edited by a number of lawyers in a firm. It can be a live auction, with bids published for everyone in real time. It can be a repair history for your car, viewed by you and your dealer on your computers while you talk over the phone. It can be, well, just about anything.

On top of that, Jabber is completely XML-based, open source and deployed as an infrastructural service on the Net. An enterprise, a household or an ISP could have a Jabber server, just like they have a domain name server, a web server or a mail-hosting service. What's more, Jabber translates to any familiar instant-messaging flavor, including ICQ, AIM, IRC, MSN and the emerging IETF/IMPP protocol. Everyone can use their own client if they like, or they can use a Jabber client. Want more? Jabber also features message filtering, MIME encoding, crypto (e.g., OpenPGP) and game communication. Webb's Andre Durand, founder of Durand Communications and now general manager of Jabber.com, Inc. (a subsidiary of Webb) describes Jabber as “a real-time router for XML documents that has the potential to bridge not just proprietary IM networks, but wireless devices, embedded systems and a whole slew of new, real-time messaging-based applications which have yet to be developed.”

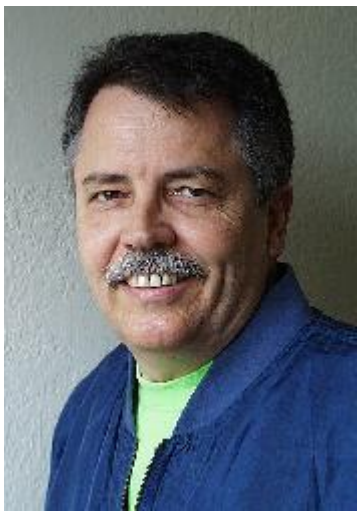
“You can create all kinds of applications that run on Jabber,” Perry says. “Voice-over IP integration, software development collaboration, community buying aggregation, auction and reverse auction systems, enterprise groupware—the list just goes on and on.”

Right now, Jabber's profile is still low. It's kind of like Mosaic before it was released by Marc Andreessen and his buddies at the University of Illinois. (You can find an early Jabber version deployed at Corel, if you want to start trying it out.) But a groundswell is building. In February, when I was on an open-source panel at the New York New Media Association, Daniel Frye of IBM began gushing about Jabber in a way that put even my own enthusiasm to shame.

So clearly, this is a space to watch. Unfortunately, Mr. Joshi of AWZ.com isn't watching closely enough. In his op-ed piece, he correctly calls AOL's belligerence “short-sighted” and instructs the company to “look no farther than

the Linux operating system.” He predicts “it is very likely that AOL is going to learn the Linux lesson the hard way.” But he misses Jabber.

Jabber will teach the lesson, but it won't be hard for anyone. That's because Jabber doesn't compete with AOL or anyone else. It simply enlarges their scope. “I look at instant messaging as a wild frontier,” Jeremie says. “Right now, there is nobody around. We're free to build all kinds of stuff, and everybody else is free to add value to what we're doing.” Sound familiar? Keep watching, or go to the developer section of Jabber.org and see what you can do for the cause.



Doc Searls (info@linuxjournal.com) is Senior Editor of *Linux Journal*, co-author of *The Cluetrain Manifesto* and a member of the advisory board for Jabber, Inc.

Archive Index Issue Table of Contents

Advanced search

Best of Technical Support

Various

Issue #75, July 2000

Our experts answer your technical questions.

Configuring ipchains

Current setup: I'm running **ipchains** as a firewall and to proxy my other machine to the Web via DSL. This Linux firewall has two NIC cards: one with a public IP address, the other on my private 10.100.100 network. I have a web server on the private network (10.100.100.20). How do I configure ipchains to redirect all web requests at my firewall to the web server on my internal network? —Mitchel Vernor, mitboy@hotmail.com

You'll need to use a port forwarding tool (such as **redir** or **ipmasqadm**) to "redirect" all requests that try to connect to port 80 (assuming you are using the default port) of your firewall's address to the web server's address. **redir** is a user-space program that has some limitations, but works with older kernel versions. **ipmasqadm** (which I recommend) works with 2.2.x (some 2.1.x also) and it is kernel-based.

Assuming all other input, output and forward chains are present, you should simply add:

```
ipmasqadm portfw -f
ipmasqadm portfw -a -P tcp -L EXTERNAL_FIREWALL_IP 80
-R 10.100.100.20 80
```

to your startup script. —Mario de Mello Bittencourt Neto, mneto@argo.com.br

More information on port forwarding under kernel versions 2.2 and higher, including examples, can be found at www.monmouth.demon.co.uk/ipsubs/portfw-2.2.html. —Chad Robinson, Chad.Robinson@brt.com

If you don't have ipmasqadm on your system, you can find it here: http://juanjox.kernelnotes.org/. You need at least kernel 2.2.x for this to work

(although there are patches for 2.0.x), and your kernel must have **IP: ipportfw masq support (EXPERIMENTAL)**, which is an option you can select only if you checked "Prompt for development and/or incomplete code/drivers" during configuration. —Marc Merlin, marc_bts@valinux.com

PPP Networking

Can I have a PPP connection that supports TCP/IP using a null modem between Win98 and Linux boxes? Any one of the machines needs to dial up to the other. I am using Red Hat 5.2. —Kiran, ajay@cc.usu.edu

Yes, you can. I would set up the Win98 machine to "call" the Linux system by double-clicking on a dial-up connection. I prefer this way, since the Windows machine expects to "log in" to the remote system and the Linux box can provide that "login". For this to work, you have to take care of the proper setup (pin out) of the serial cable that connects the machines together. Also, depending on the cable setup, it would be a good idea to tell the Windows machine, at the dialup port configuration setup, not to wait for the dial tone before dialing. The Linux box will not provide a dial tone. On the Linux side, you can create a new login ID with a password and configure it to have the PPP daemon as the login shell, instead of the usual bash. Also, you have to set up the serial port on the Linux box to provide the login automatically all the time. A good page that explains this in great detail is www.linuxgazette.com/issue41/smyth.html. —Felipe E. Barousse, fbarousse@piensa.com

Recovery after Partitioning

A friend who uses Debian 2.1 tried to install Red Hat 6.0 into another partition of his hard disk. The installation process of Red Hat has deleted the partition table of his hard drive. We need a way to recover some important files which are in the Debian partition. We can't access the partition. Can you explain some way to do it? —Alexis Serafin, sith@arrakis.es

If you know exactly what the partition table looked like before the crash, you can simply write a new partition table that looks exactly like the old one, mount the old partitions and back up the data. If the install process was aborted immediately after the new partition table was written, this alone may solve the problem. If all else fails, try this: make a partition that includes the whole disk (say, /dev/hdb1), mount that and raw-copy the entire disk to a big file (on another disk) with something like:

```
dd if=/dev/hdb1 of=/bigdisk/recovered.data bs=512
```

Then, try to piece the files together using /bigdisk/recovered.data. Personally, I've tried only the dd step, so there may be problems with this approach

beyond the fact that stitching up the files manually is awfully difficult. Consider it a desperation strategy. I know it's never helpful to hear this after the disaster occurs, and please don't think I'm not sympathetic, but the best way to deal with problems like these is always prevention: keep backups, and print out critical information such as partition tables before you need it. This is the voice of bitter experience speaking. —Scott Maxwell, maxwell@ScottMaxwell.org

While it is hard, you can look for the boot partition signature (55 AA at the end of the block, I believe) and locate the partitions on your disk. The above requires a disk editor, such as **diskedit** from Norton, and once you have the partition offsets, you can go to the partition table (first block of your disk, apply the partition table view and key in the numbers). Note that it's not trivial, and having another computer as a model to look at and copy from helps quite a bit. You can also try **fixdisktable** which automates this (only for primary partitions, though); it can be found at bmrc.berkeley.edu/people/chaffee/fat32.html. —Marc Merlin, marc_bts@valinux.com

Removing Another OS

A frequently asked question is how to remove Linux from a computer. What about removing Windows 98 safely? I had Windows 98 on the first partition of my hard disk (/dev/hda1) and Red Hat 6.0 on logical partitions (/dev/hda5, 6, ...). I was able to dual-boot using LILO, which was installed on the master boot record. One day, I decided to part with Windows 98 forever. I ran **mke2fs** on /dev/hda1 and divided it into three partitions. Now when I boot my computer, LILO does not appear at all, and I need to boot using a floppy.

I tried to install LILO (by running /sbin/lilo), but in vain. I tried removing LILO and re-installing it, but when I ran **/sbin/lilo -u**, I got the following error message:

```
The boot sector of /dev/hda does not have a LILO signature.
```

I booted the machine using a Windows 98 boot disk, ran **fdisk /mbr** and reinstalled LILO. Nothing doing.

I got a CD of TurboLinux from the March issue of *Linux Journal*, installed it on the new partitions I created and had LILO installed in the master boot record. It didn't work. What should I do to boot using LILO again? —Tam Laying, tamlayin@oupchina.com.hk

Make sure your lilo.conf file looks like this:

```
boot=/dev/hda
compact  # faster, but won't work on all systems.
ramdisk = 0
map=/boot/map
```

```
install=/boot/boot.b
prompt
timeout=100
image=/boot/vmlinuz
        label=linux
        root=/dev/hda5
        read-only
```

Then run **lilo**, and it should install a fresh boot block. Your problem may have been linked to partitions being shifted or renamed when you created the new ones. —Marc Merlin, marc_bts@valinux.com

Film and Video

When playing movie files in Linux, the video is very jumpy and there is no sound. I have set up my Soundblaster Live! which works okay. I also have a Voodoo2 3dfx card—do I have to set this card up to allow trouble-free movie playing? My video card is an ATI all-in-wonder pro with 16MB. The PC is a PII/333 with 128MB RAM. —Graham Bell, grahambell@bigfoot.com

There are many issues when playing video (and audio), not only in Linux but in other operating systems as well. First, if you intend to play video from the Internet, such as streaming video, make sure you get a good connection, the faster the better. This way, the video quality will improve substantially. There may be adjustments on your software related to quality of video vs. video-playback speed: the best image quality may be slower, and some frames may get lost. Also, be sure to check that the formats you are trying to play are consistent according to the software you are using in terms of versions. If your software does not fully support the video formats you want to play, it may not work as expected. I would set up all the hardware and make “local” tests to ensure everything works fine, then I would go watch on-line broadcasts. —Felipe E. Barousse, fbarousse@piensa.com

You did not mention which video program (xanim?) or video format (avi, mpeg, real video) you are using. There are some known issues with each format I've mentioned, and the client part available is somewhat limited. Recently, Real Networks released a client (RealPlayer) with better Linux support, and it is running okay. —Mario de Mello Bittencourt Neto, mneto@argo.com.br

Permission to Change

I am connecting a Linux server to a Macintosh. I have installed Netatalk, and everything seems to be running. When I go to the Macintosh, I can see the Linux server and access any file I want from it. However, if I try to save to the Linux server, I get a message telling me I do not have “Make Changes” permission and cannot do that. My Windows 98 machines work fine, so I do not believe it is a **chmod** issue. I think it is something I am missing in the NetaTalk configuration. —Aime Emery, aemery@northtroyconsulting.com

It looks like there is a problem related to passwords. NetaTalk must be compiled with shadow password support if your Linux uses them. Also, there is a huge variety of options to configure NetaTalk and all its related protocols. A useful web page for you to check on these parameters is http://thehamptons.com/anders/netatalk/. —Felipe E. Barousse, fbarousse@piensa.com

Missing Host Key?

After installing Red Hat Linux, when it is booted, I get:

Failed Message Starting sshd.error Could not load host key: /etc/ssh/ssh_host_key: no such file or directory How do I fix this? —Qamar Ansari, qamar@hotmail.com

sshd tries to find ssh_host_key by looking for the HostKey entry in its configuration file (normally /etc/ssh/sshd_config), falling back to a default of /etc/ssh/ssh_host_key. Ensure the HostKey file name in the configuration file matches the location of the ssh_host_key file you want to use. Some **ssh** installations put all ssh files directly under /etc, so maybe your file is there for some reason (although this is unlikely). If the ssh_host_key file doesn't exist at all, you can create a new one by running

```
ssh-keygen -f /etc/ssh/ssh_host_key
```

as root. —Scott Maxwell, maxwell@ScottMaxwell.org

Sender Not Receiving

I am unable to receive any e-mail from MSN; however, I can send mail. I am perplexed. I am positive I entered the POP3 server correctly. The problem is consistent with all mail programs I have tried. Although I am disenchanted with Microsoft, my choice of ISPs is limited here. —Frank Elston, felston@msn.com

Any ISP that answers the phone for a reasonable price is a good ISP; there is no shame in using MSN. Try getting your POP e-mail manually. Run the command **telnet my.pop.server**, where my.pop.server is the POP3 server given to you by MSN. When you connect, type the command **USER myusername**. Then type the command **PASS mypassword**. If you receive an error at any point, you will know if you are using the wrong server, user name and/or password. —Chad Robinson, Chad.Robinson@brt.com

First, are you positive about having POP service contracted from your local MSN shop? POP3 has several setup parameters; some of them are security related. Play around with SSL-enabled connections and test if it works by toggling any of

these configurations. Besides, MSN should provide you with all relevant information on how to connect to their service, at least from a Windows PC. From that information, you can take what is needed to configure your Linux machine. —Felipe E. Barousse, fbarousse@piensa.com

Resources

Archive Index Issue Table of Contents

Advanced search

New Products

Ellen M. Dahl

Issue #75, July 2000

RAIDpod, iBrow, Eon—The Anything Box and more.

RAIDpod

LAND-5 Corporation introduced their RAIDpod drop-in compact storage solution, designed to instantly add SCSI RAID disk array storage for any server enclosure. The RAIDpod holds up to three hot-swappable disk drives, supports all levels of RAID and offers up to 109GB of native storage. Up to five RAIDpods can be daisy-chained together to provide 546GB of storage capacity. The high-performance RAIDpod includes a powerful Wide Ultra2 backplane, offering transfer rates of up to 80MB/second, selectable on-board termination and backward compatibility with single-ended SCSI.

Contact: LAND-5, 9747 Business Park Ave., San Diego, CA 92131, 888-226-6544 (toll-free), 858-566-3611 (fax), http://www.land-5.com/.

iBrow

Merinta, Inc. announced details about the flexibility and customizability of the iBrow Solution, its end-to-end software platform that powers media-rich Internet Appliances (IAs). The iBrow software platform is Netscape 4.6.1-compliant and offers several benefits and areas of flexibility, such as multiple CPU chip set architectures; multiple operating systems, including Linux 2.2, a robust Java architecture and software backplane; and a unique remote

management feature, providing corporate customers with the option to have Merinta remotely update and modify the content on devices deployed worldwide. The iBrow Solution is hardware- and OS-independent.

Contact: Merinta, Inc., 9430 Research Blvd., Suite 200, Austin, TX 78759, 512-349-5800, 512-346-7062 (fax), info@merinta.com, http://www.merinta.com/.

Eon—The Anything Box

Neoware Systems, Inc. introduced Eon—The Anything Box, a customizable Linux information appliance designed specifically for business-to-business applications. It can be tailored for a variety of B2B applications and configurations including web kiosks, routers, firewalls, cash registers, thin clients, e-mail stations and security devices. Eon—The Anything Box is bundled with Neoware's new NeoLinux OS: a small, secure, centrally managed version of Red Hat Linux designed specifically for information appliances. It can be configured with software and add-in hardware for a wide variety of environments including wireless applications, as a network box or a point-of-sale station.

Contact: Neoware Systems, Inc., 800-636-9273, info@neoware.com, http://www.neoware.com/.

ATX35 E-Disk

BiTMICRO Networks unveiled the world's fastest EIDE solid-state flash disk, the ATX35 E-Disk. It is supported on a wide variety of platforms including Linux. The ATX35 has an average access time of 0.10 ms and 16.6MBps burst read and write transfer rates. Available in industry-standard 3.5-inch form factor, the ATX35 is fully ATA-2, ATA-3 and ATA-4 compatible and is available with up to 18,944MB of capacity. All E-Disks offer the highest levels of data integrity available on flash storage with their advanced Reed-Solomon Error Detection

and Correction, which protects customers' data and file systems against corruption.

Contact: BiTMICRO Networks, Inc., 48499 Milmont Dr., Fremont, CA 94538, 510-623-2341, 510-623-2342 (fax), info@bitmicro.com, http://www.bitmicro.com/.

LONE-TAR version 3.2.3.1

Lone Star Software Corp. announced a new release of its LONE-TAR backup product. Among the major enhancements in v3.2.3.1 are several bug fixes, support for long file names, improved RAW Partition support, hot binary restore, improved handling of error recovery when restoring from a pipe, and the **-zSYMLINK_ADJ** flag option, which can rename symbolic links on the fly while restoring, so they are all relative to a given directory. Available platforms include Linux.

Contact: Lone Star Software Corp., 509 East Ridgeville Blvd., Mount Airy, MD 21771, 800-525-8649, 301-829-1623 (fax), sales@cactus.com, http://www.LONE-TAR.com/.

Mortgage Builder through Linux

MBSI announced a new platform for its loan origination system, Mortgage Builder through Linux. The Linux port gives mortgage brokers, bankers and credit unions a low-maintenance server that is fast and stable. Mortgage Builder was created with a new generation of computer application software tools called System Builder by Informix Software. It provides users with a post-relational 3-D database that allows up to 1,000 users to operate on a variety of platforms.

Contact: Mortgage Builder Software, Inc., 24370 Northwestern Hwy., Suite 310, Southfield, MI 48075, 248-208-6142 (fax), http://www.mortgagebuilder.com/.

RAID Controllers

ICP vortex began shipping 32-Bit PCI-wide/Ultra2 SCSI RAID controllers, offering higher performance levels for low-end controllers. The ICP GDT6118RS, GDT6128RS, GDT6518RS, GDT6528RS and GDT6538RS models offer one, two or three Ultra2 channels and support data transfer rates of 80MB/sec per channel, with up to 15 devices attached to each channel on cables up to 12 meters in length. The GDT-RS series controllers feature the fully integrated i960 RS processor from Intel running at 100MHz and an SDRAM interface. The controllers are PCI 2.2-compliant with full bus mastering capabilities. All ICP controllers support Linux, are I2O ready and fulfill all standards for CE and FCC.

Contact: ICP vortex Corporation, 4001 E. Broadway, B-20, Phoenix, AZ 85040, 602-414-0414, 602-414-0444 (fax), sales@icp-vortex.com, http://www.icp-vortex.com/.

Red Hat Linux 6.2 and eCos 1.3

Red Hat, Inc. introduced its complete open-source family of operating systems for e-commerce servers and Internet-enabled, hand-held and embedded devices. Version 6.2 is the latest edition of the Red Hat Linux distribution. eCos (embedded configurable operating system) 1.3 is the latest version of Red Hat's application-specific OS for embedded systems. The configurability of eCos 1.3 enables developers to create an embedded device, and it features a built-in TCP/IP stack that enables embedded devices to communicate with the Internet. In addition to the Professional Edition, Red Hat Linux 6.2 is also available in a standard and a deluxe edition.

Contact: Red Hat, Inc., ecos-info@redhat.com, http://www.redhat.com/services/ecos/.

smartNIC Linux Driver

smartBridges announced the release of its Linux software driver for the smartNIC USB networking adapter. The smartNIC adapter directly connects the hot-plug feature of the USB port into the Ethernet network, allowing the system

to bring up networking support on the fly. The driver also automatically updates the network configuration, making the system network-enabled and ready to use. The adapter draws its power from the USB port itself. For mobile computer users, the adapter functions like a conventional PCMCIA network card. For desktop computer users, it functions like a conventional PCI networking card.

Contact: smartBridges Pte. Ltd., 21 Heng Mui Keng Terrace, KRDL Bldg., Singapore 119613, Singapore, +65-773-9230, +65-774-1282 (fax), info@smartbridges.com, www.smartbridges.com/smartNIC/smartnic.html.

SuSE Linux version 6.4

SuSE announced the newest version of SuSE Linux. Features of version 6.4 include easier installation with enhancement to SuSE's graphical installation tool, YaST2; expanded and simplified hardware detection and configuration; a quick-start beginner's manual; support for memory to 3.5GB; SuSE Proxy Suite, SuSE's open source firewall tool and FTP proxy; and a journaling Reiser file system.

Contact: SuSE GmbH, Schanzaeckerstr. 10, D-90443 Nurnberg, Germany, +49-911-740-53-31, +49-911-741-77-55, suse@suse.de, http://www.suse.de/.

uClinux System Builder Kit

Rt-Control announced a new release of the uClinux system builder kit, version 2.0.38.1-pre5. The kit adds support for the ARM7TDMI and i960 microprocessors, providing multiple platform support for deeply embedded microprocessors. uClinux is a Linux-based distribution targeted at microprocessors that do not support memory management units (MMU). The latest uClinux system builder kit CD-ROM is available at the uClinux web site, https://www.uclinux.com/orderdesk.

Contact: Rt-Control Inc., 195 The West Mall Suite 608, Toronto, ON M9C 5K1, Canada, 416-241-2708, 416-243-1173 (fax), info@rt-control.com, http://www.rt-control.com/.

SuperAuk DSL MultiServer 6.1

WholeLinux, Inc. released its debut product, the SuperAuk DSL MultiServer 6.1. The full-blown network appliance software installs automatically in four minutes on almost any PC, and is the first desktop-based administration scheme allowing push-button maintenance of firewall, mail, web and file shares for local networks of Windows and Macintoshs. A Windows machine can become a secure DSL server which ties together Windows, Macintosh and Linux boxes on a local network, running Linux applications while the DSL server runs in the background. An HTML Wizard allows graphical network setup. The resulting SuperAuk desktop will also function as a standard Linux workstation.

Contact: WholeLinux, Inc., 2610 Chanticleer Ave., Santa Cruz, CA 95065, 831-462-4881, info@wholelinux.com, http://www.wholelinux.com/.

WordPerfect Office 2000 for Linux

Corel Corporation began shipping its WordPerfect Office 2000 for Linux, a comprehensive office productivity suite. Both the Standard and Deluxe Editions include the download version of Corel LINUX OS, featuring an updated 2.2.14 kernel and X-server 3.3.6, new control panels for hardware management and USB support, improved hardware support during the automatic install, preliminary RPM support in Corel Update, more hardware drivers for video and sound, and improved Windows connectivity in the File Manager. Both editions install on major Linux distributions and are compatible with the Windows version of WordPerfect Office 2000 and Microsoft Office 2000.

Contact: Corel Linux, 800-772-6735, 561-733-6033 (fax), custserv2@corel.com, http://linux.corel.com/.

Diffpack Datafilter Toolbox 1.0

Numerical Objects AS announced their Diffpack Datafilter Toolbox 1.0. This plug-in module to the Diffpack Multi-Physics Simulation Framework provides easy and safe import facilities for third-party finite element meshes. The Datafilter Toolbox can be used either as a stand-alone file-to-file converter or as a linkable library facilitating direct mesh import into customers' Diffpack applications. The Diffpack product line is available on Red Hat Linux 6.0/6.1 and other platforms.

Contact: Numerical Objects AS, Forskningsveien 1, P.O. Box 124 Blindern, N-0314 Oslo, Norway, +47-22-06-73-00, +47-22-06-73-50 (fax), sales@nobjects.com, http://www.nobjects.com/.

Linux-Mandrake Secure Server 7.0

Macmillan USA, partnering with InformIT, announced Secure Server 7.0, a secure web server built within the new Linux-Mandrake 7.0 distribution. Secure Server 7.0 provides graphical tools for easy installation. The Apache-based web server utilizes RSA's BSAFE SSL-C technology for encryption and security. Secure Server 7.0 is designed for Linux professional web server administrators responsible for managing an e-commerce site, intranet or any web site requiring security.

Contact: InformIT, 800-716-0044, orders@informit.com, http://www.informit.com/linux/.

Archive Index Issue Table of Contents

Advanced search

Mastering Algorithms with C

John Kacur

Issue #75, July 2000

While you might not have “mastered” algorithms after reading this book, it is still a well-written book that I recommend without reservations.

- Author: Kyle Loudon
- Publisher: O'Reilly & Associates
- E-mail: info@ora.com
- Price: $34.95 US
- ISBN: 1-56592-453-3
- Reviewer: John Kacur

While you might not have “mastered” algorithms after reading this book, it is still a well-written book that I recommend without reservations. In the preface, the author explains why his “approach is not what one normally thinks of in connection with books on data structures and algorithms.” He rightly explains that many books on data structures and algorithms have an “academic feel about them, and real details such as implementation and application are left to be resolved elsewhere.” Indeed, the strength of this book is that instead of snippets of code, we are presented with full programs as useful implementations.

The book is divided into three parts. Part I, *Preliminaries,* is the shortest. The author doesn't try to teach the reader C, but instead provides a useful review of some tricky topics such as the use of pointers (generic pointers, function pointers casts and so on). He touches lightly on recursion and reminds us what tail recursion is and why it is efficient. There is also an overview of O-Notation for analyzing algorithms. Some people might complain that the book doesn't go into enough depth here, but if you view it as a companion text to the more theoretical books from your college courses, then this is just the right amount of information you need to continue on to the implementations.

Part II, *Data Structures*, is where the book starts to shine. There are chapters on linked lists, stacks and queues, sets, hash tables, trees, heaps and priority queues and graphs. Each chapter is broken down further. For example, the one on linked lists discusses singularly linked lists, doubly linked lists and circular lists. The implementation of linked lists is where we first see the value of Loudon's good software engineering practices. Public interfaces are documented in separate header files, and private functions are static so they remain in file scope.

Because programming styles tend to be personal, some readers are likely to quibble with Loudon's coding conventions. However, since he picked a style and applied it consistently, his code is very clean and readable. For example, all structures have typedefs and names, where the name of the structure is the name in the typedef followed by an underscore. No shortcuts are taken, so unlike many books that demonstrate the principles of a linked list by using a data type of int, Loudon uses a pointer to void for a generic implementation that can use any data type.

Part III is called *Algorithms*. The chapters here are Sorting and Searching, Numerical Methods, Data Compression, Data Encryption, Graph Algorithms and Geometric Algorithms. While these chapters are not necessarily comprehensive (and how could a one-volume book be comprehensive?), Loudon presents some interesting topics that are *not* traditionally covered in books on algorithms, such as data compression and data encryption. It was particularly interesting to read about Lempel-Ziv compression, given the recent copyright controversy with GIF graphics.

Conclusion

If you are a beginning programmer, you should first read a book such as *The C Programming Language* by Kernighan and Ritchie. If you are new to data structures and algorithms, this is an excellent book with real implementations to study. I would recommend it as a companion to the more traditional academic books typically assigned in college courses. If you are an intermediate to expert programmer, you might still appreciate this book as a practical reference that won't bog you down in theoretical detail, yet will allow you to get a program up and running quickly.

John Kacur (jkacur@acm.org) has a B.A. in Fine Arts, and a B.Sc. in Computer Science. He recently moved to Toronto, Canada to accept a job with IBM.

Archive Index Issue Table of Contents

Advanced search

Book Review: Red Hat Linux 6 in Small Business

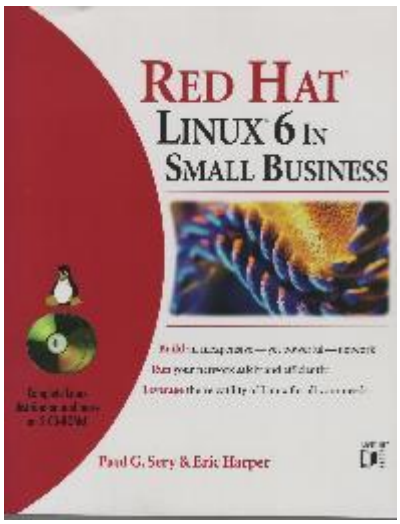
Paul Dunne

Issue #75, July 2000

Red Hat Linux 6 in Small Business, by Paul G. Sery and Eric Harper

Red Hat Linux 6 in Small Business

- Authors: Paul G. Sery and Eric Harper
- Publisher: M&T Books (An IDG imprint)
- Price: $29.99 US
- ISBN: 0-7645-3335-5
- Reviewer: Paul Dunne



Well, out of all the books I've reviewed for *Linux Journal*, this is the one with which I was least satisfied. As a writer myself, I take no great pleasure in having to criticize another's work. But in this instance, I must.

This is not a fat book—and there's nothing wrong with that, too many computer books are padded out—but, unfortunately in this instance, the lean has been cut away with the fat. Let me step through the chapters and explain what's wrong.

Part I

After the first chapter, on installing Red Hat, we move to chapter 2, “Navigating Linux”, a straightforward although somewhat brief introduction to the shell. It is brief at least partly because, to quote the authors, “this book relies heavily on the use of the graphical X interface”. This is an unwise dependence. The command line is not an optional alternative. There is no unified graphical interface to administering a Linux box; it's command line or bust. Sugaring the pill in the way our authors attempt to do simply does not work.

Chapter 3, on editing files, is a decent if short introduction to joe and vi. It would be greatly improved by the inclusion of a command summary for each editor considered.

Chapter 4, configuring the X Window System, is in my view out of place in the book; but even so, it is inadequate. It is far too short, and does little more than tell the user to run Xconfigurator.

Chapter 5, on getting help, is a concise guide to sources for further Linux information.

Now comes Part II, Managing Your Linux Network, and here our troubles really begin.

We start with the sixth chapter, on system administration. This is wholly inadequate. It attempts to paper over the complexity of this job by introducing the user to some graphical tools bundled with Red Hat. As I've said above, this approach is fundamentally misguided.

Chapter 7, Managing Your Network, is better, but again it suffers from the fault of simply directing the reader elsewhere when it comes to the hard stuff. For example, “The experienced network administrator can modify any of these configuration files by hand ... Just be aware that the scripts and files are often interrelated with each other and modifying one can change another's behaviour. You can view the man pages of these to gain more understanding of the services”. Well, quite! An explanation of the complex structure of Red Hat's System V style rc file layout would have been particularly helpful here. Simply referring the reader to the man pages is a cop-out.

Following is a chapter on Samba and one on printers. In the latter, there is no explanation of /etc/printcap, the basic printer config file on every Linux system. We are shown some GUI tool to make changes. We leave the chapter much as we entered it, with no understanding of how printers work under Linux, nor how to configure them.

Chapter 10, backups, is basically an explanation of the Arkeia backup software bundled on the companion CD-ROM. I found it a good guide to using the software.

Part III, Connecting Your Network to the Internet, starts with Chapter 11 on connecting to the Internet. Off we go again—"Using the network configuration"—but no explanation of what this program is actually doing. "Configuring a DNS server from scratch can be a very complex task", a task which the authors duck by pointing the reader to a few example files on the CD. It would have been better to leave the topic out altogether than deal with it so half-heartedly.

As a side note, dip has been obsolete for years. Why no mention of pppd? To their credit, the authors do give an introduction to using diald.

Chapter 12 covers creating a simple firewall. Although short, this chapter is good. It doesn't avoid explaining the command necessary to set up a Linux packet-filtering firewall, but again, there is no explanation of why the firewall rules listed are used.

Chapter 13 is on configuring a Linux e-mail server. This is very bad—one and a half pages on Sendmail! They take up more space in explaining how to use the Netscape e-mail client. Simply put, this chapter will not tell you how to configure a Linux e-mail server.

Part IV takes a complete change of focus, looking at "office productivity" tools, then back to networking by installing Apache. The latter chapter is again ridiculously short, and plainly omits any detailed explanation of exactly how Apache is configured. As for the former, well, given the size of the book, they can either do a good job of explaining how to use Linux as the main server for a small business, or they can look at it as a desktop platform. There isn't space for both. Perhaps this partially explains why the book ends up doing neither.

We finish with a curious document, the IDG Books Worldwide Inc. End-User License Agreement, which contains the following gems:

> 1. License Grant. IDGB grants to you (either an individual or entity) a non-exclusive license to use one copy of the enclosed software program(s) (collectively, "the Software") solely for your own personal or business purposes on a single computer (with a standard computer or a workstation component of a multiuser network). [...]

Whoa! You *what*?

> 2. Ownership. IDGB is the owner of all right, title and interest, including copyright, in and to the compilation of the Software [...]

Seems reasonable. Copyright on the *compilation*, not on the software itself.

> 3. Restrictions on Use and Transfer.
>
> (a) You may only (i) make one copy of the Software for backup or archival purposes [...] You may not (i) rent or lease the Software, (ii) copy or reproduce the Software through a LAN or other network system [...], or (iii) modify, adapt, or create derivative works based on the Software.

and a lot more of the same. This is just plain wrong, it seems to me.

> 4. Restrictions on Use of Individual Programs. [...] None of the material on this Software Media or listed in this book may ever be redistributed, in original or modified form, for commercial purposes.

Hey—I can do what I want with Red Hat, including burning and selling my own CDs.

The book also includes the GPL (right after this license) and a note to the effect that the Red Hat distribution may be used "in accordance with the GNU General Public License" In which case, surely the IDG license does *not* apply to the Software? So why are they bothering to print it in the book?

Insofar as this license agreement purports to apply to the Linux components of the CDs bundled with the book, then it is plain wrong. For instance, we read, "You may only (i) make one copy of the Software for backup or archival purposes". The only copyright IDGB has here is to the bundle itself; that is to say, to the collection. I can make as many copies of Red Hat as I like, and IDGB can do nothing to stop me. This license is out of place in a book about open source.

Conclusions

In conclusion, this book is unsatisfactory. It is too sketchy, tries to cover too much ground, and is short on detailed, hands-on demonstrations of what to do. There is no clear focus and, as a result, the book fails to be either a detailed technical resource for system administrators or a decent introduction to Linux as a server OS for the less experienced.

It is true that some parts are better than others. The firewall chapter, for instance, is a decent (albeit brief) introduction to this important subject. But, in contrast (and this is unhappily the more common case), the chapter entitled “Configuring a Linux E-mail Server” simply doesn't deliver what it promises.

At the very least, the book needs to be supplemented with on-line resources. In that case, why buy the book? Of course, given the speed with which Linux changes and the inevitable lag in publication times, any book will need supplementing by on-line resources to a degree. Here, I'm afraid, the book is so heavily dependent on them as to render it superfluous.

Paul Dunne (paul.dunne@bigfoot.com) is an Irish writer and consultant who specializes in Linux. The only deadline he has ever met was the one for his very first article. His home page is at dunne.home.dhs.org.

Archive Index Issue Table of Contents

Advanced search

Low-Bandwidth Communication Tools for Science

Enrique Canessa

Clement Onime

Issue #75, July 2000

No access to the Internet? Browse the Web via e-mail instead!

Dissemination and management of knowledge is essential for scientific enterprise and sustainable development. For several decades, the Abdus Salam International Centre for Theoretical Physics (ICTP) in Trieste, Italy, has paid special attention to the needs of developing countries to foster, through training and research, the progress of science.

The Centre long ago realized the importance of information retrieval systems on the Internet, including the distribution of in-house preprints, yearly activities and public access catalogs.

On a technical level, Linux provides us with a cost-effective alternative for promoting distance electronic collaboration (see Resources). Based on the Linux OS, virtual laboratories and the extensive use of digital communication tools can help reduce scientific isolation, while filling the need to transfer knowledge to developing countries in the Southern Hemisphere in an unprecedented way (see Resources).

Following these principles, we have started building prototype, on-line scientific tools to further enhance electronic collaboration and to support the use of web navigation and database search by e-mail. Below, we describe two tools that Salam ICTP offers the low-bandwidth scientific community. Both packages use state-of-the-art technologies and software developed in-house, and are distributed under the GNU General Public License (GPL).

www4mail—Web Navigation and Database Search by E-Mail

The ICTP **www4mail** software allows navigation and search of the entire Internet via e-mail, using any standard web browser and a MIME (Multipurpose Internet Mail Exchange)-aware e-mail program. At first glance, it may appear similar to one of the several web-to-mail software interfaces; but the www4mail program introduces new features not previously available. In short, e-mail messages containing filtered HTML pages are automatically passed to the www4mail server when links to other web sites are selected while browsing.

Written in modular Perl, the program allows retrieval of web pages, searching of arbitrary databases, filling out of web forms (GET and POST conduct web database searches) and following of links (on-line browsing), all by e-mail. It is multi-lingual, easy to manage and supports current Internet standards (MIME, HTML 4.0, etc.).

Developed from scratch on the Linux platform, **www4mail** has been used successfully on the BSD platform and contains some optional optimisations that are Linux-specific. For example, www4mail can monitor the system load average, directly from the Linux /proc file system and, at high load averages, queue requests for later processing.

Here are some major features of www4mail:

- sends replies as e-mail attachments or in the body of an e-mail message, depending on the type of request options sent by the e-mail client through the web browser
- supports scripting, once the browser can display it
- delivers most types of web documents, including JavaScript and cookies
- handles dynamic contents, parsing text HTML and source HTML
- preserves the original layout of requested web pages
- retrieves information from FTP sites and Usenet news servers
- handles meta tags; that is, if a web page is redirected or relocated by the use of a meta statement, www4mail automatically warns about the possible relocation of the information and provides suitable links for the new location at the top of the reply page
- handles frames, inserting suitable links to each framed document
- supports user authentication for password-protected web/FTP sites
- traps error messages and sends them back to the user
- provides support for text-only access for compatibility with the alternative “Agora” and “GetWeb” web-mail servers
- serves filtered requests to reduce bandwidth

- supports the transfer of binary data
- allows web pages to be downloaded as PostScript files, to be viewed or printed locally (see for manuals)

4.23.00 - It was in the C-edit directory as of May 1.

www4mail (see logo in Figure 1) was developed mainly to help researchers from developing countries browse the Web using only e-mail and slow Internet links. While the amount of information on the Web has grown exponentially in the last few years, there is still a large community of Internet users who have access to only e-mail, or their Internet providers do not offer full Internet connections (some of them still use UUCP) or who cannof afford to have an expensive account with full Internet capabilities. Many of these users live in rural areas of developing countries, and rely on e-mail to access essential medical and business information as well as for interpersonal communication and world news. Having the ability to query available databases (such as AltaVista, HotBot, etc.) or preprint repositories with one simple e-mail and receive the output in a few minutes (or hours) could help them tremendously with their scientific work.

At present, www4mail can be tested by sending an e-mail message to www4mail@wm.ictp.trieste.it, or to any other place where the gateway is installed (e.g., Bellanet-Canada, www.bellanet.org/email.htm), listing the requested URL(s) in the body of the message.

Over 50 server configuration options are currently available for setting parameters such as maximum quota per user, gateway administrators, maximum size of each request, or to split sizes for large files. (Type **help** in the body of the e-mail message for further details).

The installation procedure of the server is simple. For example, under Red Hat Linux, create a user account called www4mail (**adduser www4mail**), log on as user www4mail (**su - www4mail**), extract the tar archive in the home directory for www4mail (**tar zxvf www4mail.tar.gz**) and perform a few extra operations (e.g., to enable forwarding). It is necessary to create a link from the executable /home/www4mail/bin/www4mail to the /etc/smrsh subdirectory in order to keep the sendmail MTA (mail transfer agent) happy. To optimize its configuration, some preliminary monitoring is necessary.

www4mail has been very useful for many people from many different countries, often receiving over 12,000 requests per day. You can view weekly statistics at http://web.bellanet.org/www4mail/).

ScientificTalk—Real-Time Mathematics Discussions via Web

ScientificTalk (see logo in Figure 2) is a profession-specific prototype tool for scientists, students and teachers to exchange information via a web browser, using a display of math equations in a synchronous manner. The project focus is on users' interests in such things as mathematics and scientific notation. Our motivation follows an early goal for the Web to be a readable and writable collaborative medium.

Unfortunately, the large tag repertoire of the HTML 4.0 language does not cater to mathematics, since they cannot mark up complex mathematical expressions. Usually, to create technical documents with mathematical or scientific content, web authors resort to methods involving images (e.g., screen captures of equations), which means the sharing of scholastic and scientific material by lecturers, students, etc. is often a many-step process. There are a few available applets and plug-ins that can render MathML in a browser (which are not necessarily designed for synchronous collaboration).

The Mathematical Markup Language, MathML, is a recommendation of the W3C, which provides a foundation for including mathematical expressions in web pages. As an application of the Extensible Markup Language (XML) and with adequate style-sheet support, MathML will ultimately make it possible for browsers to natively render math expressions, including threaded on-line discussions. (Some applets and plug-ins are currently available, which can render MathML in a browser.) See W3C at http://www.w3.org/Math/ for a complete list of technical/scientific document viewers and renderers such as the Scientific MessageBoard WebEQ, the IBM techexplorer, EzMath editor and LaTex2HTML.

ScientificTalk is a Perl script for a standard multiway graphical web chat. This CGI-based application is portable across platforms and allows the viewing of occupants, sending input to specific users, etc. While chatting on-line, it converts textual input or standard LaTeX—a popular computer language for composing formatted scientific text for high-quality printing—into HTML. The math displayed on the browser is rich because of the LaTeX typesetting and Ian Hutchinson's powerful TeX-to-HTML translator, **tth**, available at http://hutchinson.belmont.ma.us/tth/.

For those not familiar with LaTeX commands, ScientificTalk has an external symbols keyboard as well as a composer and messenger window for the user input. There is no need for extra plug-ins or high-speed networks—all input is passed via text-mode only. On the client side, Netscape (v4.0 or greater) needs a simple character-set configuration (for details and demo, connect to http://sv7.ictp.trieste.it/.

Although the ScientificTalk prototype has proven that it is possible to carry out synchronous math discussions on the Web between distant clients today, our to-do list is still long. For example, it would be useful to save a complete session as a LaTeX file (in order to restart an on-line discussion from a given session or collaboratively write LaTeX documents on the Web), display plots from a given function, create small transparent .gif files, and extend its language capabilities to symbols in other domains, such as chemistry.

Concluding Remarks

More opportunities for learning and growth are available if we can shar ideas via a computer environment that is responsive to our professional needs. For example, using simplified scientific notation on the Web can lead to faster, more effective results. Electronic tools, designed for collaboration and based on Linux, will continue to play an important role in an increasingly interconnected world. Off-line browsing via web-to-e-mail servers such as www4mail is still a reality from remote areas of the world, and most likely will remain so, as the number of Internet users is expected to double to 300 million by the year 2005.

Acknowledgements

Resources

Dr. Enrique Canessa (canessae@ictp.trieste.it) is a theoretical physicist currently working as a scientific consultant at the ICTP. His main areas of research and interest are in the field of condensed matter and scientific software applications. He has been lost in the Internet since 1987.

Clement Onime

Archive Index Issue Table of Contents

Advanced search

Security Technologies for the World Wide Web

Wael Hassan

Issue #75, July 2000

The main topics discussed are HTTP authentication, proxy servers and firewalls, cryptographic techniques, Internet security protocols and Secure Socket layer (SSL).

- Author: Rolf Oppliger
- Publisher: Artech House
- E-mail: artech@artechhouse.com
- Price: $69 US
- ISBN: 1-58053-045-1
- Reviewer: Wael A. Hassan

Having been in the security and networking field for several years now, I have read many technical books about web servers, networking technologies and cryptography. These books discussed different aspects of the technologies, but not a general view of the Web as an entity. Moreover, I have always wanted to see a book that could be read by somebody who is not a specialist. Rolf Oppliger, the writer, works for the IT Security Group of the Swiss Federal Office of Information Technology and Systems (BFI). He teaches at the University of Zurich, Switzerland, and is also the author of Internet and Intranet Security (Artech House, 1998) and Authentication Systems for Secure Networks (Artech House, 1996). Dr. Oppliger is the computer security series editor at Artech House and received his M.Sc. and Ph.D. in Computer Science from the University of Berne, Switzerland.

The book has about fifteen chapters. The main topics discussed are HTTP authentication, proxy servers and firewalls, cryptographic techniques, Internet security protocols and Secure Socket layer (SSL). He then moves on to explain electronic payment systems, certificate management and network access layer security in detail . After that, he covers certificate management, executable content and scripting languages, mobile code and copyrights. In the last few

chapters, the author touches on issues like privacy protection, anonymous browsing and censorship on the World Wide Web. Very few writers cover these subjects.

This book explains the different aspects of the Web, from both the user point of view and system administrator or network architect. It has plenty of references, web sites and pointers to where information can be found. Very well-written, it states a list that's almost like an index to technologies under discussion. Whereas technologies such as encryption, security and Hyper Text Transfer Protocol have copious sources of information, Electronic Commerce does not have enough references. Rolf points out several sources where information about e-commerce can be found. I particularly liked the section on electronic commerce.

The book targets security for the most part; the discussion, however, is restricted to IP-based networks. Other different network protocols and technologies such as ATM and frame relay did not get their share of the discussion. A deeper explanation of virtual private networks would have enriched the book.

In conclusion, Rolf Oppliger presents what can be considered a premier introduction to the end-to-end security model, environment, tools, protocols, as well as privacy. It is not too technical, and is thus appropriate for a broad audience. I certainly recommend getting a copy; it is worthwhile reading.

Wael Hassan (wael@ieee.org) likes security and networks and enjoys code writing. He says that the greatest inventions come unexpectedly. That's why he prefers to think on the beach or while he is skating.

Archive Index Issue Table of Contents

Advanced search

Getting Started in Computer Consulting

Ralph Krause

Issue #75, July 2000

This book provides an overview of a computer consultant's life and gives you general information you can use to plan and run your business.

- Author: Peter Meyer
- Publisher: John Wiley & Sons
- E-mail: info@wiley.com
- Price: $18.95 US
- ISBN: 0471348139
- Reviewer: Ralph Krause

The siren call of a career in computer consulting: leaving the anonymity of a big company, being your own boss, and surviving by your wits. With today's good economy and the growth of Linux, you may think the time is right to take the plunge. But, do you know what services to offer in your area? Should you bill by the hour or by the project? Should you use a broker, or not? *Getting Started in Computer Consulting*, written by Peter Meyer, can help you answer these questions and make your new career a successful one.

The book does not provide step-by-step instructions for incorporating, accounting, or the myriad of other details required for running a business. Neither does it concentrate on things like marketing or the art of contract writing. Instead, it provides an overview of a computer consultant's life and gives you general information you can use to plan and run your business.

The book starts by explaining a little bit of what computer consultants do, followed by information on determining what services you should offer, how you could set up your business and methods of marketing and pricing your services. It finishes up by taking you through some of the ethical problems a consultant might face. The book is a bit uneven in places; some chapters are vague, while others provide quite a bit of detail.

A few charts and tables are in the book, including charts showing hours worked by consultants and consultant income. Many of the computer and marketing terms used throughout are defined on the pages where they are used. The book also contains a glossary, which some people might find helpful. Some web addresses for consulting organizations and job listings are also given.

I consider the book's most useful aspect to be the marketing information provided. It explains how to distinguish yourself to your potential clients by stressing the benefits they get by hiring you (e.g., increased sales) instead of the skills you possess (e.g., database consultant). Mr. Meyer calls this a USP or Unique Service Proposition, and he provides examples of good ones as well as not-so-good ones.

The chapters on determining your rates are also very useful at the beginning of a consulting career. They help you determine whether you should charge a flat rate or by the hour, when to raise your rates, and the signals that high rates send to a client. A seven-step process is presented for getting a contract at a price you want for work the client deems important.

The book provides information on finding out about and obtaining state and federal government contracts. It also tells you under which circumstances you can bypass the lengthy bid process to be awarded a contract quickly.

One chapter is dedicated to brokers, negotiations and contracts. It tries to help you decide whether you should use a broker and what things to look for before choosing one. It explains the benefits you can get from contracts, and in what situations they will not help you. Finally, it explains how you can negotiate with a potential client so that both of you are satisfied with the outcome.

Working for clients might not be your only source of income as a consultant. The book talks about the benefits of writing, lecturing and creating your own software on the side. In addition to generating income during slack times, these tasks also serve to increase your network of contacts and promote name recognition.

As a consultant, your most valuable asset is your reputation. Some common ethical considerations, such as working for competing clients and what to do if you overbill, are given in the final chapter. These examples can help you avoid the not-so-obvious pitfalls you might encounter in the beginning of your consulting career.

This book does not provide detailed instructions on starting and running a business. It does help you decide whether you want to be a computer consultant, and gives an idea of what you need to think out ahead of time and

plan for in order to become a successful one. While the book is a little vague in some areas, I think its discussions of billable hours, marketing services and determining prices make up for this. *Getting Started in Computer Consulting* makes a good addition to the beginning consultant's library.

Ralph Krause (rkrause@netperson.net) is a freelance computer consultant in Michigan. His current career goal is to stay out of automotive paint shops.

Archive Index Issue Table of Contents

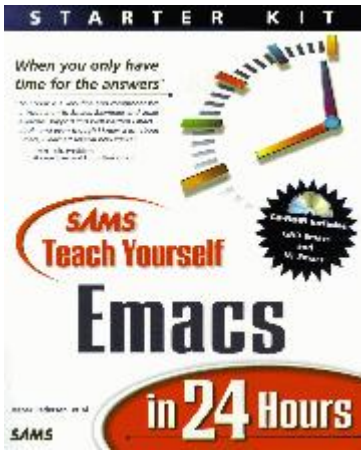
Advanced search

Teach Yourself Emacs in 24 Hours

Ralph Krause

Issue #75, July 2000

The authors seem to know Emacs well, and they have done a good job of selecting the features to present to new users.



- Authors: Jesper Pederson, Jari Aalto, Charles Curley, Eric Ludlum, Larry Ayers and Jeff Koch
- Publisher: SAMS
- URL: http://www.mcp.com/publishers/sams/
- Price: $24.99 US
- ISBN: 0672315947
- Reviewer: Ralph Krause

I have been using Linux for about a year now, and in this time, I have used Vim for text editing. I had tried Emacs a few times, but found it difficult to learn. I decided to try learning Emacs again with help from SAMS' *Teach Yourself Emacs in 24 Hours*.

While I can say I learned enough from the book to use Emacs (I'm using it to write this review), I would hesitate to recommend this book to anyone else. The

biggest problem I found was the sheer number of typos it contained. In addition to that, the accompanying CD-ROM is not well-organized, either.

Let me quickly say that the authors seem to know Emacs well, and they have done a good job of selecting the features to present to new users. The examples they give are generally clear and illuminating. They also highlight the differences between Emacs and XEmacs in the examples, so the book is useful no matter which version of Emacs you are using.

If you do purchase this book, the first thing you should do is go to Mr. Pedersen's web site at www.imada.ou.dk/~blackie/emacs and print out the book's errata pages. Unfortunately, even with these, I still couldn't get some of the book's examples to work, such as running the etags command on HTML files.

In addition to the printed errors, I also ran into problems with the accompanying CD. The first was that I couldn't access it on my Linux box unless I was logged in as root. Doing work as root is not a good idea in general, and in some instances, at work for example, it may not even be possible. When I did access the CD, I found that its directory structure left something to be desired.

The CD's root directory contains only two entries: opt and usr. The opt directory contains the Windows and Linux versions of Emacs, while the usr directory contains Emacs Lisp files and add-ons. This layout is a bit confusing, and there are no index or README files to indicate what is where.

For Windows users who don't already have Emacs loaded, the second chapter covers installing and configuring Emacs under Windows. I found the book's suggested directory layout confusing, so I didn't attempt to create it on my Windows 98 system. Instead, I just created an .emacs file in the root directory of my C: drive.

The book implies that the CD will automatically start installing Emacs when inserted in a Windows machine, but it doesn't appear to be set up for this. I ended up installing Emacs using the SETUP.EXE program, and it ran correctly after installation. The CD also contains Zip files if you want to install Emacs manually.

Though the installation instructions are somewhat unclear, the book did provide clear instructions on setting up printing and associating files with Emacs under Windows.

The rest of the book is well-arranged, starting with the basics of Emacs and leading up to its advanced features. I found that each chapter presented

enough information to increase a new user's knowledge without drowning them. Examples are given for almost every topic brought up. Each chapter ends with a summary of the material presented followed by a short Q&A section. Finally, a few exercises are given for readers to attempt on their own.

The early chapters cover basic editing, working with buffers and searching. An explanation of the Emacs help and configuration system is in the middle of the book. Next, the special editing modes for C, Java and LaTeX files are discussed. There is a chapter covering the use of Emacs' version-control functions, along with compiling and debugging programs. There are two chapters on Gnus (a news and mail reader that runs under Emacs). Beginning Emacs Lisp, an introduction to key bindings, and how to use add-ons comprise the final chapters of the book.

The book makes several references to add-ons included on the CD, especially the sams-lib.el file which contains a number of specialty functions created by the authors. Some attempt is made to provide brief descriptions and installation instructions for the add-ons, but only a few from the CD are actually covered in the book.

I am amazed at just how powerful Emacs is, and this book does a good job of introducing a new user to the sheer number of features available in it. The authors also did a very good job of choosing the right amount of material to allow someone to use Emacs without overloading them. I wish they had spent a little more time on such things as regular expressions and the Windows installation directions, though.

Unfortunately, the large number of errors contained in the text along with the confusing layout of the accompanying CD almost keep this book from being usable by its target audience: beginners. I would expect that the next release will have corrected these problems, and then it will be an invaluable addition to the beginning Emacs user's bookshelf.

Ralph Krause (rkrause@netperson.net) lives in southeastern Michigan, where he runs a small computer consulting business. In addition to trying to earn a living, he enjoys reading and playing with his dogs, Dakota and Purdy.

Archive Index Issue Table of Contents

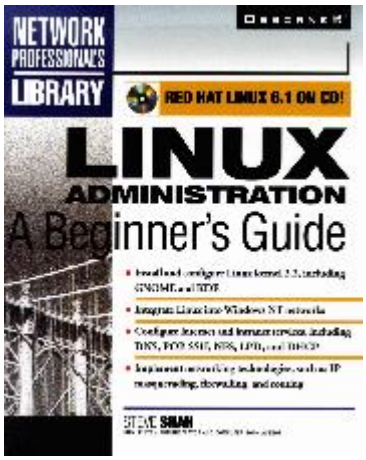
Advanced search

Linux Administration: A Beginner's Guide

Harvey Friedman

Issue #75, July 2000

Although it does have “beginner” in the title, this is not a book for someone entirely new to computers.



- Author: Steve Shah
- Publisher: Osborne McGraw-Hill
- URL: http://www.osborne.com/
- E-mail: customer_service@mcgraw-hill.com
- Price: $39.99 US
- ISBN: 0-07-212229-3
- Reviewer: Harvey Friedman

Although it does have “beginner” in the title, this is not a book for someone entirely new to computers. Rather, it was written for one who might have prior experience as systems administrator for another computer system, and is finally ready to try Linux. In particular, the author compares it to Windows NT, with some lovely diagrams showing 1) authentication, 2) network, 3) boot process and 4) shutdown process. I used these comparison diagrams to learn about NT servers (assuming they were correct) because my only previous experience with NT was with the NT workstation.

Enough about NT, however. This book was a delight to read. Each chapter uses the classic teaching organization of an introduction telling the reader what will be covered, the coverage itself, and then a brief summary outlining what was covered, plus a list of books for further reading. The overall structure consists of five parts: Installing Linux, Single Host Administration, Internet Services, Intranet Services and Advanced Linux Networking.

The inside front cover of the book contains a “functional menu” listing many common tasks that a system administrator has to do and a reference to the chapter containing it. This implies that each chapter is self-contained, although many of the later chapters require information from earlier ones. I recommend reading it from cover to cover. Also included with the book is a CD-ROM containing the “Publisher's Edition” of the Red Hat Linux 6.1 distribution. What this means is that the reader has a stripped-down version of the distribution. URLs for Red Hat and other useful sites (such as where to download “ssh”) are provided, so no one should be too peeved.

Shah gives good descriptions of using RPM to grab packages to install on one's working system, as well as complete illustrations of downloading a tarball, unzipping (if necessary), untarring and moving to a new directory. He then leads the reader thru the process of finding a **Makefile**, configuring, compiling and installing the package.

The description of setting up anonymous FTP is the most complete I can recall reading in recent memory. It is not just a cookbook recipe to follow, but also includes reasons to do things. Shah even contrasts setting up FTP via NT and discusses the relative security of each.

The Advanced Linux Networking section seemed too short. It covers IP aliasing, packet filters, and ipchains. After discussing the theory behind these, Shah gives concrete examples of how one might set them up, including filtering out the “ping of death”.

I did stop counting typos after a dozen or so, but most of them will probably not detract from the reader's understanding. One of the most annoying errors was on page 222; the text describing DNS did not agree with Figure 12-1.

I think this book is well-written enough in explaining what is behind much of Linux that it could likely reduce many of the FAQs on Usenet newsgroups—that is, if everyone who had installed or wanted to install Linux would read it. Of course, most Linux newbies should read the HOWTOs, as well.

Overall, I think so highly of this book that I rate it in a class with Nemeth, et al.'s *UNIX System Administration Handbook*. Or maybe I've reviewed too many “...

for Dummies" books. Well, I *have* reviewed enough of them and, yes, this book is good.

Harvey Friedman can be reached via e-mail at fnharvey@u.washington.edu.

Archive Index Issue Table of Contents

Advanced search

LINUX JOURNAL

AIPS and Linux: A Historical Reminiscence

Patrick P. Murphy

Issue #75, July 2000

The Astronomical Image Processing System looks at the sky using the radio wave section of the electromagnetic spectrum.

It was a dark and stormy night. Well, maybe not—but there was plenty of inclement weather on a late fall day in 1993, in central Virginia. I was waiting with eager anticipation for the arrival of a third-year college student from Virginia Tech, who had some precious cargo in his possession: a working version of our flagship application, AIPS, that he had ported to a brand-new and relatively obscure operating system. He also brought something that would revolutionize the basis for computing at my workplace: a QIC-60 cartridge containing a very early, well-hacked and customized version of the SLS distribution of Linux. I believe it was based on kernel 0.99.12, although it could have been 0.99.11.

I work at the Headquarters of the National Radio Astronomy Observatory (NRAO) in Charlottesville, Virginia. Our mission is to provide world-class facilities for astronomers to observe the universe through radio waves. Despite what most people think, we don't *listen* to anything. Instead, we gather signals, grind them through some impressive hardware and software and end up with pictures of what the sky would look like if we could *see* in the radio part of the electromagnetic spectrum.

On that particular day in 1993, my main duty was systems programming and world-wide installation support for AIPS, the Astronomical Image Processing System. This is NRAO's main application for processing the signals from our telescopes. It is a combination of command-line interpreter, graphical image display and a large (300+) set of programs or “tasks” that perform more-specialized functions. These functions range from simple bookkeeping tasks to serious number-crunching algorithms such as deconvolution, maximum entropy, Fourier transforms and more.

Figure 1. The Very Large Array in its Most Compact Configuration

What is NRAO?

Established in the waning years of the 1950s, the National Radio Astronomy Observatory is a facility that now comprises telescopes at several widely scattered sites across the United States. Headquartered in Charlottesville, the main instruments it operates are the Very Large Array (VLA), located 50 miles west of Socorro, New Mexico; the Green Bank Telescope (GBT, due to be commissioned in August 2000) in West Virginia, and the Very Long Baseline Array (VLBA) which has ten large telescopes in locations ranging from Hawaii to New Hampshire, and Washington State to the U.S. Virgin Islands. Visitor centers are located in Green Bank and at the VLA. Anyone who has seen either the movie *Contact* or *2010* has seen the VLA; both those movies were shot partially on location out on the dusty plains of San Augustin at the VLA site. And no, there isn't a canyon next to the VLA; that was artistic license on the part of the *Contact* directors! The canyon is actually Canyon de Chelly in neighboring Arizona.

In addition, the NRAO is working with several European partners in a project to create another array, this time for higher-frequency (millimeter wavelength) radio waves. This venture will comprise 64 moderate-size movable telescopes to be located on a 15,000+ foot high plain called Atacama in the remote Chilean Andes mountains. Currently in the design phase, this Atacama Large Millimeter-wave Array (ALMA) promises to open more new frontiers in astronomy.

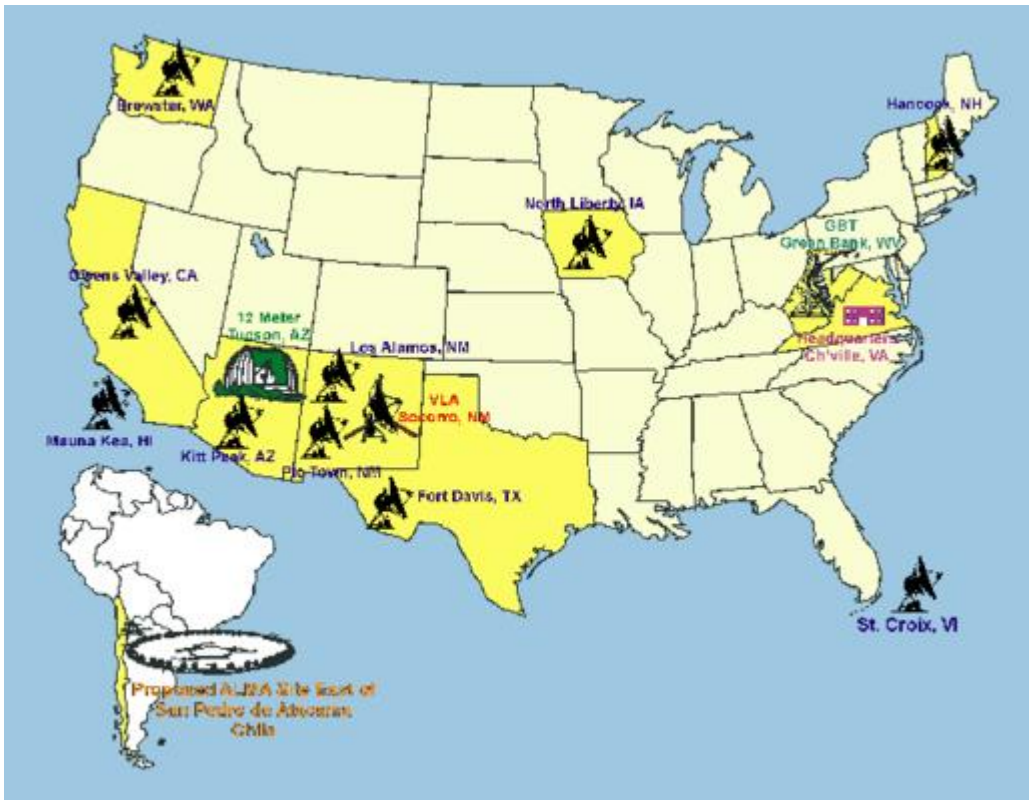


Figure 2. NRAO Facilities

As can be seen, we use arrays of radio telescopes in most of our instruments. Working together, they can give a much better picture than if they act alone. The technique used to correlate the signals from all the telescopes in an array is called *Aperture Synthesis* and is used at all our current arrays. This technique is the single most important *raison d'etre* for AIPS.

How Does Aperture Synthesis Work?

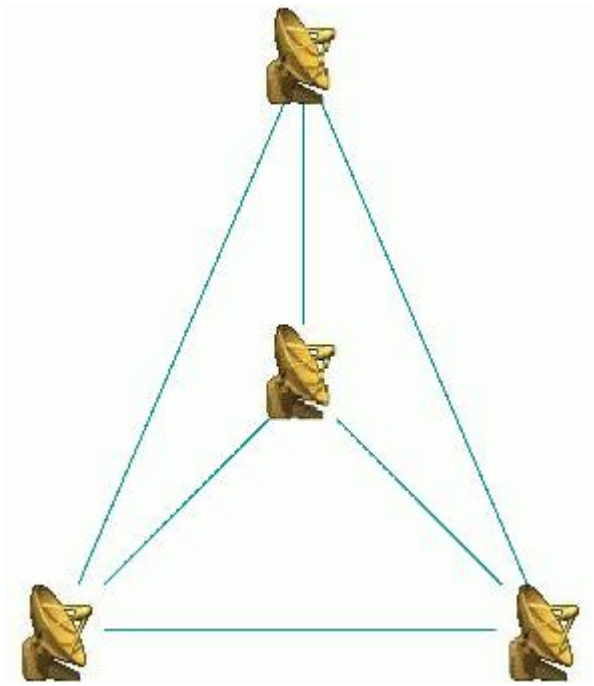


Figure 3. The Six Unique Baselines in a Four-Antenna Array

Without going into excessive technical detail, here's how aperture synthesis works. Several radio telescopes (antennae) are used, and the signals from each of them are captured, sent via wire or microwave “waveguide”, digitized and fed into a special-purpose computer known as the “correlator”. This one-of-a-kind computer takes the signals from each antenna and correlates them with all other signals. The result is a series of signals called “baselines”, as they represent the correlation of the signal from the ends of a baseline joining the two antennae. For a set of *n* antennae, you get *n*x*(n-1)/2* baselines. If *n* is 4, for example, you get (4x3)/2 or 6 baselines (see Figure 3). You don't correlate an antenna with itself, and you don't count antenna.1-antenna.2 and antenna.2-antenna.1 baselines as separate.

When the signals are correlated like this, you end up with a set of “visibilities” that is almost like a Fourier transform of the signals as they arrive at each antenna. Basically, each baseline gives a point on the “U-V” plane, and gridding, interpolating and transforming the result can produce an actual image. The truly cool thing is that this image has the same resolution as if it were taken with a telescope the same width as *the whole array*. So the VLA in its “A” configuration, with over 15 miles separating the farthest-flung telescopes, is the equivalent of a 15-mile-diameter telescope. It gets better; with the VLBA, we've got the better part of the Earth's diameter between the antennae in Hawaii and St. Croix. As many of you may know, in astronomy it's diameter that counts when you want to “zoom in”. Instead of measuring angular distances in seconds of arc (the moon appears to be about 30 *minutes* of arc, or 1800 seconds of arc across), we measure things in milli-arc seconds with the VLBA.

AIPS—A Brief History

AIPS is a package to support the reduction and analysis of data taken with radio telescopes. It is most useful for arrays of telescopes like the VLA and VLBA. In the past few years, it has also been used successfully for “Space VLBI” (very long baseline interferometry) in conjunction with a small telescope on a Japanese satellite (HALCA or VSOP).

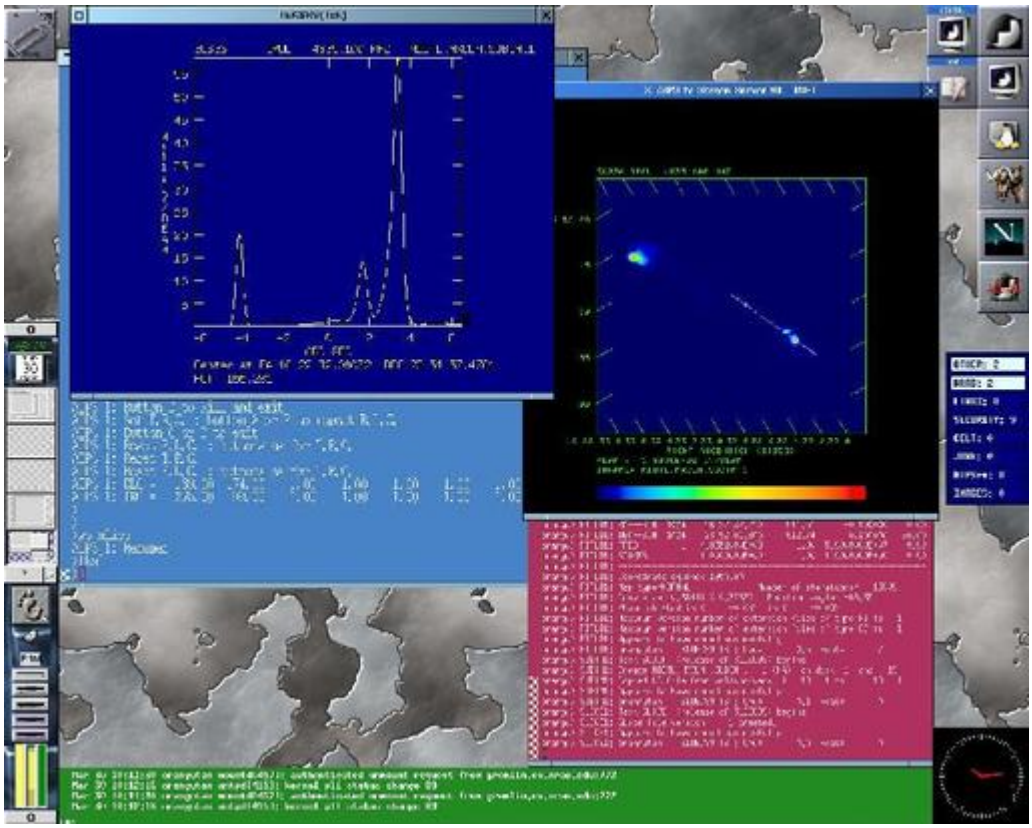


Figure 4. Screen Shot of a Typical AIPS Session

AIPS is what most of us would now describe as “legacy software”, having been originally coded in a truly ancient dialect of FORTRAN (predating even the venerable FORTRAN IV). AIPS now uses FORTRAN 77, although it has been digested successfully by at least one FORTRAN 90 compiler.

A Modcomp computer in Charlottesville was the first system to actually host a working AIPS system, and it quickly spread to a guest UNIX system hosted on an IBM 360 mainframe. From there, it spread in the early 1980s to VAX/VMS systems, often with an attached floating-point systems array processor (this peculiar device was the moral equivalent of the 80387 floating-point accelerators that some old-timers may remember being part and parcel of many 386 systems). In the late 1980s, UNIX came back into AIPS' universe in a big way, first with the Sun-3 series of Motorola-68020-based systems and then with a series of others, including Cray (Unicos), Convex and Alliant systems.

By the time the 80s were winding down, the dominance of VMS in the AIPS universe was being seriously questioned. Performance on new upstarts like Sun was starting to challenge their price/performance ratio, and the first SPARCstations totally blew them away. In the early 1990s, AIPS moved to a smorgasbord of UNIX variants: AIX, Stardent (briefly), Ultrix, HP-UX, SGI's Irix and DEC (oops, Compaq) OSF1. A port to an IBM 3090 was attempted, but failed due to accuracy problems with the non-IEEE floating-point format thereon. In the middle of this flurry of activity, the port to Linux by Jeff Uphoff was made.

The Blacksburg Connection

In the fall of 1993, NRAO got a query from a radio astronomer at Virginia Tech in Blacksburg, requesting permission for one of his students to copy AIPS to his PC for an attempted port to a new system called Linux. (At that time, AIPS was still proprietary code, released to non-profit organizations under a rather cumbersome license and user agreement; this changed later.) Polite skepticism was the immediate reaction of most people then in the NRAO AIPS group, but we thought it would be an interesting exercise. We had been giving some thought to the issue of running AIPS on personal computers at the time, but had not pursued it.

A scant two weeks later, I received a technical support call from the same Jeff Uphoff, where he alluded to some difficulty in compiling one routine using **f2c**/**gcc** (remember, this was 1993, before **g77** came to prominence). As it turned out, the problem was minor, and Jeff had successfully completed the whole port, even to the point of running the benchmark "Dirty Dozen Tasks" or DDT suite. This benchmark takes about an hour to run on a legacy SPARCstation 1 or SPARC IPX. On Jeff's poor little "uppieland" 386, it took over a day, but it ran to completion with acceptable accuracy results. We were equally impressed with his ability to print PostScript output on his little DeskJet 500 printer; our HP and QMS printers at the time all cost many thousands of dollars.

Needless to say, this feat and the contribution of the modifications necessary for Linux made a big impact at NRAO. Within a month, we had invited Jeff to visit and install a copy of Linux with AIPS running on it on a system in our lab. That was the dark and stormy evening referenced in the introduction above, and it saw both of us huddling over a (then state-of-the-art) Gateway 486/66 system, boot floppies in hand, busily preparing it for AIPS. Within a couple of hours, we had the system installed, looking for all intents and purposes like a Sun, even down to the Open Look window manager, and busily munching away on AIPS data. We benchmarked it at about half the speed of the Sparc IPX systems we had at the time. Seeing that the 486 cost a lot less than an IPX, this got our attention once more.

Faster and Faster, and Make it GNU

Within a few months of the original port, NRAO had Jeff Uphoff on its payroll, and the race was on to improve the performance of AIPS on Intel hardware. In the process, the NRAO Charlottesville Computing Division ended up with many Linux machines performing server duties, and several programmers and scientists volunteered for (in some cases, demanded) Linux desktop systems. The operating system was also spreading like wildfire on many of our home systems as an adjunct to, or in some cases a replacement for, that OS from Redmond.

However, it took the use of the EGCS version of the GNU g77 FORTRAN compiler to push the Intel/Linux platform to the forefront of the Radio Astronomy community. In 1995, using EGCS version 1.0.2, we (well, okay, it was Jeff again; why isn't he writing this article?) succeeded in getting AIPS to build under g77. This improved the AIPSMark (our benchmark, defined as 4000 divided by the elapsed time in seconds to run the DDT on our large dataset; bigger AIPSMarks are better and a Sparc IPX is 1.0) on a Pentium Pro 200 from 3.3 to about 5. With further coaxing via aggressive use of optimization parameters, the resulting AIPSMark went over 6. In this fell swoop, the price/performance curve that was previously occupied by Sun, IBM, DEC and HP was shattered once and for all. By 1998, NRAO was ordering Linux/Intel desktops as the workstation of preference for the scientist in place of SPARC Ultra systems. In 1999, Linux started to edge out the high-performance public workstations such as Alpha and high-end SPARC; our star performer in Charlottesville is a Pentium III Xeon 550MHz system with an AIPSMark of over 21. This is still not close to the stratospheric performance of top-of-the-line HP and Alpha systems (both around 34 AIPSMarks), but in price/performance terms, it's still a knockout.

During this time, another significant change came about. All this exposure to copylefted code was taking its toll on us. As mentioned earlier, AIPS was originally released under a restrictive user agreement that prohibited redistribution and was unpalatable or even unacceptable to some in our own astronomical community. Not only that, but the administrative costs associated with it were a burden.

Thus, several of us started a rather vigorous campaign to shift AIPS over to the GNU (Free Software Foundation's) General Public License on its next periodic release. There was virtually no opposition to the principle, and after a brief review by our business division, we were given the green light. Applying the three-paragraph copyleft statement to tens of thousands of files was no easy feat, but we managed to automate it. (Shell scripts and Perl scripts can do almost anything.) The distribution of AIPS took off at this point, and nowadays if you encounter a radio astronomer anywhere in the world, chances are she will have an AIPS CD-ROM somewhere or have it installed on her system.

What's Left

In early 1998, as Jeff left for greener pastures (a formerly secretive company called Transmeta), there were two major obstacles left that stood in the way of total world domination by the Intel/Linux platform in the AIPS community. These were file locking on NFS-mounted file systems, and support for files larger than 2GB. With the advent of kernel-based NFS in version 2.2, the first issue is now moot. Jeff wrote the Linux **statd**, one of the two halves of the statd-lockd pair that governs all NFS file locks mostly at NRAO, partially at Transmeta.

As for the 2GB limit on file size, there is now hope on the horizon on that front, too. With SGI's promised release of their XFS code, it's likely that this journaled file system—which offers large file support in its original Irix environment—will smash this barrier. It should also provide a more robust file system, and hopefully one that won't take quite so long to **fsck** a 20GB partition.

Before he left, Jeff did contribute to the follow-up project to AIPS, called AIPS++. This large project, now coming to maturity, has fortunately followed in its parent's footsteps by adopting both the General Public License and Intel/Linux as one of its leading platforms.

Learn More About It

If you who would like to get your hands on the software, point your browser at NRAO's main page and follow the link to “Software”. Be warned, though: it is not a small package, needs about 400MB of disk space, and has a non-trivial install procedure. It's also most useful if you want to analyze data from the VLA; if you want a simpler program, look at Bill Cotton's FITSView instead.

At many serious astronomy sites, you will find the dominant format of images and data to be FITS, the flexible image transport system. This format was invented around 1980 by a consortium of NRAO and other scientists so they could share data. The popular **xv** viewer supports the simpler versions of this format.

Resources

Pat Murphy (pmurphy@nrao.edu) joined NRAO in 1984 and has worked at three of the four main sites since then. He now runs the Computing Division at their Charlottesville HQ and acts as the local Linux evangelist, webmaster, security guy, backup system administrator and more. In his spare time, he goofs off on his personal web page goof.com/~pmurphy/maze.shtml, provides tech support for his wife Kim and her newly published e-book, and pretends he knows how to train Magic, a Belgian sheepdog.

Archive Index Issue Table of Contents

Advanced search