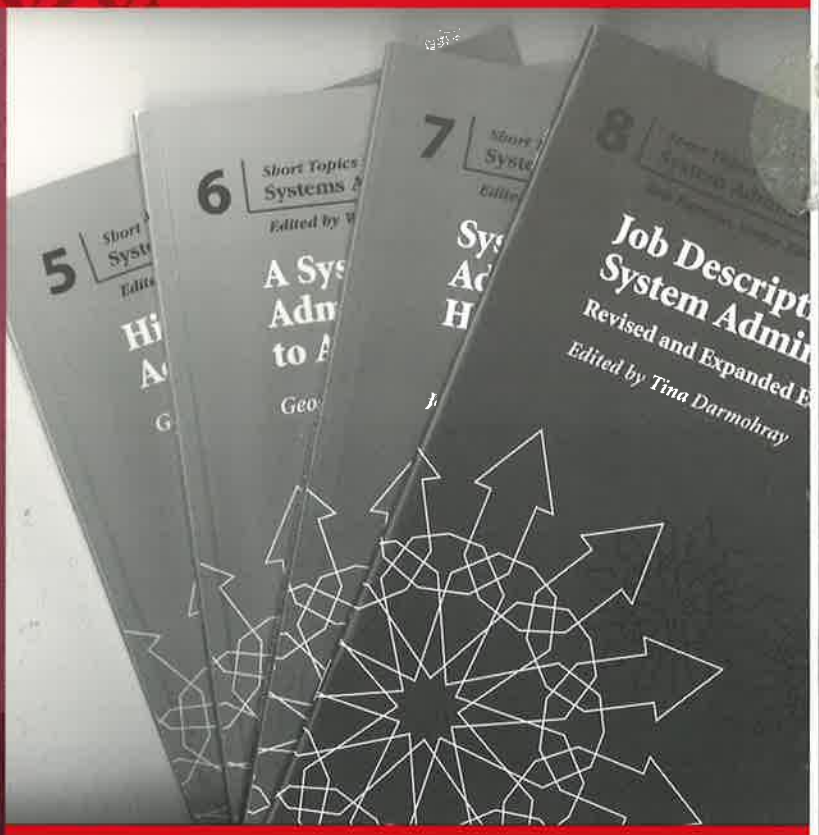
;login:

THE MAGAZINE OF USENIX & SAGE

April 2002 • Volume 27 • Number 2

inside:



USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

USENIX Upcoming Events

The 3rd International SANE Conference

Organized by NLUUG
Co-sponsored by USENIX and the NLnet Foundation.

MAY 27–31, 2002
Maastricht, The Netherlands
http://www.sane.nl

2002 USENIX Annual Technical Conference

JUNE 9–14, 2002
Monterey, California, USA
http://www.usenix.org/events/usenix02/

2nd Java™ Virtual Machine Research and Technology Symposium (JVM '02)

AUGUST 1–2, 2002
San Francisco, California, USA
http://www.usenix.org/events/jvm02
Notification of acceptance: March 12, 2002
Camera-ready final papers due: May 28, 2002
Registration materials available: April, 2002

11th USENIX Security Symposium

AUGUST 5–9, 2002
San Francisco, California, USA
http://www.usenix.org/events/sec02

16th Systems Administration Conference (LISA '02)

Sponsored by USENIX & SAGE

NOVEMBER 3–8, 2002
Philadelphia, Pennsylvania, USA
http://www.usenix.org/events/lisa02
Extended abstracts due: April 29, 2002

Internet Measurement Workshop 2002

Sponsored by ACM SIGCOMM and co-sponsored by ACM SIGMETRICS and USENIX

NOVEMBER 6-8, 2002
Marseille, France
http://www.icir.org/vern/imw-2002/
HARD submission deadline: May 10, 2002
Notification: June 28, 2002
Camera-ready copy due: August 9, 2002

5th Smart Card Research and Advanced Application Conference (CARDIS '02)

NOVEMBER 20-22, 2002
San Jose, Califonia, USA
http://www,usenix.org/events/cardis02
Submissions due: June 24, 2002
Acceptance notification: August 12, 2002
Camera-ready final papers due: September 23, 2002

2nd Workshop on Industrial Experiences with Systems Software (WIESS '02)

Sponsored by USENIX
Co-sponsored by ACM SIGOPS & IEEE TCOS

DECEMBER 8, 2002
Boston, Massachusetts, USA
http://www.usenix.org/events/wiess02
Submissions due: July 15, 2002
Notification to authors: August 19, 2002
Camera-ready final papers due: September 30, 2002

5th Symposium on Operating Systems Design and Implementation (OSDI '02)

Sponsored by USENIX
Co-sponsored by ACM SIGOPS & IEEE TCOS

DECEMBER 9–11, 2002
Boston, Massachusetts, USA
http://www,usenix.org/events/osdi02
Submissions due: May 17, 2002
Notification to authors: August 5, 2002
Camera-ready final papers due: October 7, 2002

2nd Conference on File and Storage Technologies (FAST '03)

MARCH 2003

4th USENIX Symposium on Internet Technologies and Systems (USITS '03)

MARCH 26-28, 2003
SEATTLE, WASHINGTON, USA
http://www,usenix.org/events/usits03
Submissions due: Sept. 16, 2002
Notification of acceptance: November 8, 2002
Camera-ready final papers due: January 17, 2003

For a complete list of future USENIX events, see **http://www.usenix.org/events**

contents

;login: Vol. 27 #2, April 2002

;login: is the official magazine of the USENIX Association and SAGE.

;login: (ISSN 1044-6397) is published bimonthly, plus November, by the USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

$50 of each member's annual dues is for an annual subscription to *;login:*. Subscriptions for nonmembers are $60 per year.

Periodicals postage paid at Berkeley, CA, and additional offices.

POSTMASTER: Send address changes to *;login:*, USENIX Association, 2560 Ninth Street, Suite 215, Berkeley, CA 94710.

Cover: Some recent issues in the Short Topics in System Administration *series. See* http://www.usenix.org/sage/publications/short_topics.html

motd

by Rob Kolstad

Dr. Rob Kolstad has long served as editor of *;login:*. He is also head coach of the USENIX-sponsored USA Computing Olympiad.

kolstad@usenix.org

MEMBERSHIP, PUBLICATIONS, AND CONFERENCES

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: 510 528 8649
FAX: 510 548 5738
Email: *office@usenix.org*
login@usenix.org
conference@usenix.org
WWW: *http://www.usenix.org*

Nomenclature: How Hard Can It Be?

I am writing this column in the fervent hope that people will pay attention to the words that surround us. Our economic system, fabulous as it is (and I am its biggest fan), has regrettably promoted a sort of marketing-speak that has so obfuscated some communications that I sometimes find myself at a loss trying to figure out if anyone can understand certain messages being shared.

Consider this one from an employee of a financial company: "Without Microsoft, I couldn't do my job." This person evidently believes that everything from the spacebar to the network connection somehow falls under the auspices of Microsoft. This would include (but not be limited to): the really fast microprocessor (probably an Intel or AMD component - engineering that we can show the aliens when they arrive), a graphics card (100x faster than available five years ago), a monitor, an application that might not even belong to Microsoft, DRAM from overseas, power from the local utility company, etc. etc. Yet who gets the credit? The operating system company: "Where do you want to go today?" Wow.

The headline blares: "Linux Enables Successful Space Shuttle Mission." Yes, someone on the mission had a laptop running Linux. Was that *the* big key to the mission's success? I doubt it.

Or how about this printer claim: "Printer rated at 8ppm color, 12ppm black and white." A rational person could interpret this as, "Oh, I reckon I'll probably see 6-8 pages per minute when I print color output on this printer." Bzzzt. Wrong answer. What it means is: "We will given you a written guarantee that this printer will never, ever, ever exceed 8ppm color or 12ppm black and white no matter under what circumstances." Has any printer of this model ever printed 8 color pages in any minute? Probably not. But the limiting factor is not the page feeding mechanism!

"Fumblebum's color text editor allows me to edit twice as fast as my old editor." Where did we get this word "allow" in the context of computer software? Programs "enable" (and security programs "permit" – sometimes even "allow"). I fear that simple word usage like this makes people think that somehow software or computers or somesuch *grant them permission* to perform a task. Who is the master? Who is the slave?

This same sort of marketing jargon permeates our industry at all levels. One of my favorites is the "Designated Guru;" Bill Joy sometimes fills this niche at Sun Microsystems. You can read history books that discuss how "Bill rewrote the entire UNIX operating system from scratch." I respect Bill's technical ability as much as the next guy, but I don't reckon he wrote the OS from scratch.

Does the technological consuming public really need the kind of hyperbole, misdirection, and icon-worshipping that advertising would lead us to believe? Check out any trade magazine that prints in four colors. Look at the ads. Are these ads aimed at super-technical types? I think not. Read the articles. "Drimkel Corp credits its entire successful 3rd quarter to the Excel spreadsheet macro designed by Joe Glerky." I just don't think it helps. I usually try to provide some sort of prescription to address this problem. The only thing I can imagine is that we should all try to use the proper words when we discuss technology. We should try to have a slightly broader understanding than "Bill Gates has rewritten Windows to be the world's fastest Web Server." Maybe we could go for just one more level of detail.

I'm going to try; I hope you will, too.

point/counterpoint

At Who's Expense?

by Tina Darmohray

Tina Darmohray, co-editor of *;login:*, is a computer security and networking consultant. She was a founding member of SAGE.

tmd@usenix.org

The local DA has expressed an interest in talking to me about a break in to Lawrence Livermore National Laboratory's computer systems. Apparently she is looking to gain insight about typical Laboratory computer security and incident response from folks who have worked at LLNL. As her contact went into more detail, I realized that I knew both the "hacker" and the "hackee."

The break in incident in question was reported in September of 2000:

"A 21-year-old Minnesota computer employee was arrested at home on Monday for allegedly hacking into the Lawrence Livermore National Laboratory. Benjamin Troy Breuninger, known on the Internet as "konceptor," had not jeopardized the lab's classified nuclear research material, but he allegedly accessed the lab's administrative information, causing about $50,000 in damage. It appears that the attack was random and that Breuninger didn't have any personal gripe with the lab."

Once they mentioned the names "Ben" and "Konceptor," I immediately remembered that I had met Ben in October 1998 at a conference in Orlando, FL. I taught a tutorial with Phil Cox at that conference. In the evening session following our tutorial Phil was the "White Hat" and Ben was the "Black Hat" in a "Hackers and Defenders" session that had been arranged by the conference.

We didn't know Ben prior to the session, but when we saw him it was immediately obvious that he was "just a kid." Despite that, he held his own technically. The depth of his knowledge was impressive. He clearly had a handle on the fundamental workings of the Internet. He was well versed in such areas as DNS, ports and services, and a wide array of protocols and programming languages. He described in detail how he and his peer group spent their weekend nights: his phone buddies would hijack a phone line off an office building and they'd run a long wire into the bushes where Ben would sit w/his laptop and do the "cracking." Many folks from the audience told Ben they didn't like what he and "his kind" were up to. Ben's stance was that they were just having fun, learning, and, bottom line, not hurting anyone.

After the presentation Phil and I cornered Ben. We learned that he was indeed just a kid: old enough to drive, but I don't think yet old enough to vote. He said he'd never been out of his home state, and that this was his first trip on an airplane. It was obvious that while Ben was well versed in the Internet, he wasn't worldly in any other way. During our conversation, Phil and I tried our best to persuade Ben to apply his knowledge in legal ways: college, career in system administration.

Looking back, I have to wonder about the message Konceptor's all-expense-paid trip to Orlando sent to him and whether it amounted to unintentional exploitation. It's easy to understand why Konceptor didn't take our advice. Why should he? His hacking hobby had just landed him his first trip on an airplane to a conference where a room full of adults listened to what he had to say. At his age, that had to be huge for him, as well as among his friends. From his perspective, his hacking had paid off big time and he was being taken seriously. Hindsight suggests Ben's perception, as well as "his own good" should have been more carefully considered.

Counterpoint

by Rob Kolstad

Ben is surely walking the wrong path, especially for a 21 year old who absolutely should know better, having been told so repeatedly. Of course, he is not the only one. His life is now going to be dramatically more complex and challenging as he deals with both the legal system and lawyers while learning that our country really does have absolute rules that, when broken, can have severe penalties.

The main questions are: was his trip to the security conference exploitative? Was it a reward for "being bad"?

Tina mentioned the black hat/white hat presentation at which the participants were squarely put into roles ("good" vs "bad" – or maybe even "good" vs "evil"). Just as Ben had not been out of his state, similarly most participants had not seen a "hacker on the hoof." Tina mentions his apparent lack of worldliness and it was extremely apparent in any dealings with Ben.

I think it can be argued that the opportunity to come to the conference to share – and to meet the security community – had a huge potential for good. This clearly naive kid had the opportunity to see, meet, and greet professional security people and learn all about the potential for employment and gratification in that part of our economy. I can imagine no better way to stimulate someone to move into a more productive role than "system cracker for fun." I know I spoke to him one-on-one as did a myriad of other attendees. I imagine (without direct knowledge) that the messages he was sent were clear and consistent.

As happens often, Ben made up his own mind, unswayed by the 2,000-fold members of the "opposition". He will learn first-hand that "just having fun" is perceived by others in a different way. Some people will never be able to participate in our society in a reasonable way; more's the pity.

the tclsh spot

by Clif Flynt

Clif Flynt is president of Noumena Corp., which offers training and consulting services for Tcl/Tk and Internet applications. He is the author of *Tcl/Tk for Real Programmers* and the *TclTutor* instruction package. He has been programming computers since 1970 and a Tcl advocate since 1994.

clif@cflynt.com

This article starts a short series describing Tk techniques to make applications more professional looking.

When I'm doing a quick and dirty Tk GUI, I usually just use pack or grid to display the widgets in a usable manner. I don't worry about resizing the windows, providing hints, or any of the other features that I consider minimal requirements in a commercial package.

As soon as this package gets shown to a client, my lapses become obvious and embarrassing.

These features are actually pretty easy to put into a Tk application, and the nice thing is that they can be added after the GUI layout has been tuned. This helps keep the code small and manageable during the main rework stage of development.

In this article, I'll describe how to configure the geometry managers and widgets to allow individual widgets to be resized when the main window is resized.

Tk uses a paradigm similar to other windowing systems in that generating and displaying graphic objects are two separate events. The graphics objects are created with Tk widget commands like label, button, entry, text, and canvas. Mapping the objects onto the display is handled with one of the geometry managers – place, pack, and grid (and, of course, whatever the system window manager is).

Tk's three geometry managers provide different levels of abstraction in defining how a display should appear. The place command gives you the most control and requires the most thought to use. The pack and grid commands let your script describe the layout in more abstract terms, and the geometry manager will define the exact positions of the widgets.

The place command takes arguments that describe the location of each widget to be displayed and the size of the widget. The syntax is:

Syntax: place *.widgetName ?-option1 value1? ?-option2 value2? ...*

place	Place a widget at a particular location in a frame.	
.widgetName	The name of the widget to be placed.	
-option value	Key/value pairs that define the behavior of the widget being placed.	
	-x/-y pixels	Describe the location of the widget in pixels, numbering from the upper-left-hand corner.
	-relx/-rely float	Describe the location of the widget as a fraction of the size of the window. 0.0 is the left (for -relx) or top (for -rely) edge, and 1.0 is the right or bottom.
	-anchor location	Tells whether the widget should be anchored on the north, south, east or west edge (nsew), or center. By default, this is nw, the upper-left corner.
	-width/-height size	An integer that defines the size of the widget in pixels. By default, this is the natural size for the widget.
	-relwidth/-relheight float	Defines what fraction of the parent window should be devoted to this widget.

Using the place command, it's fairly easy to make a display that would have two text widgets, and corresponding labels above them, and have the widgets expand to be wider when you resize the window:

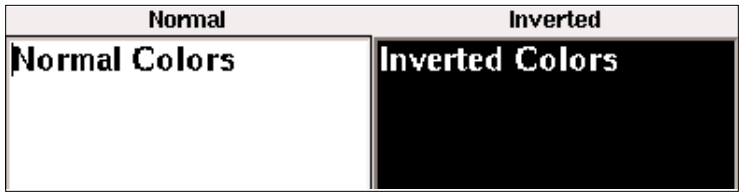
```
label .l1 -text "Normal"
label .l2 -text "Inverted"

text .t1 -background white -foreground black -font "helvetica 16 bold'
text .t2 -background black -foreground white -font "helvetica 16 bold"

place .l1 -x 0 -y 0 -relwidth .5 -height 15
place .l2 -relx .5 -y 0 -relwidth .5 -height 15

place .t1 -x 0 -y 15 -relwidth .5 -relheight .99
place .t2 -relx .5 -y 15 -relwidth .5 -relheight .99
```

The display for this code would resemble the window below. When the window is resized, the text widgets would expand or shrink, while the labels would stay 15 pixels tall.



For simple displays like this, or free form displays (perhaps a simulation of a car instrument panel), place works well. However, if you change the font in the labels, you'd have to change the height of the labels and the upper location of the text widgets. If the display were complex, tuning the appearance could become very tedious.

The pack command doesn't support locating widgets at specific coordinates. The pack command allows you to declare that a widget should be as close as possible to one edge or another of the window it's contained in.

Syntax: pack *.windowName ?-option1 value1? ?-option2 value2?*

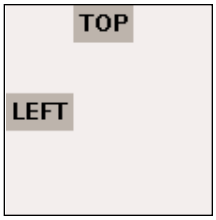
The options for pack include:

-expand boolean	If this is true (true, 1 or Y), then leftover space after the windows have been placed will be given to this window. If multiple windows are true for being expanded, the space will be divided evenly.
-fill x/y/none/both	If the -expand option is true, this tells the geometry manager that this window should expand in the x, y, or both dimensions. By default the value is none.
-anchor location	Tells whether the widget should be anchored on the north, south, east, or west edge (n, s, e, w), or center of the available space. By default, this is center.
-side left/top/right/bottom	Tells the packer which side of the open space to place this widget against.

The packer can be conceptualized as starting with a large open square. As it receives windows to pack, it places them on the requested edge of the remaining open space.

For example, if the first window is a label with the -side left option set, pack would place the left edge of the label against the left side of the empty frame. The left edge of the empty space is now the right edge of the label, even though there may be empty space above and below this widget.

If the next item is to be packed -side top, it will be placed above and to the right of the first widget.



The following code shows how these two widgets would appear. Note that even though the anchor on the top label is set to west, it only goes as far to the west as the east-most edge of the first widget packed.

```
label .l1 -background gray50 -text LEFT
label .l2 -background gray50 -text TOP
pack .l1 -side left
pack .l2 -side top -anchor w
```

This makes it impossible to build a display like the one we built with place from windows that are first-level children of the primary window.

The trick to building rectangular GUIs with the place command is to use multiple frames.

A frame is a (mostly) invisible holder window that can be used to group other windows.

Syntax: frame .frameName ?-option value?

The options supported by the frame command include:

-height *numPixels*	Height in pixels.
-width *numPixels*	Width in pixels.
-background *color*	The color of the background.
-relief ?sunken?raised?	Defines how to draw the border edges – can make the frame look raised or sunken. The default is to not display relief borders.
-borderwidth *width*	Sets the width of the decorative borders.

Building a complex GUI with pack and frame can get confusing. When the display is not what you expected, and you can't quite figure out why, it's sometimes useful to use the -background or -relief options to make it obvious which frame is where.

Simple GUIs, however, are fairly easy with pack. For instance, to build the same GUI as was done with place, we could use two frames. One frame would hold the left-most label and text widget, while the other would hold the right-most label and text widget.

```
frame .left
frame .right

label .left.label -text Normal
label .right.label -text Inverted

text .left.text -background white -foreground black \
      -font "helvetica 16 bold" -width 5 -height 1
text .right.text -background black -foreground white \
      -font "helvetica 16 bold" -width 5 -height 1

pack .left.label -side top
pack .right.label -side top
pack .left.text -side top -expand y -fill both
pack .right.text -side top -expand y -fill both

pack .left -side left -expand y -fill both
```

```
pack .right -side left -expand y -fill both
```

One difference between the pack and place command is that with the place command, we could specify the percentage of the main window a widget could use, and Tk would scale the widget accordingly. With pack, Tk places the windows in their default sizes, and then expands them where it can.

The default size for a text widget is 80 columns by 23 lines – like an old-style monitor, or a default-sized x-term window. In order to allow pack to use small text windows, we need to define them with a small initial size, and let the pack algorithm expand them.

One advantage of this code over place is that we can change the size of the widgets (like defining a larger font in the labels) without having to recalculate the locations of the widgets.

The pack command is very useful for asymmetric layouts or simple GUIs. Using pack to arrange buttons in a frame is the easiest way to display a row of buttons.

However, it's difficult to make a gridded display like a spreadsheet with the pack command, and that's actually a very common pattern.

The third geometry manager is the grid command, which is oriented around a gridded, spreadsheet-like layout.

The grid command arranges the widgets in the rows and columns your script defines, and expands or shrinks the row/column to fit the largest widget in that row/column.

Like the pack command, the grid command can expand a widget to fit a cell, but it does not shrink them below their requested size.

Syntax: grid *widgetName -option1 value1 ?-option2 value2?* ...

The *grid* options include:

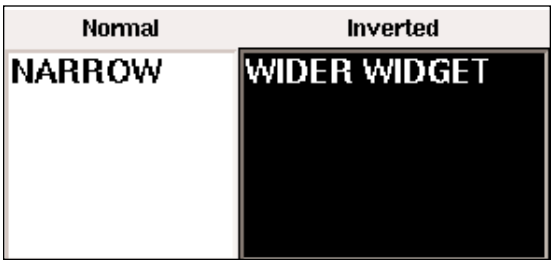
-column *columnNumber*	The column position for this widget.
-row *rowNumber*	The row for this widget.
-columnspan *columnsToUse*	How many columns to use for this widget. Defaults to 1.
-rowspan *rowsToUse*	How many rows to use for this widget. Defaults to 1.
-sticky *side*	Which edge of the cell this widget should "stick" to. Values may be n, s, e, w, or a combination of sides.

One shortcoming of the pack command is that it distributes extra space evenly between the windows that are allowed to expand. With the place command, you could define one widget to expand at a different rate from another.

Having widgets grow at different rates can be done with the grid command. The grid columnconfigure and grid rowconfigure commands let you define the behavior of a row or column in the display. There are a number of options including:

-minsize pixels	The minimum size for this widget, in pixels.
-weight integer	The weighting to assign to this column/row. The default value is 0; never change the size of the widget.

-pad pixels A number of pixels to use as padding around widgets in this column/row.



This code will create a pair of text widgets and labels, as we did above, but will make the right-hand side half again as wide as the left.

```
label .l1 -text "Normal"
label .l2 -text "Inverted"

text .t1 -background white -foreground black \
  -font "helvetica 16 bold" -width 5 -height 1

text .t2 -background black -foreground white \
  -font "helvetica 16 bold" -width 5 -height 1

grid .l1 -row 0 -column 0 -sticky news
grid .l2 -row 0 -column 1 -sticky news

grid .t1 -row 1 -column 0 -sticky news
grid .t2 -row 1 -column 1 -sticky news

grid columnconfigure . 0 -weight 2
grid columnconfigure . 1 -weight 3
grid rowconfigure . 1 -weight 1
```

To summarize the features of the three geometry managers:

place

- Complete control of widget location.
- Can overlay one widget on top of another.
- Script must define all details.
- Individual widgets can be resized.
- Can resize widgets at different rates.

pack

- Supports a space-filling abstraction for locating widgets.
- Widgets will not overlap each other; all widgets will be displayed.
- Individual widgets can be resized.
- Available space is apportioned to widgets evenly.

grid

- Defines widget layout as a grid.
- Widgets will not overlap each other; all widgets will be displayed.
- All widgets in a row or column can be resized.
- Rows and columns can resize at different rates.

In these examples, the windows were always wide enough to display all of the widgets. We added support for someone expanding a main GUI and expanding the appropriate widgets to look right, but not to shrink the display below the default minimum.

What do you do if there might be more elements to display than you've got space for on the screen? For instance, someone might want to run your application on the IBM Linux watch with an LCD that isn't wide enough to display a full button bar.

The obvious answer to this is to put a scrollbar on the edges of the primary window, so the user can scroll to the widgets they want to see.

The next Tclsh Spot will describe some tricks for making scrolled windows.

math library functions in C9X

by Glen McCluskey

Glen McCluskey is a consultant with 20 years of experience and has focused on programming languages since 1988. He specializes in Java and C++ perfor-mance, testing, and technical documentation areas.

glenm@glenmccl.com

We've been discussing some of the new features in C9X, the standards update to C. In this column we'll look at new math functions and, in particular, the philosophy behind the inclusion of the functions in the library.

Why So Many Functions?

There are lots of new math functions in C9X. One reason for the proliferation is that many functions have been generalized to work with float and long double types, in addition to the double type already supported. Here's an example:

```
#include <math.h>
#include <stdio.h>

const long double CONV_DEG_TO_RAD = 360.0L / (2.0L * 3.14159265L);

int main()
{
    long double d = sinl(90.0L / CONV_DEG_TO_RAD);

    printf("%Lg\n", d);
}
```

C has always had a sin() function, but now it also has sinf() and sinl() functions, which operate on float and long double types.

But this is only part of the picture. C9X also includes a cbrt() function, which calculates cube roots. Why is such a function needed? Why couldn't you simply say:

```
double res = pow(val, 1.0/3.0);
```

that is, compute the cube root by taking the 1/3 power of a value?

The answer is that you can, but cbrt() offers some advantages. One is that it will work on negative numbers: for example, -27.0, with a cube root of -3.0. Another advantage is that there may be a more efficient means of calculating cube root than pow(), or an algorithm that has better error properties. For example, when we run this program:

```
#include <math.h>
#include <stdio.h>

double get_cube_root1(double d)
{
    return pow(d, 1.0/3.0);
}

double get_cube_root2(double d)
{
    return cbrt(d);
}

int main()
{
    double d = 259.0;
    double res, diff;

    // get cube root using pow()

    res = get_cube_root1(d);
    diff = res * res * res - d;
    printf("diff using pow() = %g\n", diff);
```

```
    // get cube root using cbrt()

    res = get_cube_root2(d);
    diff = res * res * res - d;
    printf("diff using cbrt() = %g\n", diff);
}
```

we get the following result:

```
diff using pow() = -1.09746e-13

diff using cbrt() = -1.47105e-15
```

indicating that cbrt() has less error than pow(). Note also that the 1.0/3.0 we passed to pow() does not necessarily exactly represent 1/3, given that many fractions are not exactly representable in IEEE floating-point format.

Here's another example of why you might want to use a library function to perform an "obvious" calculation. Suppose that you're computing the square root of a sum of squares, like this:

```
sqrt(X*X + Y*Y)
```

C9X provides a function hypot() to perform this calculation. Why? Here's one example where it matters:

```
#include <float.h>
#include <math.h>
#include <stdio.h>

int main()
{
    // set up two constants equal to sqrt(DBL_MAX)

    double x = sqrt(DBL_MAX);
    double y = x;

    // compute square root of sum of squares

    double z = sqrt(x * x + y * y);
    printf("%g\n", z);

    // same computation using hypot()

    z = hypot(x, y);
    printf("%g\n", z);
}
```

The result of running this program is:

```
inf

1.89615e+154
```

x and y have values equal to sqrt(DBL_MAX). When these values are squared and then added, the result is an overflow – that is, infinity.

But hypot() works in this case. The C9X specification directly addresses the intermediate overflow issue:

> The hypot functions compute the square root of the sum of the squares of x and y, without undue overflow or underflow.

One final example. The library contains an fma(x,y,z) function, that computes x*y+z. Why would such a simple calculation require a library function? Once again, we look to the C9X specification for a clue:

> The fma functions compute (x*y)+z, rounded as one ternary operation; they compute the value (as if) to infinite precision and round once to the result format, according to the rounding mode characterized by the value of FLT_ROUNDS.

The C9X rationale document further says that there is often hardware support for the fma() function, that is, an instruction that does a floating multiply and add.

These examples illustrate the point that many of the new functions are specifically designed to meet the demands of numerical programming. Whether all these functions fit within the "Spirit of C" depends on your philosophy, but they certainly move the language toward the goal of supporting numerical programming applications.

Classifying Numbers

We'll now look at a few additional examples of new functions. One new area is number classification, where you can determine whether a given number is normal, subnormal (numbers with a minimum exponent and leading zero bit for the fraction part), infinite, NaN, or zero. Here's an example:

```
#include <float.h>
#include <math.h>
#include <stdio.h>

void classify(double d)
{
    // print the number

    printf("%g\t", d);

    // print the classification of the number

    switch (fpclassify(d)) {
        case FP_ZERO:
            printf("zero\n");
            break;
        case FP_NORMAL:
            printf("normal\n");
            break;
        case FP_SUBNORMAL:
            printf('subnormal\n");
            break;
        case FP_INFINITE:
            printf("infinite\n");
            break;
        case FP_NAN:
            printf("NaN\n");
            break;
    }
}

int main()
{
    double d1 = 12.34;
    double d2 = 1.0 / 0.0;
```

```
    double d3 = log(-1.0);

    // regular number

    classify(d1);

    // infinity

    classify(d2);

    // NaN

    classify(d3);
}
```

One of the fundamental issues with numerical programming is how to handle cases where calculations overflow, or situations where an "impossible" operation is attempted, such as taking the log of a negative number. As part of handling these cases, values that are not normal numbers need to be represented: for example, positive infinity or NaN.

Gamma Functions

tgamma() and lgamma() represent another example of new functions. These compute the gamma function and the log of the gamma function. This function is related to factorial, in that:

```
n! = gamma(n + 1)
```

but the gamma function is a more general mathematical concept.

Here's an example of using these functions to compute the number of permutations of N objects taken R at a time:

```
#include <math.h>
#include <stdio.h>
#include <stdlib.h>

int main(int argc, char* argv[])
{
    if (argc != 3) {
        fprintf(stderr, "Usage: %s N R\n", argv[0]);
        return 1;
    }

    // get N and R

    int n = atoi(argv[1]);
    int r = atoi(argv[2]);

    // compute permutations using tgamma

    double num1 = tgamma(n + 1);
    double denom1 = tgamma((n - r) + 1);
    double res1 = num1 / denom1;

    // compute permutations using lgamma

    double num2 = lgamma(n + 1);
    double denom2 = lgamma((n - r) + 1);
    double res2 = exp(num2 - denom2);
```

```
    printf("result using tgamma = %g\n", res1);
    printf("result using lgamma = %g\n", res2);
}
```

For command line arguments of 10 and 5, the result is 30,240, that is, if you have 10 objects taken 5 at a time, then there are 30,240 possible permutations. The formula for permutations is:

```
N! / (N-R)!
```

The tgamma() function value grows very rapidly, and sometimes you might wish to use lgamma() instead. We've illustrated how to perform the same calculation using logs and exp().

Rounding the Results of Calculations

A final example illustrates the many functions that you can use for rounding to the nearest integer. "Nearest" has multiple definitions, as this demo makes clear:

```
#include <fenv.h>
#include <math.h>
#include <stdio.h>

int main()
{
    // smallest integer not less than the argument
    printf("%g\n", ceil(2.4));

    // largest integer not greater than the argument
    printf("%g\n", floor(2.4));

    // set rounding direction to upwards
    fesetround(FE_UPWARD);

    // round to nearby (upwards) int, and return a double
    printf("%g\n", nearbyint(2.4));

    // round to nearby (upwards) int, and return an int
    printf("%ld\n", lrint(2.4));

    // always round halfway cases upward
    printf("%g\n", round(2.5));

    // round to nearest integer not larger than the argument
    printf("%g\n", trunc(2.9));
}
```

The output of the program is:

```
3
2
3
3
3
2
```

We've tried to present some of the philosophy around the many new C9X math functions. These features represent a major step forward for C as a numerical and scientific programming language.

a remote active OS fingerprinting tool using ICMP

by Ofir Arkin

Ofir Arkin is the Founder of the Sys-Security Group, a free computer security research body. In his free time he enjoys doing computer security research. His publications and work are available from the group's website, *http://www.sys-security.com*.

ofir@sys-security.com

During the winter of 2000, I started researching the Internet Control Message Protocol (ICMP). The protocol goals and features were outlined in RFC 792 (and then later in RFCs 1122, 1256, 1349, 1812) as a means to send error messages for nontransient error conditions, and to provide a way to probe the network in order to determine general characteristics about it. My goal was to go through the relevant RFCs quickly and then continue with other more interesting protocols. Instead, I found that ICMP can be used to fingerprint operating systems.

Techniques for OS fingerprinting using TCP packets already exist and are well known: the nmap and queso tools are examples. As I continued to discover idiosyncrasies in the responses of different operating systems to small but legal tweaks to ICMP packets, I published information about the way these packets could be used to determine the type of operating system in use at a particular destination not filtering incoming (and in some cases, outgoing) ICMP packets. The first fruit of this research was a paper that can be found at *http://www.sys-security.com/html/*.

A large portion of the research paper is dedicated to active operating system fingerprinting techniques that I have discovered during the research project. Using active OS fingerprinting methods with ICMP requires less traffic initiation from the prober's machine in determining the underlying operating system of a targeted host. With most of the fingerprinting methods, only one datagram can be enough to accomplish this.

For quite some time people have asked me for an automated tool that will correlate some of the active OS fingerprinting methods I have discovered using ICMP. But the final push for the tool was done by J. D. Glaser, a good friend of mine, who asked me if I could use these ICMP fingerprinting methods to differentiate between Microsoft-based operating systems. Less than three hours later I had a little logic drawn and tested that could differentiate between the different Microsoft-based OSes using only one to three ICMP queries.

X and Xprobe

X is the logic which combines various remote active OS fingerprinting methods using ICMP into a simple, fast, efficient, and powerful way to detect an underlying operating system a host is using. Xprobe is the tool I wrote with Fyodor Yarochkin (*fygrave@tigerteam.net*) that automates X.

Xprobe is an alternative to some tools which are heavily dependent on TCP for remote active OS fingerprinting. This is especially true when trying to identify some Microsoft-based operating systems, when TCP is the protocol being used with the fingerprinting process. Since the TCP implementation with Windows XP, 2000 and ME, and with Windows NT4 and 98/98SE are so close, we are unable to differentiate between these Microsoft-based OS groups when using TCP with a remote active OS fingerprinting process; and this is only an example.

The number of datagrams we need to send and receive in order to fingerprint a targeted machine remotely with Xprobe is small. In fact we can send one datagram and

receive one reply and this will help us identify up to eight different operating systems (or groups of operating systems). The maximum datagrams the tool will send is four. This is the same number of replies we will need. This makes Xprobe very fast as well.

Xprobe probes can be very stealthy. Xprobe does not send any malformed datagrams to detect a remote OS type, unlike the common fingerprinting methods. Xprobe analyzes the remote OS TCP/IP stack responses for valid packets. Heaps of such packets appear in an average network on a daily basis and very few IDS systems are tuned to detect such traffic (and those which are, presumably, are very badly configured).

Usually people see the types of datagrams being used by Xprobe as evidence of a simple host detection attempt; in fact, the replying machines were not only detected, but their underlying operating systems were revealed as well. In the future, Xprobe will use actual application data with its probes. This will help in disguising the real intentions of the probes.

Xprobe might change the traditional intelligence-gathering approach. With the traditional approach we need to detect the availability of a host (using a host-detection method), find a service it is running (using port scanning), and then identify the underlying operating system (with a remote active OS fingerprinting tool). If the targeted machine is running a service that is known to be vulnerable, it might allow a malicious computer attacker to execute a remote exploit in order to gain unauthorized access to the targeted machine.

Xprobe takes advantage of several remote active OS fingerprinting methods discovered during the ICMP Usage in Scanning research project. Some operating system stacks do not correctly echo several field values within the same ICMP error message. This enables Xprobe to use multiple echoing integrity tests with just one ICMP error message sent by a targeted machine.

ICMP Error Message Quoting Size

Each ICMP error message includes the IP header and at least the first eight data bytes of the datagram that triggered the error (the offending datagram); more than eight bytes may be sent according to RFC 1122. Most of the operating systems will quote the offending packet's IP header and the first eight data bytes of the datagram that triggered the error. Several operating systems and networking devices will echo more than eight data bytes. Examples of operating systems that quote more include: Linux-based on kernel 2.0.x/2.2.x/2.4.x, Sun Solaris 2.x, HPUX 11.x, MacOS 7.x–9.x (10.x not checked), Nokia boxes, Foundry Switches (and other OSes and several networking devices).

ICMP Error Message Echoing Integrity

When sending back an ICMP error message, some stack implementations may alter the offending packet's IP header and the underlying protocol's data, which is echoed back with the ICMP error message. The only two field values we expect to be changed are the IP time-to-live field value and the IP header checksum. The IP time-to-live (TTL) field value changes because the field is decreased by one, each time the IP header is being processed. The IP header checksum is recalculated each time the IP TTL field value is decreased. With Xprobe we take advantage of ICMP Port Unreachable error messages triggered by UDP datagrams sent to closed UDP ports. Xprobe examines several IP header and UDP-related field values of the offending packet being echoed with the ICMP error message, for some types of alternation.

Xprobe probes can be very stealthy.

IP Total Length Field

Some operating systems add 20 bytes to the original IP total length field value of the offending packet in the data echoed with the IP header of the offending packet in the ICMP error message. Other operating systems subtract 20 bytes from the original IP total length field value of the offending packet. And some operating systems echo correctly this field value.

IP ID

Some operating systems do not echo the IP ID field value correctly with their ICMP error messages. They will change the bit order with the value echoed. Other operating systems will correctly echo this field value. Linux machines based on kernel 2.4.0–2.4.4 will set the IP identification field value with their ICMP query request and reply messages to a value of zero. This was fixed with Linux kernels 2.4.5 and up.

Fragmentation Flags and Offset Fields

Some operating systems do not echo the fragmentation flags and offset field values correctly with their ICMP error messages. They will change the bit order with these fields.

IP Header Checksum

Some operating systems will miscalculate the IP header checksum of the offending packet echoed back with the ICMP error message. Some operating systems will zero out the IP header checksum of the offending packet echoed back with the ICMP error message. Other operating systems will correctly echo this field value.

UDP Header Checksum

Some operating systems miscalculate the UDP header checksum of the offending packet echoed back with the ICMP error message. Some operating systems will zero out the UDP header checksum of the offending packet echoed back with the ICMP error message. Other operating systems will correctly echo this field value.

Precedence Bits Issues with ICMP Error Messages

Each IP datagram has an 8-bit field called the TOS byte, which represents the IP support for prioritization and type-of-service handling. The TOS byte consists of three fields. The precedence field, which is 3 bits long, is intended to prioritize the IP datagram. It has eight levels of prioritization. Higher priority traffic should be sent before lower-priority traffic. The second field, 4 bits long, is the type-of-service field. It is intended to describe how the network should make tradeoffs between throughput, delay, reliability, and cost in routing an IP datagram. The last field is unused and must be zero. Routers and hosts ignore this last field. This field is 1 bit long.

RFC 1812 sets requirements for IPv4 routers, and this affects the TOS and precedence bits. ICMP Source Quench error messages, if sent at all, *must* have their IP precedence field set to the same value as the IP precedence field in the packet that provoked the sending of the ICMP Source Quench message. All other ICMP error messages (Destination Unreachable, Redirect, Time Exceeded, and Parameter Problem) *should* have their precedence value set to 6 (INTERNETWORK CONTROL) or 7 (NETWORK CONTROL). The IP precedence value for these error messages *may* be settable.

Linux kernels 2.0.x, 2.2.x, 2.4.x will act as routers and set their precedence bits field value to 0xc0 with ICMP error messages. Networking devices that will act the same will be Cisco routers based on IOS 11.x–12.x and Foundry Networks switches.

DF Bit Echoing with ICMP Error Messages

Some operating systems set the DF (don't fragment) bit in error quoting when the DF bit is set with the offending packet. Some OSes will not.

The IP Time-to-Live Field Value with ICMP Messages

The sender sets the time-to-live (TTL) field to a value that represents the maximum time the datagram is allowed to travel on the Internet. In practice, the TTL gets decremented each time a packet passes through a router or IP stack. The TTL field value with ICMP has two separate values, one for ICMP query messages and one for ICMP query replies. The TTL field value helps identify certain operating systems and groups of operating systems. It also provides the simplest means to add another check criterion when we are querying other hosts or listening to traffic (sniffing).

Using Code Field Values Different from Zero with ICMP Echo Requests

When an ICMP code field value different from zero is sent with an ICMP Echo Request message (type 8), operating systems that answer the query with an ICMP Echo Reply message based on one of the Microsoft-based operating systems send back an ICMP code field value of zero with their ICMP Echo Reply. Other operating systems (and networking devices) echo back the ICMP code field value that was used with the ICMP Echo Request.

TOS Echoing

RFC 1349 defines the use of the type-of-service (TOS) field with ICMP messages. It distinguishes among ICMP error messages (Destination Unreachable, Source Quench, Redirect, Time Exceeded, and Parameter Problem), query messages (Echo, Router Solicitation, Timestamp, Information Request, Address Mask Request), and reply messages (Echo Reply, Router Advertisement, Timestamp Reply, Information Reply, Address Mask Reply). Simple rules are defined: an ICMP error message is always sent with the default TOS (0x0000). An ICMP request message may be sent with any value in the TOS field. A mechanism to allow the user to specify the TOS value to be used would be a useful feature in many applications that generate ICMP request messages. The RFC further specifies that although ICMP request messages are normally sent with the default TOS, there are sometimes good reasons why they would be sent with some other TOS value. An ICMP reply message is sent with the same value in the TOS field as was used in the corresponding ICMP request message. Some operating systems will ignore RFC 1349 when sending ICMP Echo Reply messages and will not send the same value in the TOS field as was used in the corresponding ICMP request message.

How does Xprobe work?

Currently Xprobe deploys a hardcoded logic tree. Initially a UDP datagram is sent to a closed UDP port in order to trigger an ICMP Port Unreachable Error message. This sets up a limitation of having at least one port not being filtered on a target system with no service running.

A few tests can be combined within a single query, since they do not affect results of each other.

Upon the receipt of the ICMP Port unreachable error message, the contents of the received datagram are examined and a diagnostics decision is made. If any further tests are required, according to the logic tree, further queries are sent. For a detailed explanation and graphical representation of the logic please go to: *http://www.sys-security.com/html/projects/X.html*.

As always, an example is worth a thousand words...

```
[root@godfather /root]# xprobe -v www.redhat.com
X probe ver. 0.0.2
————————————
Interface: ppp0/213.8.195.154
LOG: Target: 216.148.218.195
LOG: Netmask: 255.255.255.255
LOG: probing: 216.148.218.195
TEST: UDP to 216.148.218.195:32132 [98 bytes] sent, waiting for response.
TREE: Cisco IOS 11.x-12.x! Extreme Network Switches.Linux
2.0.x!2.2.x!2.4.x.
TREE: Linux kernel 2.0.x!2.2.x!2.4.x! Based.
TREE: Linux kernel 2.2.x!2.4.x! Based.
TEST: ICMP echo request to 216.148.218.195 [68 bytes] sent, waiting for
response.
TREE: ICMP echo/echo reply are not filtered
FINAL:[ Linux 2.2.x/2.4.5+ kernel ]
[root@godfather /root]#
```

The number of tests in the output is the number of datagrams sent (represented by the word TEST in the output). With the example above of the RedHat Linux Web site, Xprobe sent only two datagrams to the target. It took Xprobe 700 milliseconds to figure out what the underlying operating system was. The time can be even faster, depending on the type of link you have. On local LANs, you will get the fastest results. For each internal logic test, Xprobe prints the word TREE in the output, representing a decision-tree criterion we check the replies against.

Many sites will block most incoming UDP packets (for good reason), but not all. When probing msdn.microsoft.com, we found that UDP port 53 (used for DNS queries) was allowed through, and we used the -p 53 option to specify that to Xprobe:

```
[root@godfather /root]# xprobe -v -p 53 msdn.microsoft.com
X probe ver. 0.0.2
————————————
Interface: ppp0/x.x.x.x
LOG: Target: 207.46.196.115
LOG: Netmask: 255.255.255.255
LOG: probing: 207.46.196.115
TEST: UDP to 207.46.196.115:53 [98 bytes] sent, waiting for response.
TREE: IP total length field value is OK
TREE: Frag bits are OK
TEST: ICMP echo request to 207.46.196.115 [68 bytes] sent, waiting for
response.
TREE: Microsoft Windows Family TCP stack
TREE: Other Windows-based OS (ttl: 48)
FINAL:[ Windows 2k. SP1, SP2 ]
[root@godfather /root]#
```

This time we succeed in our identification. This also means that UDP port 53 on msdn.microsoft.com is closed, but not covered by the firewall. It also means that ICMP Port Unreachable error messages are allowed from internal Microsoft systems to the outside world. The example above shows you why it is important to use egress and ingress filtering on your firewall.

In the future if we will receive an ICMP Destination Unreachable (Communication Administratively Prohibited) error message from a filtering device protecting the targeted host, we will be able to fingerprint the filtering device as well. Compared to other remote active operating systems fingerprinting programs, Xprobe is a very efficient one. Just remember that the official release is only 0.0.2 and that there is still a long way to go and many enhancements to introduce.

Future Development

The following issues are planned to be deployed (we always welcome discussions or suggestions though):

- Fingerprints database (currently being tested)
- Dynamic, AI-based logic (long-term project)
- Tests will depend heavily on network topology (pretest network mapping will take place).
- Path-to-target test (to calculate hops distance to the target)
- Filtering devices probes
- Logging capabilities
- DB fingerprints creator

Future implementations will use packets with actual application data to dismiss chances of being detected. Other network mapping capabilities shall be included (network role identification, search for closed UDP port, reachability tests, etc.). For more information, visit: *http://www.sys-security.com/html/projects/X.html*.

musings

by Rik Farrow

Rik Farrow provides UNIX and Internet security consulting and training. He is the author of *UNIX System Security* and *System Administrator's Guide to System V*.

rik@spirit.com

The LISA conference in San Diego last December had a profound effect on my life. As usual, it was not so much the scheduled activities as it was the little things that happen while you are there. And I am not talking about the food or trips to see the Pacific Ocean either.

I sat in on some parts of the Security track; no surprise there. I listened to Æleen Frisch say that she considered privileges in Windows NT/2K/XP a wonderful security mechanism, along with ACLs, and that these should be copied and used more. Of course, privileges are capabilities and, like capabilities, no less prone to failure and misuse. The Vixie-cron elevation of privilege exploit, which only worked because of a small mistake in the new Linux capabilities kernel feature, is one example. I don't even want to discuss the complexities created with NT's 27 different privileges, with several more in Win2K. Who needs to be an administrator when debug or restore privilege is at hand?

And these technologies are old. It seems that every time I meet Peter Neumann, he tells me about inventing ACLs for Multics back in the sixties (he obviously doesn't remember me). Capabilities were created in the seventies. If these security features were so good, why don't we have secure systems today?

Security Engineering

The answer to this question lies in *Security Engineering*, a book by Ross Anderson. Greg Rose mentioned this book as a "must read," so I ordered it from Amazon and settled down to read it. "Fifty pages a day," I told myself. And so it went for the first 250 pages or so.

Ross Anderson, who now teaches at Cambridge University, has the type of experience required to write a book like this. Perhaps there are other people who have worked in programming, banking, crypto, and medical privacy fields, but few are also as eloquent. And it is the rare technical book that can keep me awake at night (instead of putting me to sleep), but some sections of his book did keep me up. My wife was amazed.

But then, security is my field, so a book like this is much more significant to me than it would be to most people. What might make this book interesting to the larger computer audience is that it approaches security mechanisms in general, rather than focusing on some aspect of computer security, such as authentication or secure programming.

For example, ever wonder how ATMs work and what mechanisms protect both you and the bank that operates the ATM? Anderson goes into great detail explaining not only what gets encrypted but how keys are managed. Anderson also writes about some failures of the ATM system, including the fact that banks have lied about the reliability of the ATM system and how that impacts people.

Discussions of ATMs touch upon physical tamper resistance, actually a critical part of ATMs, because these systems may be (and often are) placed in hostile locations. Anderson explains just how difficult it is to produce tamper-resistant hardware by discussing the many ways that people have used to read keys hidden in smartcards: physical attacks, timing attacks, power monitoring attacks, and protocol attacks. Anderson does a splendid job by putting all this in perspective. While stealing a key from a

smartcard might take weeks of work, using costly machinery, a successful attack can pay off when the target is a widely used key, for example in a pay-TV operation.

Anderson's focus might appear to wander, but it always winds up being highly relevant. You might wonder what banking has to do with creating secure systems. Yet the banking industry has developed policies and practices that successfully prevent or detect large-scale fraud, and there is a lot to learn from bankers about protocols, encryption in practice, and intrusion detection. I was amazed not only that this was true but also by some of the low-tech mechanisms that continue to be used.

Anderson also talks about the failures of systems: what goes wrong and why it went wrong. Sometimes, it may be fraud: for example, phone phreaking with blue boxes or a bank employee managing to steal over a hundred thousand dollars from a little old lady (and later 'fessed up). Or it might be a more general example, what Anderson calls architectural errors. For example, if you are using a PC and are going to digitally sign a request to buy his book from Amazon, how do you know that what your signature has authorized is what appears on the screen? Clicking to complete the transaction may have just digitally signed something inserted by a clever virus (instead of your Amazon order). So, instead of getting his book in the mail, you might instead have remortgaged your house to Mafia Real Estate, Inc. While this sounds like it would be easy to clear up, elsewhere in this book Anderson explains that legislation has been proposed that would give a digital signature the same weight as the physical signature that you handwrite in the presence of witnesses.

Chapter 19 is entitled "Protecting E-Commerce Systems." It covers SSL and SET, but it goes much deeper than that. Anderson points out, rightly so, that most credit card fraud does not involve the Internet or even e-commerce. Some does involve systems where credit card information is stored, but this should never be the same system that is running the Web server. Most fraud is done by insiders, a much more difficult problem to solve by technology – but not by policy and practices.

Sections 6 and 7 of Chapter 19 had me so excited I could hardly sit still. Network economics explains how networked systems (not real networks) work to foster and maintain monopolies, such as Microsoft. I have often tried to understand this, and to explain it, but Anderson does a marvelous job, based in part on an article by Andrew Odlyzko: *http://www.acm.org/networker/issue/9805/ssnet.html*. In brief, Odlyzko posits that both the Internet and Microsoft have prospered because these technologies "offer an irresistible bargain to a crucial constituency; namely developers, while managing to conceal the burden it places on users."

Anderson also briefly explains the real purpose behind Passport, the part of Microsoft's .NET initiative that will handle authentication and credit card payments for participating Web sites. While the ostensible purpose of Passport is single sign-on, the real purpose is to create a web of vendors, all of whom use Microsoft for clearing transactions. Microsoft can then collect a huge amount of data about buying habits and sell it among participating vendors. A great deal for the vendors, not for the users. Note that Microsoft already collects and collates information about any visit to its many Web sites.

Anderson does not reveal the same level of passion that I feel in regard to certain monopolies. And he even, occasionally, writes something that I know is wrong (for example, most UNIX systems today have passwords stored in publicly readable files, or that a Sendmail bug in 2000 permitted reading the passwords from a protected file).

Most credit card fraud does not involve the Internet or even e-commerce.

All-in-all, this book is a must read important for anyone interested in security, not just for computers but for the systems that we interact with in everyday life. A must read.

Another LISA Story

The most moving event of LISA did not appear on the earlier schedules. Bill LeFebvre talked about his experience working as a senior sysadmin for Turner Broadcasting on September 11. LeFebvre explained how a pool of large Solaris systems can be "switched" from one domain to another "quickly" to support expected surges in visits to a Web site. For the most part, these switches can be anticipated and planned for. B – but, obviously, not in the case of terrorist attacks, which are not scheduled, and are obviously not planned for.

CNN is not your normal Web site by any means. During the US working day, it sees an average hit rate of 85,000 hits/minute, with peaks up to 300,000 hits/minute. Between 8:45 and 9 a.m. EDT, the number of hits went from 85,000 to 229,000 per minute, and as the hits continued to pour in, the Solaris servers started to melt down. By switching servers (by changing IP addresses used by load balancers and pointing the servers at different back-end file servers), the system recovered and went on to handle a peak estimated at almost 2 million hits/minute and to serve a record number of pages that day. In the afternoon, Turner sysadmins noticed an increase in the number of visits to the Cartoon Network Web servers, so they switched over added capacity to handle this. At this point LeFebvre's voice cracks (as my eyes tear up while writing this), considering the impact of that day's events on children.

On a side note, I do want to mention that the people responsible for September 11 have not been punished. The actual perpetrators died that day, but the people behind the scheme – for example, those providing the money and the training – have still not been identified. The "war" in Afghanistan has more to do with an oil and natural gas pipeline than with terrorism. More people have died as "collateral damage" in Afghanistan by the end of January 2002 than died in the attacks on the US on 9/11.

Finally, and on a brighter note, I got to chair a panel entitled "So You Want to Write a Book." As anyone who has written a book will tell you (well, almost anyone, as I have met people who really don't work very hard at it), book writing is highly overrated. It is more difficult, time consuming, and emotionally draining than most people consider. And books rarely pay off financially.

Having said that, I would like to remind people that I am still the SAGE short topics editor and will be at least until June. I am getting a taste of just how difficult it is to get firefighters, that is, most system administrators, to sit down and complete what is essentially a single chapter of a book. I am continuing the thread of the existing booklet topics but also attempting to expand them by getting some of the knowledge learned by system administrators into print. I am looking for authors (always) but, in particular, for someone who can write about UNIX user account management, from the basics to the many different packages that have been used at large sites, and has published in USENIX or LISA proceedings; and for someone who can write about network management tools; and so on. If this appeals to you, and you imagine you have the time to write a "chapter" in the book of system administration, contact SAGE at *sagebooklets@usenix.org*.

ISPadmin

Network Design and Operation

by Robert Haskins

Robert Haskins is currently employed by WorldNET Internet Services, an ISP based in Norwood, MA. After many years of saying he wouldn't work for a telephone company, he is now affiliated with one.

rhaskins@usenix.org

Introduction

This installment of ISPadmin examines how service providers large and small might set up their IP (and associated) networks to provide services to their customers. After covering some network basics, the article illustrates how a small dialup provider might set up its network and how a larger provider might. Issues surrounding the traditional small and large dialup ISP are examined. Finally, such topics as staff requirements, service level agreements, and network design considerations are pondered.

ISP Networking Background

This section contains basic networking concepts and terms and their meanings.

Paid Egress

Egress, synonymous with "exit," is how network engineers refer to the points where traffic leaves the provider's network and enters another entity's network. There are two types of egress: paid and peer. *Paid egress* is bandwidth that the provider buys from another provider to deliver traffic that is not destined for the provider's, or peer's, network. In the greater Boston area, it runs about $500 per megabit/sec (Mbps) per month without local loop charges.

Peer Egress

The second type of egress is *peer egress,* or *peering*, where little or no cost besides hardware is incurred. Peering is exchanging traffic destined for someone else's network directly with them, rather than using paid bandwidth. There are two types of peering arrangements, public and private. *Private peering agreements* are connections that take place in private, common facilities. GlobalNAPS (or GNAPS, a CLEC associated with my employer) allows no cost peering for its customers. If two customers co-located in GNAPS facilities would like to peer, and GNAPS doesn't incur any cost, the providers are allowed to peer without additional cost from GNAPS.

In order for most larger providers to peer, they require a considerable amount of traffic to be exchanged and that the traffic be "roughly balanced." For example, WorldCom requires 150 Mbps of traffic from the provider's network to WorldCom and 150 Mbps from WorldCom's network to the provider's network. (Even providers who qualify for WorldCom's free peering are required to pay for a connection into their facility.) *Public peering points* are facilities set up for the express purpose of enabling peering relationships (e.g., MAE EAST, WorldCom's widely known public peering facility; there are many such public peering points run by a wide variety of providers). In the case of public peering points, the host of the exchange point usually charges for connections into the facility, in order to cover its costs and make a profit.

Autonomous System Number (ASN)

When a provider has multiple egress points in its network, an ASN is used to identify what network the traffic originated from (in the case of outgoing packets) or is destined for (in the case of incoming packets). It usually consists of a unique four-digit

number (e.g., "AS1234") which tells other devices on the Internet which network a particular packet belongs to, when a network is multi-homed. An ASN is assigned by the American Registry for Internet Numbers (ARIN) or other Internet numbering authority.

Here are common circuit acronyms and associated speeds for the United States (from the ISP Glossary listed in the references) and a few common service provider acronyms.

Dedicated Circuit Acronyms and Speeds

DS0 (Digital Service 0):	64 Kbps clear channel (normally provisioned by the telephone company as 56 Kbps)
T1 (DS1):	24 DS0s or 1.544 Mbps
PRI (Primary Rate Interface):	single ISDN channel normally provisioned on a T1, supports both ISDN and plain old telephone service (POTS) connections
ISDN (Integrated Services Digital Network):	64 Kbps
T3 (DS3):	672 DS0s or about 43 Mbps
OC3 (Optical Carrier):	155.52 Mbps
OC12:	622.08 Mbps
OC48:	2.488 Gbps

Some Common Service Provider Acronyms

ILEC:	Incumbent Local Exchange Carrier (e.g., Verizon, Qwest)
CLEC:	Competitive Local Exchange Carrier (e.g., Level3, GNAPS)
DLEC:	Data Local Exchange Carrier (e.g., Covad)
POP:	Point of Presence
RAS:	Remote Access Server
ATM:	Asynchronous Transfer Mode
SONET:	Synchronous Optical Network
DOCSIS:	Data Over Cable Service Interface Specification

Small Provider Backbone

Figure 1 illustrates how a small provider might design its network. The box marked "Central POP" is the central site where the provider has access to the Internet. The boxes marked "Remote POP" represent off-site locations housing RAS gear or customer-dedicated connections.

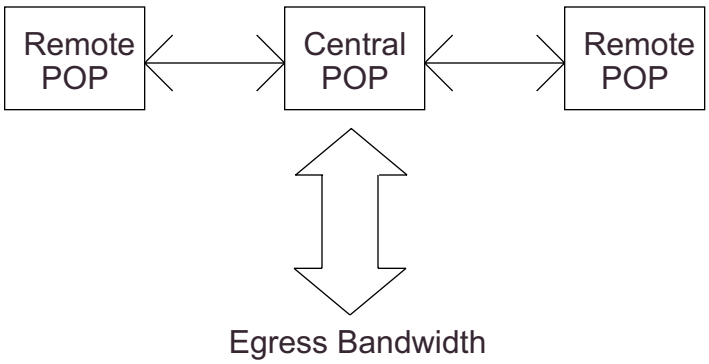


Figure 1: Small Provider Network

The characteristics of a "small" provider are centered around the following:

- One egress point for traffic
- Limited peering
- Small routers, not hierarchical
- No or limited redundancy

Of course, limiting cost is what usually drives these attributes. While multiple egress points are desirable

from a reliability standpoint, redundant access to the Internet is simply beyond most small providers. The size of the paid egress is likely to be measured in T1s, not T3s or DS3s.

Peering is another area that most providers won't be able to afford, or qualify for, except for very specific situations. A small ISP may utilize private peering in facilities, but likely won't use public peering.

A small provider probably uses smaller routers, with limited port counts and functionality. For example, the Cisco 2500/2600 series routers would be used in most places except for the provider's hub (where servers might be located, for example), where a larger router like the Cisco 3600 router could be used.

There is normally no redundancy engineered in a small provider's network. The cost and complexity is beyond the small provider (and even can be too much for larger providers as well).

Larger Provider Backbone

Figure 2 illustrates how a larger provider might design its network. The boxes marked "Core" indicate the core routers/nodes of the network. Each core node usually has two or more connections to other core nodes in the providers network, forming the backbone of the provider's network. The boxes labeled "Border" indicate remote POPs that terminate customer connections. In the case of a "traditional" ISP, customer connections might be leased lines running at T1 or T3 speeds. In the case of a dialup ISP, the border routers are facilities with RAS gear serving dialup customers. In the case of a cable modem ISP, the border routers are cable head ends where traffic exits the cable provider's network and enters the Internet. (See the DOCSIS Web page for more information on this topic.) Egress can take the form of peering points or paid bandwidth. Egress points are normally on the provider's core network, where fast routers and interconnects are located.

The characteristics of a larger provider might be the following:

- Multiple egress points
- Multiple peers
- Large routers set up hierarchically
- Some redundancy

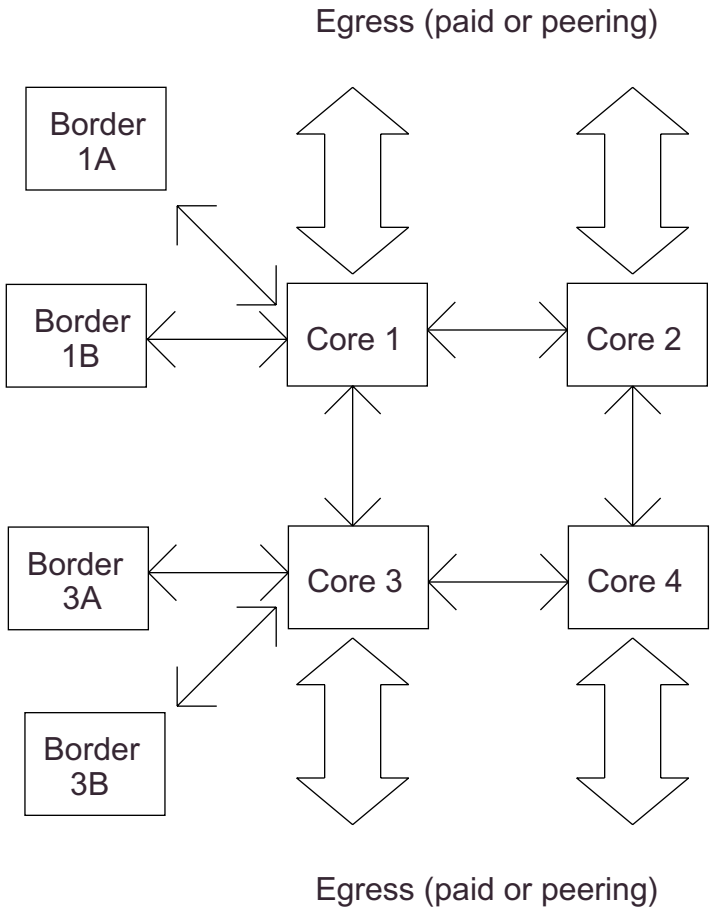


Figure 2: Large Provider Network

Cost is less of an issue for a larger provider. It will likely have multiple paid egress points for redundancy, at T3 speeds. A big service provider will have multiple peers at both private and public peering points. Large routers such as the Cisco 7000 series routers or Juniper Networks M-series will be used, set up in a border/core arrangement. The provider's backbone network will likely have some redundancy, so the loss of a single link or POP won't take down the entire network.

Most larger dialup ISPs have merged with telephone companies at this point.

The border and core router design is a common method used by larger providers to segregate their networks. Slower links terminate at the border routers, which send traffic to nearby core routers. Core routers have faster links along with peering and paid bandwidth connections. Border routers are where customer connections are normally terminated. An exception to this might be when the customer purchases high-speed bandwidth (OC3 and above) . Ordering a "fast" connection may cause the provider to terminate the connection at the faster core routers where higher speed line cards are available. This design results in faster access on and off the provider's network for the customer. As one might expect, "faster is better" in more ways than one.

The backbone protocol is likely to be ATM. While ATM is designed for voice and data networks, it is a mature technology and in wide use. Another option is IP over SONET, though this is normally utilized in OC12 and faster links. As previously mentioned, redundancy is engineered to the extent possible (within economic reason) in a larger provider. In some cases, additional markets are justified by providing additional network paths to certain POPs. Of course, the provider's service level agreements with its customers often dictate where and how certain links are provisioned for redundancy.

Small Provider Dial Network

In the case of a dialup ISP, a smaller provider will usually purchase T1 PRI line(s) in the markets they would like to reach. Often, connections from the local ILEC are purchased and terminated in the ISP's facilities in the region served by dialup. If the ISP chooses to use a CLEC, very often a large coverage area can be obtained from one POP location. For example, GNAPS serves a substantial part of Massachusetts, New Hampshire, Vermont, and Rhode Island from its location in Quincy, Massachusetts. The customer simply purchases appropriate PRI (and rackspace) and obtains coverage for the entire region and only has to site equipment in Quincy. If the same coverage was desired from Verizon, numerous PRIs would have to be ordered and facilities would be required in many Verizon POPs to obtain similar coverage.

Often, a small provider will have a small state or regional dial coverage. National/International coverage might be provided by a contract with a larger provider, if necessary. For example, both GRiC and IPASS provide national/international coverage.

Large Provider Dial Network

Large ISPs usually utilize DS3 PRI lines across the country. Using such high-bandwidth lines and associated equipment enables the provider to reduce costs. This is because DS3s and associated RAS equipment are cheaper by the port in larger capacities.

Most larger dialup ISPs have merged with telephone companies at this point. The only possible exception to this (outside of WorldCom and other really big providers) is StarNet, which has managed to stay independent. Most other providers in this space (Concentric, Split Rock, Ziplink, etc.) have merged with CLECs or gone out of business. This consolidation of the industry is a testament to the cost of PRI lines, as consolidation reduces the cost of these lines on the balance sheet.

Larger ISPs often have POPs across the country. If the ISP is associated with a ILEC/CLEC, coverage outside of the home territory will be through another ILEC/CLEC. This will ensure that the provider has a wide coverage base.

Miscellaneous Topics

The <insert word here> Service Provider

These days, there is no shortage of differing types of service providers. Most *<insert word here>* service providers are a variant of the Web-hosting provider. These include application service providers and storage service providers, among others. These types of providers typically have a backbone network as outlined in the "Larger Provider Backbone," above. An important difference would be the fact that in a dial or dedicated environment, the direction of traffic is usually inbound, whereas in a Web-hosting environment, data flow will normally be outbound.

Staff

At Ziplink, four network engineers handled the backbone network which included approximately 70,000 ports. The RAS engineering staff consisted of approximately six full-time engineers. Of course, a Network Operations Center staff was available to both groups, in order to troubleshoot and perform simple fixes.

Central vs. Distributed Network Design

Some providers may not utilize a backbone network for some or all of their POPs. This means they simply purchase egress at every location where their RAS gear is located and forego the costs and headaches associated with running one's own backbone network. The downside of such a design is that the provider has little control over these individual connections and is at the mercy of the egress providers. Costs will be higher when the provider runs its own backbone network, as cross-country network links will usually be more expensive than buying egress at each POP.

Service Level Agreements

Service Level Agreements (SLAs) are formal definitions of the type of service the provider will give to the customer. SLAs tend to vary widely from provider to provider, and customer to customer, depending upon each party's particular business needs. Of course, a provider wants the most flexible SLAs as possible, while the customer wants 100% uptime no matter what extenuating circumstances may exist.

Wholesale Dial Providers

Many substantial end-user ISPs (such as MSN and Prodigy) have a small dialup network or none at all. Instead, they purchase access from a wholesale dial provider such as Level3 and let them manage the RAS gear, ports, and associated headaches. The end-user ISP purchases access in the form of ports, time (hours), and/or users.

Security/DoS attacks

No discussion of this topic would be complete without some mention of the security issues related to providers. Service providers are often the victims of attacks, as they lease fast connections to other providers. Many attacks take the form of denial-of-service (DoS) attacks, where an attacker stops an ISP's customers from being able to access the services they purchase by filling up the ISP's network connections. Distributed DoS attacks are a variant of the DoS attack, except the attacker mounts its attacks

from multiple hosts. Detecting and mitigating these sorts of attacks are the topic of much current research. DoS attacks are stopped by implementing appropriate filters on egress routers. DoS attacks do not show signs of decreasing, at least in the near future.

A good source of information for learning about the service provider business in general is the ISP Planet home page listed in the references. Until next time, please send your questions and comments to me!

References

American Registry for Internet Numbers (ARIN): *http://www.arin.net/*
Avi Freedman's Multi-Homing page: *http://www.netaxs.com/~freedman/multi.html*
Cisco Systems: *http://www.cisco.com/*
DOCSIS starting point: *http://www.docsis.org/*
GlobalNAPS: *http://www.gnaps.com/*
GRiC: *http://www.gric.com/*
IPASS: *http://www.ipass.com/*
ISP Glossary: *http://isp.webopedia.com/*
ISP Planet: *http://www.isp-planet.com/*
Juniper Networks: *http://www.juniper.net/*
Level3: *http://www.level3.com/*
MAE Services and Facilities: *http://www.mae.net/*
MSN: *http://www.msn.com/*
Prodigy: *http://www.prodigy.com/*
Qwest: *http://www.qwest.com/*
StarNet/MegaPOP: *http://www.starnetusa.net/*
Verizon: *http://www.verizon.com/*
WorldCom Business Internet Dial:
http://www1.worldcom.com/us/products/access/dial/
WorldCom: *http://www.worldcom.com/*
WorldCom's Peering Policy: *http://www.uu.net/peering/*

a recipe for a successful linux user group

by Rick Moen

Rick Moen is a sysadmin, member of the San Francisco Bay Area Linux community, and Board member for the Bay Area chapter of LISA.

rick@linuxmafia.com

The advice given here can apply to any flavor of user groups, not just Linux. *Ed.*

Having seen (and run) quite a few Linux user groups (LUGs), and having observed some thrive and others die, I can hazard some firm recommendations. If you're thinking of starting a LUG, or are running one now, please ponder these lessons, drawn from other LUGs' experience. In fact, please consider reviewing this list from time to time, as a kind of checklist.

1. You need a Web page.

I can't stress this enough. The Internet is crucial to Linux – it made Linux possible and is where everything happens. If your group isn't on the Net, it might as well not exist.

By "the Net," I mean not just Web pages, which are its most visible service, but also mailing lists, Usenet newsgroups, and FTP file archives, among other things. It's your source for software, the forge where open source tools are designed and crafted, your method of publication, your social club, and your research library.

Each major function of your group should have a Web page; if you start doing InstallFests, create an InstallFest page.

2. Your Web page needs a reasonable URL.

The usual *http://www.some-isp.com/~username/lugname/* URL isn't good enough. You want people who know no more than the group's name to find you easily. For that, *http://www.lugname.org/* is ideal in the USA – and you can use similar formulas elsewhere, such as *http://www.lugname.org.au/*. Consider choosing a group name whose Internet domain isn't taken (check at *http://whois.internic.net/whois.html*, for com/org/net domains) and then paying to register that domain, and have an ISP virtual-host it. It's not that expensive.

Given that this is the Linux world, the odds are that one or more of your core volunteers owns a co-located Internet host and will be willing to host your pages and domain for free.

The odds are that your Web page will start out somewhere less desirable, such as a subdirectory of someone's home page, or a free hosting service such as Geocities – but you should aim toward having your own domain, in the longer term.

Remember that the Net is worldwide. If the best/cheapest hosting is at, say, a friendly LUG site on another continent, take it.

3. You need a regular meeting location.

Changing meeting locations risks losing attendees like mad. Why? Because some will come to the prior meeting location, get discouraged, and maybe even conclude that your group has folded – and also because finding out how to get there, where to park, whether the neighborhood's OK to walk in, etc., is a strain on people, each time you move.

The location doesn't have to be impressive. I've seen a college cafeteria suffice for one group, and a small downstairs room in someone's house does well for another. It just has to be reliably usable.

One LUG in my area fell apart largely because the president set an aggressive meeting schedule, and then failed to show up to unlock the meeting room.

4. You need a regular meeting time.
"Regular" usually means following an easily remembered and used formula, suitable for people's calendars, pocket planners, and Palm Pilots – such as fourth Thursday. Don't get fancy with things like "every other Thursday." Make it so anyone with a calendar can easily figure out when the next meeting will be.

5. You need to avoid meeting-time conflicts.
Check out the schedules for nearby technical events: Linux user groups, Perl groups, and whatever else your target audience is likely to want to also attend. Don't pick recurring dates that those other groups are already using. Hint: first and second Tuesdays, Wednesdays, and Thursdays are over-popular meeting days. Their attraction is that they're easy to remember – and mid-week days are generally good for people. But, in my immediate vicinity, for example, there are four competing groups sharing first Tuesdays.

6. You need to make sure that meetings happen as advertised, without fail.
One LUG in my area fell apart largely because the president set an aggressive meeting schedule, and then failed to show up to unlock the meeting room. Would-be attendees looked up the next meeting date on the Web, showed up, found a locked door, and (soon) gave up on the group entirely. So, if possible, have multiple people arrange to show up early. Also, post signs/flyers near the meeting site.

If you need to cancel or reschedule an event that you've already been advertising as "upcoming," don't simply remove the original listing on your Web pages; continue to list it, prominently marked as cancelled/rescheduled.

7. You need a core of several Linux enthusiasts.
LUGs have succeeded wonderfully on the strength of ongoing efforts from as few as four energetic and inquisitive people. That's really all you need, but one or two are not enough. Email is terrific for coordination.

Your core enthusiasts don't need any Linux knowledge initially, but must be "self-starters," have Internet access, and know how to use it well.

8. Your core volunteers need out-of-band methods of communication.
By that I mean outside your user group's regular electronic means of communication. One college LUG operated all email mailboxes used by its officers through the LUG's Web server machine, which then went down at the beginning of summer recess. It thus proved the proverbial "single point of failure" for group communications. Most officers change residences at the academic year's end, and there weren't even summer LUG meetings scheduled for regular dates, so the LUG nearly collapsed because its principals lacked any means of getting back in contact with one another.

Having circulated a list of stable non-LUG email addresses, telephone numbers, and/or postal addresses would have averted this near-disaster.

9. You need to get on the main lists of LUGs, and keep your entries accurate.
- *http://www.ssc.com/glue/*
- *http://lugww.counter.li.org/*
- *http://nlug.org/webring/*
- *http://www.redhat.com/apps/community/LUG/*
- *http://www.linux.org/groups/*

- *http://www.linux.com/interact//lugs/*
- *http://dmoz.org/Computers/Software/Operating_Systems/Linux/User_Groups/*

Assign someone in your group to re-check your LUG list entries periodically, say, every quarter. You'll be amazed at how inaccurate they become over time. Keep a list of all the places where you have such entries and also a "publicity" list (of places you send notices of upcoming events). Sometimes, it helps to print these out and use them literally as checklists.

An inaccurate LUG-list entry is often much worse than none at all. Directing prospective members to an obsolete URL, or telling them the wrong meeting date, actively hurts your membership effort.

So, before submitting an entry to any LUG list, do some spot checks on the existing entries' general level of accuracy. Widespread inaccuracy (e.g., dead links, wrong information on meeting dates and places) may indicate a hidden gotcha. Some lists are so badly maintained that their staffers ignore corrections you send in. For example:

- *http://www.currents.net/resources/usergroups/usanc.html*
- *http://dir.yahoo.com/Computers_and_Internet/Software/Operating_Systems/Unix/Linux/User_Groups/*

Both of these lists are traps for the unwary LUG leader in that they accept additions but appear to ignore correction/update notices. Once your entry becomes out-of-date, it stays that way.

10. You must have login access to maintain your Web pages, as needed.
An unchanging page that someone else created for you isn't good enough. You need to be able to fix/edit/enlarge your site on short notice. Typically, this requires login access via SSH[1] (or Telnet, if necessary) to the hosting Web server's command shell.

A number of Linux groups attempt to get by with a static page on some site to which they themselves lack maintenance access – for example, on a parent group's existing Web site. The convenience isn't worth the disadvantages; don't go that route.

11. Design your Web page to be forgiving of deferred maintenance.
Much as we'd like our LUGs' "upcoming events" and other time-sensitive information to be always current, it isn't going to happen. Sometimes you don't re-check and update them for a week or two. Therefore, always list several months' upcoming events. (You know when they'll be because you have a meeting-date formula, right?) That way, when you're unavailable for Web-page maintenance for two months running, the Web page will still include current meeting information.

Somehow, my local LUGs' webmasters seem resistant to that simple idea, with the result that most list only one upcoming meeting at a time, which, for three quarters of the month, because of the inevitable deferred maintenance, ends up being last month's date.

The whole point of listing specific upcoming meeting dates is to make it unnecessary for casual visitors to work out when the next second Tuesday will be, by doing it for them. But that effort is wasted when the only meetings shown are already past. It makes new LUG members that much less likely, and additionally may lead some to think your LUG is now defunct.

An inaccurate LUG-list entry is often much worse than none at all.

Believe it or not, some prospective members will assume your group costs money, unless you say otherwise.

12. Always include the day of the week when you cite event dates. Always check that day of the week, first, using gcal[2] (or cal). gcal is your friend.

Why always include the day of the week on event listings? Because that gives viewers their best shot at remembering your event, how many days away it is, and how it fits into their schedules.

Additionally, the fact that you've furnished the correct day of the week for each date reassures visitors that you haven't messed up and listed the wrong date (which happens depressingly often) – in effect, a cross-check. Conversely, take care not to list a meeting's date correctly, but with the day wrong. That conveys the (probably accurate) impression that your event calendar can't be trusted.

It doesn't hurt to print out the current year's "gcal" listings for reference whenever you're doing calendar work. Mark significant holidays for your country on it. You can get them from *http://www.holidayfestival.com/*.

13. Place time-sensitive and key information prominently near the top of your main Web page.

Don't banish all meeting information to your events page, or tuck it into an unobtrusive text box for aesthetic reasons. Ensure that the most prominent items on your site are the ones viewers need most. Consider using "STRONG" or "EM" HTML tags on particularly important items, such as your date formula (e.g., "second Tuesdays at 7 p.m.").

Displaying time-sensitive information prominently is useful not only because that tends to be what viewers seek most often, but also because such text calls attention to itself when it needs updating. Think of this as comparable to putting perishables near the front of your refrigerator, datestamp outwards.

14. Include maps and directions to your events.

Some prospective members will be comfortable with maps, others with directions; you'll want to help both. Maps can be generated (for the USA, at least) on MapQuest or MapBlast. Have them as links for each listed event location. If there's a trick to parking nearby, describe it. If public transit is available, give details.

Remember: directions and maps are (particularly) for people who aren't yet members, not for your existing "in-crowd." The Web page for one LUG near me used to omit maps and directions, giving only the meeting date/time and the name of the building where it met. Don't fall into this common trap of making your pages useful only for existing LUG members.

One of the themes of this essay, in fact, is to try at intervals to look over your LUG's public information as if you were an interested newcomer. Does your public information (e.g., your Web pages) tell prospective members what they need to know? Is the most important and most time-sensitive information also the most prominent? If not, you need to redesign it.

15. Emphasize on your main page that your meeting will be free of charge and open to the public (if it is).

Believe it or not, some prospective members will assume your group costs money, unless you say otherwise. This is especially true of people accustomed to traditional user groups, which generally must charge dues to finance their dead-tree-media newsletters and other money-intensive operations. (See addendum #1, below.)

Conversely, if there will be an attendance fee (e.g., to pay for dinner), say so prominently, with the other event details.

16. You'll want to include an RSVP "mailto" hyperlink on some events.
Set up an email alias of "*rsvp@groupname.com*" pointing to the real mailbox of one or more volunteer, and include it as a link on event listings when you need an advance head-count (e.g., at restaurants).

Other aliases you'll want – if you operate your own Web server – may include "webmaster," "postmaster," "president," "webteam," and "help." If you use such aliases consistently for roles people fill, you'll be better equipped to smoothly handle the inevitable turnover in volunteers.

17. Use referral pages.
Over time, you'll find it desirable to reorganize your Web documents, rename pages or directories, or move the whole site (or part of it) to a different Web server. That's inevitable – but don't forget that other Linux groups, user group lists, search engines, and assorted individuals will have already created links to your old URLs without telling you they've done so. Thus, if you just move/rename your Web documents, you will break one means by which interested parties are accustomed to finding you.

The cure is to leave a "referral page" at the URL where the document used to be, guiding the user to its new location. Get in the habit of doing this whenever you would otherwise leave nothing where there used to be a page. It's better than their getting a 404 error.

Here's an example:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<HTML>
<HEAD>
<META HTTP-EQUIV="refresh" CONTENT="2; URL=http://www.newlocation.com/">
<TITLE>New location for FOO Home Page</TITLE>
<HEAD>
<BODY>
<H1>The FOO home page has moved!</H1> The new location is
<A HREF="http://www.newlocation.com/">http://www.newlocation.com/</A>
</BODY>
</HTML>
```

18. Make sure every page has a revision date and maintainer link.
It's traditional to put some variation on "Send feedback to [*link*]" or "Copyright © 2000 by [*link*]", or "Webmaster: [*link*]" where "[*link*]" is the maintainer's name with a "mailto" hyperlink. And there's a good reason for this tradition: it ensures that "feedback" comments – from sharp-eyed readers who catch errors, omissions, and ambiguities in your public information – will reach the right person. Make it easy for the public to help you, and some of them will.

Putting a datestamp at the bottom of each page, whenever you update it, reassures visitors that the site really is current. You may know that it's a living site with current data, but newcomers won't. Dead sites with obsolete data are far too common on the Web, and the datestamp marks your page as freshly edited. (Equally, it reminds you of when you last overhauled the site.)

Putting a datestamp at the bottom of each page, whenever you update it, reassures visitors that the site really is current.

Need I mention that the worst-maintained LUG sites in my area lack revision dates and maintainer links? It's true.

19. Check all links, at intervals.

Your pages will probably contain numerous links to external sites, as well as to other parts of the same site. The remote sites can and do move around (or disappear) without your having heard, and local links can and will break because of your own imperfect maintenance.

There's a quick-and-easy way to catch all such breakage: open your Web browser and clear its cache, which makes all links appear as unread. Now, visit each link in turn. If you do this at intervals (e.g., every six months), you'll at least find any broken links and maybe other people's referral pages for remote pages that have moved.

It's possible (but a nontrivial task) to find all sites that link to your pages by analyzing your Web server's log files. That knowledge will prove useful when, for example, your Web site moves or gets substantially reorganized. (You can then advise the other webmasters.) If you're feeling ambitious, try to find them. Search engines are useful for that task: Google, Infoseek, AltaVista, and others. (Don't forget dejanews.com, for searching Usenet posts.)

One tactic that doesn't work: putting a note to webmasters on your site, asking them to let you know of links to it. I had such a request on my Bay Area Linux Events page (*http://linuxmafia.com/bale/*) for five years, with zero responses.

20. You may want to consider establishing a LUG mailing list.

Any reasonable Linux host (machine) with a constant Internet connection can run mailing-list software. GNU Mailman (*http://www.list.org/*), for example, is easy to set up and administer, and it comes with automatic Web archiving for your mailing list. Many groups find it useful to have both a main discussion list and a low-volume announcement-only list.

Do not announce your upcoming meetings only to your mailing lists. By definition, those will reach only existing members. The whole point of having a public presence (e.g., Web pages) is to both serve existing members and attract new ones.

Some commercial services let you set up "free" mailing lists on their servers, where their gain lies in revenues from mandatory ads auto-appended to all posts, plus, of course, the ability to sell your subscription list to other advertisers. Beware that you may find you're not the "owner" of your own list, in the event of a dispute over its management.

Some groups have founded Web-based discussion forums, instead of email mailing lists, either on their own servers or on commercial services. I have yet to see one that wasn't stagnant and inbred. The advantage of email mailing lists is that email is widespread and generally convenient for people.

If you do run LUG mailing lists, someone will have to function as list "owner," taking care of administrative requests and enforcing list policy. The latter category will include dealing with job recruiters. Many LUG lists have been overwhelmed with job ads from professional recruiters, sometimes posted multiple times a day. A policy that has worked well at one LUG is to require that all job ads be submitted to a club officer, who then posts it with a subject header starting with the string "JOBOFFER:". That

way, the number of job postings can be controlled, and people not wanting to see them can filter on subject headers.

21. You don't need to be in the Internet Service Provider business.
Leave the ISP business to the professionals. You won't be able to beat their prices, so don't try. When the moochers in your crowd ask for dial-in lines and shell accounts on the group's Web server, say "No."

22. Don't go into any other business, either.
I hear of LUGs being suckered into the strangest, most cockamamie business schemes. Don't: don't try to be a Web design firm, a technical support firm, a network design consulting firm, or a LAN cabling contractor. Or any other business. Not even if you're told it's for a wonderful charitable cause.

Along the same lines, remember that you are not a convenience for job recruiters. If allowed, they will spam your mailing lists and abuse every possible means of communication with your members. Nor are you a source of computers for the underprivileged, a repair service for random people's broken PCs, or a help desk for non-Linux operating systems and applications. Believe it or not, you will be pestered by all of the above sorts of strangers, on the "nothing ventured, nothing gained" theory.

23. Walk the walk.
It's painfully grotesque to see so-called Linux user groups mailing out announcements using MS-Outlook, Eudora, or Netscape Messenger for MS-Windows (or MacOS), or other proprietary mailers for legacy operating systems; visibly maintaining their Web sites using MS Front Page, Adobe Page Mill, or other junkware; and hosting their LUG mailing lists on eGroups / Yahoo Groups. Fortunately, these groups are in the minority, but they convey the message of Linux being suitable in neither desktop nor server roles.

If you are going to promote and explore Linux, you need to use it. If you don't know what good, open source tools for Linux exist to create and manage Web sites (such as Bluefish, Quanta Plus, and PHP), then ask around. Ditto for mail user agents: ask around, and you'll hear about excellent native-Linux mailers such as Mutt, Sylpheed, KMail, Mahogany, Balsa, Post Office, Aethera, Evolution, Pronto, and Spruce. Ditto for mailing list hosting. It's just unbelievably feeble and lame to have eGroups or some other "free" commercial service run your mailing list when GNU Mailman comes already set up and working on major Linux distributions, complete with automatic Web archiving and Web-based administration – plus you can even add to it mnoGoSearch as an archive search engine, if you wish.

Don't volunteer to look like losers in public. As the saying goes, a LUG needs to "eat its own dog food."

Addendum #1: A Note about Parent Groups

Many a LUG has gotten started as a subgroup of a more established parent, usually a general purpose computer user group. For example, SVLUG (*http://www.svlug.org/*), one of the world's largest LUGs, is technically a SIG (Special Interest Group) of the Silicon Valley Computer Society (SVCS, *http://www.svlug.org/~svcs/*).

The advantage to such an arrangement is that you can gain insurance coverage, incorporation, and acknowledged nonprofit tax status without having to handle the paperwork and expense of those efforts yourself. Depending on the people involved, the

It's painfully grotesque to see so-called Linux user groups mailing out announcements using MS-Outlook.

A LUG can operate with expenses approaching zero.

relationship can also create genuine symbiosis (such as SVLUG providing free Internet services for SVCS).

There are also offsetting disadvantages: established PC user groups are often in decline, and they tend to be run by people devoted to legacy proprietary OSes who have no understanding of Linux or open source software. The potential for culture clash is a serious one, and the odds are that your LUG members will have little interest in the parent group's other functions.

Old-line PC user groups tend to have annual dues of US $40 and up, for all members, and often charge admission for their monthly meetings. They adopt this model in order to finance their paper-published newsletters, (sometimes) to rent meeting space, and to pay sundry administrative costs such as telephone and corporate-filings fees.

By contrast, a LUG can operate with expenses approaching zero. Its publications can be Web-based; notices can be sent via email instead of on flyers; meeting space can usually be gotten for free at ISPs, colleges, Linux-oriented companies, or other institutions friendly to Linux, and can therefore be free of charge to the public. Having thus arranged to have roughly zero revenues as well as expenses, there is little need for tax-exempt status or incorporation. About the only thing you forego without incorporation is insurance and liability protection, which shouldn't be a problem for modestly careful people.

So, the advantages of having a parent group are somewhat smaller than would appear at first glance. You may find the parent group trying to dictate your subgroup's policies, including the content and location of its Web pages, and unhappy with your members who disregard the parent group and fail to pay its membership dues. This has led some Linux SIGs to split off from their parents as independent LUGs. Others quickly become the tail that wags the dog, as with SVLUG and SVCS. Some groups achieve other long-term arrangements, with varying degrees of happiness on both sides.

Be aware of the issues and possible outcomes, in any event.

Addendum #2: LUG Checklist

The following checklist may be useful for your group, once established.

1. Web page:

a. Meetings:

[ ] Current meeting info? Is it prominent?
[ ] Day of the week? Beginning time? Ending time?
[ ] Double-checked day/date matches against a calendar?
[ ] Location? [ ] Map? [ ] Directions (car, public transit)? Parking tips?
[ ] Information on the next several meetings?
[ ] RSVP mailto link on meetings where this is needed?
[ ] Note that meetings are free and open to the public (if they are)?
[ ] If there's a special fee, is it disclosed next to the event listing?
[ ] If location / time / date formula has changed recently, is this noted prominently?
[ ] Have you checked for event conflicts with other nearby groups, or with holidays?

b. General:
[ ] Includes event date formulas (e.g., fourth Tuesdays)? Prominently?
[ ] Maintainer mailto link?
[ ] Last-revised date?

c. Periodically (maybe every quarter):
[ ] Checked all links on your site for dead links?
[ ] Checked your Web server's logs for pages requested but not found? (You'll want to put a referral page at that URL.)
[ ] Read all your Web content attentively for outdated content?

2. Other, periodical:

[ ] Reviewed/updated all LUG lists that have entries concerning your group?
[ ] Reviewed all sites that link to yours? Advised their webmasters of needed corrections?

Addendum #3: Additional Reading

You will certainly want to read Kendall G. Clark's "Linux User Group HOWTO"; Paul L. Rogers' "Linux Advocacy HOWTO" makes a good companion piece.

Don Marti's "Linuxmanship" essay should be required reading for all Linux activists.

Eric S. Raymond's "Conventions at Lightspeed" addresses the running of larger events by hacker-types.

Kenneth R. Kinder's "Linux Myth Dispeller" is a bit outdated, but still good.

The redoubtable Christopher B. Browne has similar pages.

The most recent version of this essay can be found at *http://linuxmafia.com/~rick/essays/newlug.html*.

Notes

1. SSH is a protocol for encrypted remote login (and inter-system file transfer) protecting both your login authentication and the subsequent session data against third-party eavesdropping. Despite needing supporting software at both ends (server and client), it is rapidly replacing Telnet, because the latter is horribly insecure and a leading means of account theft and system break-in. The best-known implementation is also named SSH. Client software is available for every consumer OS: see my list at *http://linuxmafia.com/pub/linux/security/ssh-clients*.

2. gcal, the GNU calendar program, is a simple perpetual calendar utility included on typical Linux systems. Typing "gcal" shows the current month, while "gcal 2000" shows the twelve months of that year. On some systems, it will be named "cal".

wiki, blog, zope, and other communicative grunts

A Quick Look at Some Modern Collaboration Tools

by Strata R. Chalup

President, VirtualNet; Starting as a Unisys 68K admin in 1983, Strata Chalup is now an IT project manager but allegedly has retained human qualities. Her mixed home network (Linux, Solaris, Windows) provides endless opportunties to stay current with hands-on tech.

strata@virtual.net

I've been using some interesting Web-based collaboration tools for about 18 months now, and have been meaning to write about them. Hal Pomeranz gave me a good excuse when he obliquely mentioned Wiki as a tool at LISA last December.[1] I heard several folks afterward saying "Wiki? What's that?" and thought to myself, "That's right, I never did finish writing that up!"

A few of these tools require you to use JavaScript on your client browser, and I will warn you when that is the case. Most, however, are fully Lynx compatible and can even be used with Netcat or Telnet to port 80 if you are really in a pinch. I will say that doing Web-based collaboration by cut-n-paste over a Telnet connection is not necessarily in the spirit of easy update and publishing that drives these tools. We're an odd bunch of ducks, though, and it's nice to know that if you absolutely have to do something like that, in many cases you can!

Patterning Toward Collaboration

"Wikiwiki" is colloquial Hawaiian for "quick." The full name of the first wiki was the WikiWikiWeb. Wikis were invented, largely in collaboration, by the object-oriented programming community. Dale Cunningham, a founder of the Portland Pattern Repository, has been credited with much of the design and implementation. I believe at this point that many of you are familiar with patterning as a design and/or soft-skill tool in programming. A capsule summary is that in the early '90s, many of the SmallTalk luminaries realized that they could apply to language design (and later, to software engineering) the concept of "patterns" captured by Christopher Alexander and the Berkeley Design Group in their seminal work *A Pattern Language*. While Alexander's work was strictly in physical architecture, i.e., buildings and landscape, the idea of meta-patterns was so intuitively useful that it has become an integral part of object-oriented language design and software reusability engineering. The Portland Pattern Repository was founded as a collaboration space for language and software pattern development.

In system administration, patterns would be the things that come out of an applied taxonomy, such as best practices or common failure modes. As work on system administration taxonomy continues, primarily pioneered by Geoff Halprin and Rob Kolstad, sysadmins will begin to join the patterning community and develop pattern repositories of our own. I gave a short talk at BayLISA in 1998 about sysadmin patterns and the pattern concept in general. My interest in wiki and similar collaboration tools springs from my desire to see the sysadmin community start to create our own patterns. It is my hope that the recent progress on taxonomy and growing a "Book of Knowledge" for our profession will get us to the stage where we can start explicitly patterning in a collaborative process during 2002. Some of the modifications I'm working on for AtisWiki are specifically oriented toward enabling this process, but I'm getting ahead of myself here. Let's define a wiki first, and then start ripping into it.

NOTE

1. During his acceptance speech for the SAGE Lifetime Achievement award – yay, Hal! Congrats! :)

The Wild and Wonky Wiki

So, what is a wiki? It's easy to confuse a wiki with many other things which it is not, so let's build up the definition in stages. In the most basic description, a wiki is an interactive Web-based authoring system. It is most often implemented as a series of CGI scripts acting on a shared document repository. The repository usually consists of flat ASCII files, which may or may not contain HTML or HTML-like tags specific to a particular wiki implementation. Often there is a source code control system behind it used for check-in and check-out by the scripts. On more advanced wiki environments, there is the concept of user authentication, automated tagging of updates (date, time, user), per-user locking of files, access control, and various degrees of granularity of read-modify-publish for individual documents or subtrees.

A wiki is an interactive Web-based authoring system.

This is indeed a somewhat pedestrian description, and if that were all there were to a wiki, we'd be bored and stop right here. Now for the next layer, which is indeed more interesting. What makes a wiki so fast and easy to use, that is, why is it "wikiwiki" (quick)? When a document is saved or displayed, the parsing rules are used as a shorthand to determine if something is a hypertext link or not. The parsing rules are generally configurable within any wiki implementation, but the most common rule is that a mixed-case word is a hyperlink. Thus "ServerConfiguration" would appear as a link, and "server configuration" or "serverconfiguration" would not.

Why is this useful? Because if there wasn't already a ServerConfiguration page, the wiki would now create one for you. The link to this blank page would be immediately distinguishable. Unlike a regular link, which is simply highlighted in traditional Web fashion, an uninitialized "blank page" link is modified with a configurable pattern, usually "_?" added to the link name. Thus our blank page would show up as "ServerConfiguration_?" when the saved document was redisplayed. There are also shorthands for common types of markup, like beginning a line with "*" or "**" to create a bulleted list. Types of markup supported vary by wiki implementation, of course.

Although the authoring system is Web-based, the documents are not necessarily HTML documents. Most wikis have options to include HTML code (and display it properly) on pages after editing them. Some will include several different markup systems, including ASCII text, and let you choose to which markup group the page is assigned. Choosing the wrong type for what is already on the page can result in a page that will not display. When I was running a 9/11 information site wiki this September, I had folks trying to edit the pages and changing the type to "plaintext," which nuked all the links, or to "transclude," which nuked ALL the HTML and left the page uneditable until one constructed an "Edit" URL by hand. The "Keep It Simple, Silly" principle is a good one and probably dictates that wikis won't attempt to sanity check or force page markup types anytime soon. In the meantime, experimenting in the wiki "SandBox" area and setting the page wrong can ruin your whole day, especially if you're a newbie. I consider that a serious conceptual flaw. Which brings us to . . .

Wiki Warts

A Maze of Twisty Little Passages, All Different

As you can readily see, the ability to grow a document repository in a very organic, ad hoc fashion can easily turn into a double-edged sword. I'm an AwldPhart, and I like to see some kind of master index or master map to EveryDarnPageinThisWiki; unfortunately, most wikis do not support such a thing. The best one can do is to go back, often way, way, waaaayyy back, in the RecentChanges page. As its name implies, it is updated when any document in the repository is . In most wikis, it is created automatically, but

The WikiWay, as practiced in the patterning and OO communities using wiki, is anathema to control freaks.

not rotated automatically, so it is either very very long or has been hand-maintained and moved into month and year pages. Obviously, then, RecentChanges is also a valid wiki page, meaning that anyone can go in and screw it up, excuse me, edit it by hand.

A modification to some wikis is the keeping of a parallel index, essentially a content index as well as a RecentChanges. I'd like to see mods that kept a MasterIndex by date, as well as a user-tweakable ContentIndex. There's no reason why a wiki can't update multiple pages on a save command, although the more "shared" or "master" pages, the more likely one is to run into locking or halting issues. I think that as time goes on, features of threaded mailing lists will find their way into wikis, such as having all pages by a particular author linked into an Author page. This is lagging, as we will see later, because the concept of registered users and authentication is not widely practiced in the wiki world.

Who Moved My Cheese?

The ability (and tacit permission) of others to edit one's content, rearrange it, and generally move it around can induce a white-hot stellar phoenix reaction in those of us accustomed to having info stay where we put it. The WikiWay, as practiced in the patterning and OO communities using wiki, is anathema to control freaks. Can all two or three of the sysadmins out there who are not control freaks please raise your hands? Exactly.

What's the fix for this? None, on a public wiki belonging to another community! Content yourself with an RCS- or CVS-capable wiki, of which there are several, and know you can pull your info out of the Backups section of the document page. The other way, which irritates the "philosophically correct" wikians, is to use some combination of directory permissions and modified wiki scripts to enforce an author/commentator split between content updaters.

With Stone Knives and Bear Skins . . .

Another aspect of wiki use which will put some folks off is the lack of a real editing environment. The wiki Edit page in most implementations is simply a free-form text box which is pre-loaded with the existing page content. If you are doing markup in HTML, you're editing the tags by hand. This is not necessarily a problem for most of us, myself included, who still do a lot of HTML in vi. You will find that it grows very tedious when updating a large page. This is especially true when your text is pasted in from elsewhere and you don't realize it contains things that could be taken as WikiFormat: spaces and asterisks in a traceroute output, for instance, parsing as nested <UL> type lists.

Some wikis (such as TWiki) include a Preview button so you can see if your edits look reasonable before you commit. Actually, those which include a Preview button sometimes make that the only button, forcing you to do a preview step before being able to Save. That gets annoying very very quickly.

Where There's a Will, There's a Wiki

One thing which has kept many of us from joining the "Web revolution" with any enthusiasm is a mind-set which embraces the command line, two-way ASCII communication, and keeping things scriptable and simple. Of all of the tools we're going to mention today, the wiki is the most UNIX-philosophy friendly and easily tweakable in your scripting language of choice. While there are many wiki implementations in "newfangled" immersive environments like Squeak or Zope, the majority out there are Perl, Tcl, and Python. There are even a few done in C, suitable for revamping and cus-

tomization by the most stubborn (or overcommitted) fossils who haven't picked up much Perl yet. Who, me?

My preferred wiki at this point is AtisWiki, a fairly simple variant implemented in Perl as CGI scripts. Wiki variants are well suited to becoming Apache modules. I have found AtisWiki is fairly efficient when run under the mod_perl facility in Apache. I am not alone – there is now a UserModWiki, currently at version 0.92, which started with AtisWiki and made it an Apache module. Note that AtisWiki uses RCS natively, yet file locking is not strongly implemented – the HTTPd user is checking out the files, and there is currently no concept of sessionization that I can see in AtisWiki. I'm hoping that the UserModWiki developers have this on their list of features!

I still prefer AtisWiki for my own applications, though, because it is so straightforward and easy to customize. I am planning to integrate an alternate-channel publishing step into the AtisWiki "save" process. Instead of only updating the viewable Web page, one could also trigger additional actions upon a successful save. These actions would be based upon the (valid HTML) meta-tags in the document. The integration of existing Perl code for an RSS feed would be a tremendously handy application for a wiki, especially given the traditionally close coupling between the Webstream publishing communities and the wiki communities. In fact, it's something of a mystery to me why this does not already exist.

RSS is Rich Site Survey protocol, essentially a "headline news" format which supports keywords and meta info. Developed originally by Netscape in ancient times (i.e., the mid-'90s), it largely languished, but its potential is vast. A minor squabble over the future of RSS, and a brief interlude of incompatible protocols, has settled out nicely. RSS is a wonderful broadcast medium for things like system error messages or logging info and alerts, although it is currently used almost exclusively in a Web publishing context right now. Check out Jeff Barr's Headline Viewer tool, or the Syndic8.com site for aggregating and rating RSS feeds. Try using Headline Viewer or another RSS infeed viewer to get your SlashDot, FreshMeat, and similar news. You may never go back to those giant page views!

Another variant which has great appeal for me is DolphinWiki, implemented in C. It's currently a "closed" source wiki, which in this case means you have to ask the developer for a copy. He's a nice guy, has written a nice wiki, and will send you a copy if you ask nicely! DolphinWiki has some features I especially like, most notably closer tracking of document changes than AtisWiki, and some of the navigation/auto-index features I mentioned in my wish list above.

BayLISA is currently using a wiki variant called TWiki, which comes with some interesting presets for an office or work-group collaboration. We keep our speaker prospects and schedules in the wiki, and have been adding "legacy knowledge" culled from mailing list archives and other missives that have had no clear home. The ad hoc nature of the wiki makes it easy for one of us to spend an hour or so on a slow afternoon putting up a document area with useful bits that previously had no real "home." It's a lot more intuitive to make a page and add to it by creating other pages, like MeetingOpenChecklist, than it is to create subdirectories in a file tree and try cute naming games with filenames to try to convey the same information. Since we authenticate to log into the TWiki, it tags the files with the creator, so we know who to ask if something is not clear.

Of course, since TWiki (and all the other wiki implementations I have looked at) has no concept of static Web pages, the wiki info and the baylisa.org pages routinely get out of sync and must be fixed by hand. This is particularly irksome since we have a

RSS is a wonderful broadcast medium for things like system error messages or logging info and alerts.

Zope is, depending on whom you talk to, a toolkit, a language, a philosophy, a way of life, all things to all people.

makefile and script that update the speakers list, the library files, and similar info. As I mentioned above, when we finally combine wiki interactive document creation with a push/publish model à la Blogger (see below), we will *really* have the next generation of collaboration services.

You may be interested to know that there are more co-development projects associated with sysadmin and IT for TWiki than for any of the other wiki implementations I have examined. One which may be of particular interest to *;login:* readers is the TicketWiki, a trouble-ticket system implemented using TWiki. One can create normal wiki pages, or new tickets, and there are some goodies for tracking tickets and modifying them, and enough customization hooks to let you put on a fuzzy suit and stick yourself onto the wall like Velcro.

But Wait, There's More

Blogging: Blogger, by Pyra

Blogger is a Web publishing service which lets you create a page template and then add arbitrary text and markup to it via a publishing bookmark in your browser. The bookmark itself is a script which takes whatever you've selected on a browser window, plus the URL of the window itself, and uses that to fill in the template, then plunks you in an edit form via the site. You've undoubtedly seen many Web logs in the news since 9/11, so are familiar with the format. I keep several (neglected) public blogs, and also some private blogs where I keep notes to myself. When I'm scanning through a lot of Google or AlltheWeb searches to learn about some technical problem or tool, Blogger lets me pull the important excerpts from a page and capture them, along with the URL and my commentary on what I'm seeing. It's a great way to create annotated catalogs of resources as well.

Easy to use, easy to publish, easy easy easy. I love Blogger, though I don't use it as much as I'd like these days. Why? You have to use JavaScript, for one, and it updates via FTP for another. Wups. The smarts don't reside on a server you control yourself, but that's not a fatal flaw for my application. On the bright side, the content lives where you want it, on their site or on yours, and the service is free. Even if blogger.com were to go poof, you'd at least still have all your pages, and might be able to whip up some kind of replica on your own.

Immediacy is the core of blogging, yet an email-based blog would be very handy indeed. How would it be different from a standard mailing list archive? The shiny candy-coating formatting around the goodies, plus whatever you want to build into it. Maybe someone will build this as a gateway service. The first blogger add-on of which I'm aware debuted recently, namely "BlogSkins,", a forum for swapping blog formatting templates. Go team!

Zope! The Python and Squeak people . . .

The folks who love Zope are off on FreshMeat building all kinds of way cool things. Zope is, depending on whom you talk to, a toolkit, a language, a philosophy, a way of life, all things to all people. Okay, it's not that bad yet, but there's a lot here to dig into. If I had to do a one line summary, I'd say it's a Web/markup language in the same way that Perl is a string manipulation language, but that doesn't do it justice.

I'm not much of a coder these days, and tend to be rusty at learning new languages, so I have little personal experience with Zope beyond downloading a few toolkits and saying, "Whoa, cool!" Oh yes, and listening to my Web publishing friends rave about it.

If you are into Python, or remember Common Lisp with nostalgia, or are bursting with the joy of Squeak, then Zope is the next new toy for you.

The Zope user community can't be beat – they are warm, welcoming, and reach out to newbies in a variety of formats. Everything from Web logs to ZDP (the Zope Documentation Project) is out there to help you get up to speed.

Scripting Radio: Radio Userland, by Dave Winer and friends

Radio Userland is an immersive scripting and publishing environment powered by Manila, the same engine used for Dave Winer's commercial authoring environment called Frontier. I have found it rather impenetrable so far, but many programming-oriented friends have enthusiastically embraced it. Available only in Windows and Mac for most of its evolution, Radio Userland has features that include native use of XML, extensible scripting, built-in push authoring and blog-equivalent tools, private file-sharing, and more.

Drawbacks include a support community (especially the author) that seems to be inclined to be a bit defensive, and the current lack of any UNIX versions. Linux is supposedly "in the works," and it's a good idea to check for updates frequently. I think that checking out Radio will inspire folks to make Zope be all that it can be, but some of you may say, "Hey, Radio is a better fit for me." Dave, don't hate me! :-)

No More Mailing List Drift! QuickTopic, by Steve Yost

This is not an immersive publishing tool like the others, but it is a handy ad hoc publishing environment that complements email and is such a neat idea that I mention it here. Formerly known as "Take It Offline," Steve's system allows you to authenticate, start a topic, and then notify the mailing list or individuals that the topic exists. You can read the topics on the Web, or be notified by email of new postings. The emails from QuickTopic have routing headers constructed on the fly to funnel replies back to the topic.

Right now the service is free, and topics persist somewhat indefinitely, though ancient topics may someday be cleared to create more room. QT is extremely handy in keeping a mailing list focused while allowing folks to participate in specific off-topic threads with the rest of the mailing list community – at least, the ones who are interested! The system tracks all of your subscribed topics for you so that you can aggregate them onto a single page for browsing, although most folks get updates by email.

Why isn't this just Yahoo Groups? Because the focus and design is on a temporary home for ephemeral information, in conjunction with a mailing list community. Plus being completely free, voluntarily user-supported, and (so far) without advertising. QuickTopic is very lightweight and is not trying to be the kitchen sink. Your mileage may vary, but I find this handy enough to mention.

And was there a point to all this?

You bet! While most of us are on the cutting edge of technology in our own fields and specialties, the way we sysadmins communicate, as a profession and amongst ourselves, tends to be more in the form of jungle drums or carrier pigeon! There are a lot of amazing tools out there that we could be using in our community, and I hope that I've brought a few of them to your attention. Let's go out there and have a little fun.

BIBLIOGRAPHY

AtisWiki:
http://www.ira.uka.de/~marcus/AtisWiki.html

Blogger, by Pyra:
http://www.blogger.com/

BlogSkins:
http://www.blogskins.com/

Dolphin, a C wiki:
http://www.object-arts.com/wiki/html/Dolphin/FrontPage.htm

QuickTopic:
http://www.quicktopic.com/

Radio Userland:
http://radio.userland.com/

Radio Userland – Dave Winer on publishing:
http://www.scripting.net/

RSS – Jeff Barr, RSS Godling (hi, Jeff!):
http://jeffbarr.editthispage.com/

RSS – O'Reilly's RSS info site:
http://www.oreillynet.com/pub/a/rss/

RSS project – NewsClipper:
http://newsclipper.sourceforge.net/

RSS uses discussion:
http://discuss.userland.com/msgReader$10115

The Wiki Way: Quick Collaboration on the Web, Ward Cunningham & Bo Leuf, Addison Wesley.

TicketWiki:
http://TWiki.org/cgi-bin/view/Codev/TicketWiki

TWiki main page:
http://TWiki.org/

UseModWiki: *http://www.usemod.com/cgi-bin/wiki.pl?UseModWiki*

WikiWeb entry point:
http://c2.com/cgi/wiki

Zope Development Home:
http://dev.zope.org/

Zope Documentation Project:
http://zdp.zope.org/

Zope newbies site:
http://www.zopenewbies.net/

ZopeLabs (Zope cookbooks):
http://www.zopelabs.com/

planning a model retirement

by Ray Swartz

Ray Swartz started financial planning in 1988. Thanks to good luck, a good plan, and hard work he enjoyed an extended vacation for all of 2001. He doesn't know how long this will last, but his model tells him not to worry about the near future. He hopes the model is right.

ray@trainingonline.net

In my last column, "When Do You Plan To Retire?" I stated that "to have a nice retirement you have to plan to have a nice retirement." As such, I suggested you collect data on your expenses, income, and investments, and begin designing your future.

What are you supposed to do after collecting all this information? Just generating the numbers and stating your retirement plans is an extremely valuable exercise, even if that is all you do. However, once you have the numbers, it would be nice to know how close you are to achieving your retirement goals and what specific steps you should take now and in the future to get where you want to go financially.

One option is to bundle all this information up and deliver it to your financial advisor. From it, your advisor will generate a large notebook full of charts, graphs, and an analysis of your ability to finance the retirement you've designed. The most important part of this report is the recommendations that you should follow to reach your retirement goals. Typically, these "financial plans" cost a few hundred dollars.

These reports often take into account estate planning, taxes, investment returns, inflation rates, college costs, and other financial issues. Generally, they are comprehensive and very useful. This is the first step I took when I started my financial planning back in 1988.

While my report was quite informative, it had certain limitations. Many assumptions are needed to complete the planning process, and as time went on some of these assumptions didn't turn out to be accurate. For example, the planning process required me to make estimates of my future income. But, since I worked for myself there was no way to accurately predict what my income would be in coming years. Thus, the recommendations became less useful as the years went by and my income diverged from my predictions.

As my plan became out-of-date, I wondered what I should do? While I could have had another plan prepared, financial reports cost money. I didn't want to have to generate a new plan every few years just to keep updating my income and expense numbers.

What's more, the generated financial plan was completely static. There was no way to get answers to "what if" questions: for example, if I have some extra cash at the end of the year should I make extra mortgage payments, invest the money, or donate to charity?

Planning Your Own Future

After reviewing my planning report, I realized that I could create a personalized version of it in a spreadsheet program. I started out with a simple model that began with my current holdings and expenses and then calculated what happened over time using this equation:

cash next year = cash this year + investment gains + income - expenses - taxes.

This simplified set of calculations allowed me to see where I was headed if I kept to my earning and spending habits. I could also make a few changes and instantly see the result. As things arose, I fiddled more with the sheet.

Over time, the sophistication of the model grew. I added a way to calculate federal and state taxes using tax tables instead of estimating it. I split my investments into taxable and non-taxable (stocks, IRA, life insurance, etc.). I added a calculation for my house

mortgage. I separated my expenses into those that were tax deductible and those that weren't and further refined them to those that would be subject to the inflation rate. In fact, every time a new financial twist entered my life, some new columns were added to the spreadsheet.

Now, whenever I wanted to perform "what if" analysis, I simply changed a few numbers and saw what happened. This greatly assisted many decisions that I had to make about how to handle my money.

As the model progressed, it involved more interaction among the data, and the limitations of the spreadsheet paradigm began to get in the way. Handling complex logical tests was difficult, error-prone, and cumbersome. For example, when I ran out of cash, where did the money to pay my expenses come from? Was it from my IRA, insurance, after-tax investments, or home-equity line of credit? There are various tax consequences depending on which one is used, and writing a single-cell logical test with variable names like A23 got to be too much.

In addition, financial calculations involve auto-correlated data. That is, a year's financial numbers depend on one another. For example, the amount of cash available to pay for expenses depends on this year's income minus this year's expenses. But, suppose that after calculating how much money is required to pay expenses, you discover that you need more cash than you have. Getting access to additional cash will change the expense amount due to interest on borrowed money or taxes from sold investments. Unless I was very careful, my model would get out of phase and give me bogus answers. I began spending lots of time reality testing my, now quite large, planning spreadsheet.

Another problem involved the model's time frame. I had chosen years as my time unit. In essence, my model transacted all my financial business once a year. This meant that money I spent in January wasn't deducted until December 31, and I was allocated interest on it for the entire year, even though in reality, that money had already been spent. Moving the model to a monthly frame to eliminate this foolishness was simply too hard to do.

Like the paper report, the spreadsheet model was static. The spreadsheet had to have all the numbers before it could do the calculations. That meant that I had to supply data for future values that were not knowable. Examples are the inflation rate, investment returns, and interest rates. I took the best guesses I could, but there was no way to factor in the uncertainty inherent in them. If I depended on an 8% return and the market tanked, I could be in serious trouble.

Even with all these shortcomings, I found the spreadsheet model very useful. I based many financial decisions on its predictions. What's more, I learned a great deal about financial planning from designing, debugging, writing, and maintaining it.

Creating a Custom Plan

As some major personal and financial decisions loomed and I began to depend more and more on this model, I was concerned that its shortcomings were beginning to skew my planning. The only way to create a model of the world as I saw it was to program the whole thing from scratch to react just the way I thought it should. I started this effort in mid-2000.

By this time I had a long list of requirements necessary to make the programming effort worthwhile. I wanted this new model to:

- Enable me to keep track of my investments and cash balances.
- Update investment values from the Internet on a daily basis.
- Enable inflation, investment returns, and interest rates to vary over time.
- Produce different scenarios that represented the worst, average, and best futures I could expect.

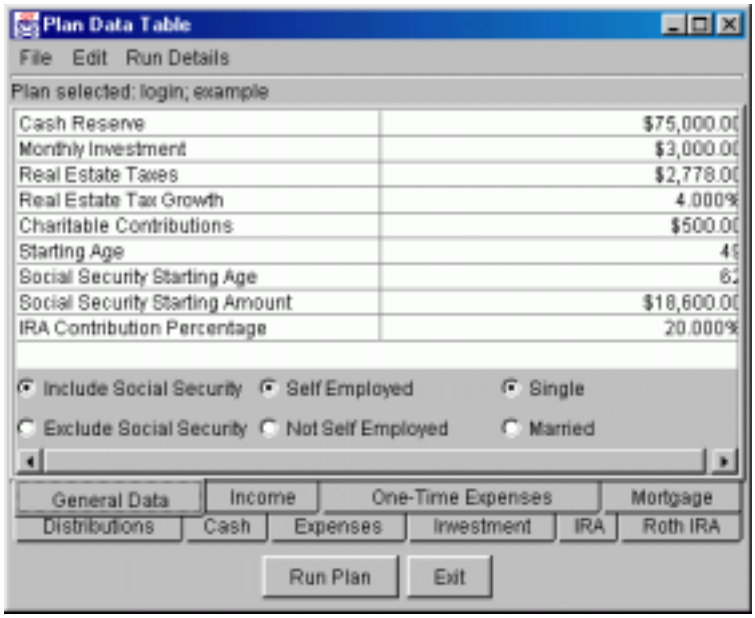


Figure 1: FinPlan General Data

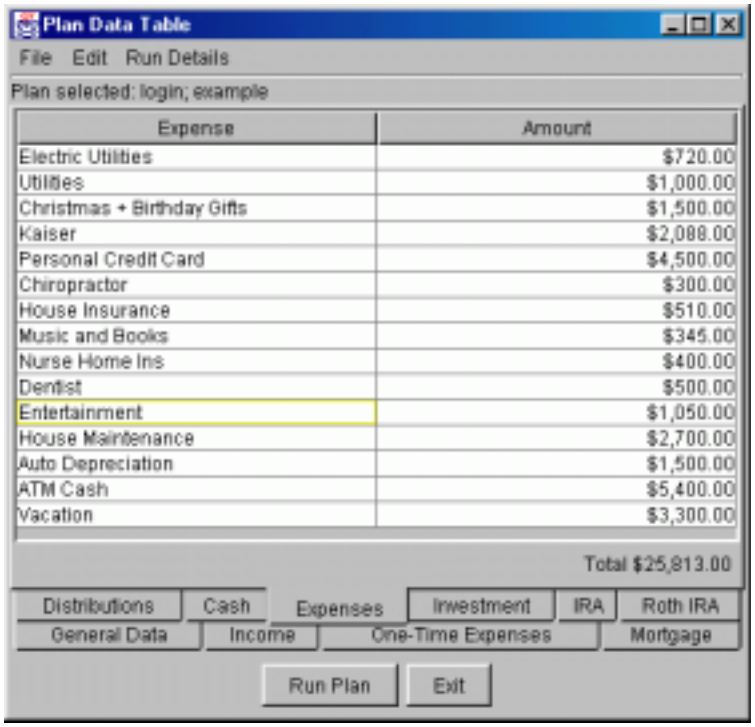


Figure 2: FinPlan Expenses

- Enable me to explicitly state how my assets were to be consumed as they ran out.
- Enable me to apply taxes accurately over time.
- Enable me to have a monthly time frame.
- Enable me to change data as easily as I did in the spreadsheet model.

I considered several different ways to write this program. To ease input, I thought about writing it with a Web front end to handle input and a Perl back end to do all the calculations. While this would have worked, it required a suite of programs to generate and process all the Web pages necessary to hold and transfer all the data. It also required me to install a Web server on my PC, which I didn't want to do.

Another thought was to write the code using programming tools connected to an SQL database. I decided against this to minimize the execution requirements of the code, which might prevent me from using this code in a different location or distributing it to friends.

Finally, I decided to use Java and write the interface with Swing. The resulting program, which I call FinPlan uses a tabbed main window (See Figure 1).

Each tab represents a multi-column table that serves as both display and input. Those tables holding accumulating values keep running totals at the bottom. Figure 2 shows the yearly expense table. Note the running total at the bottom of the right-hand column.

These data windows allow me to see at a glance the current assumptions I am making and the value of my current holdings, to update the data in these tables interactively, and to quickly modify something and see what effect it might have on my financial situation. Note that the values in these tables are what make up my financial plan.

How Random Can You Get?

I wrote FinPlan to be a dynamic model. That is, I didn't want to get just the one answer the spreadsheet could give me. Instead, I wanted to randomly choose values for the unknowable variables – investment return, interest rate, inflation rate – and then see what effect they had as they moved in the specified range. Figure 3 shows the distributions table. Here, I am using a Gaussian (normal) distribution and I've specified the mean and standard deviation for investment return, interest, and inflation rates.

In order to see the effect of my current plan, I start with the numbers in the tables. Then, on a monthly basis, I factor in income, investment returns, interest, social security income, general expenses (modified by inflation), one-time expenses, mortgage payments, and taxes. I continue this until I run out of all money (cash, investments, IRA, and Roth IRA). I repeat this process 5000 times. The number of iterations is user-specified. An entire run takes about five seconds on my machine (Pentium III PC).

Figure 4 shows the output data displayed after a run of the model. The output window is a three-column summary of when my money ran out during each of the 5000 iterations. By the way, this plan is completely made up and, aside from my age, the data are not meaningful.

The first column shows the year, my age, how many times this was the "money-out" year, and the percentage. The second column is a cumulative percentage. The third column is the average investment return, inflation rate, and interest rate for the plans that ended that year. There is complete run summary data at the bottom. The highlighted years represent 10% and 50% cumulatively. I cut off each plan at 100 years of age.

This run of an example plan tells me that the earliest I ran out of money was at age 74 and that my investment return over the life of that iteration was 5.3% with an inflation rate of 4.0%. This outcome occurred once in 5000 iterations; it is unlikely to happen in real life.

While FinPlan has been an enormous help in making financial decisions, it has drawbacks. First, it treats all investments the same. Thus, I get the same return on all my investments regardless of what they are in. This is not realistic. The same thing is true of cash in various types of accounts. Also, complicated financial instruments, like life insurance, aren't accurately represented.

There is also a personal drawback. With the push of a button, I can get a daily update on how all of my investments are doing and how the future plays out as a result. When the market drops, I get to see its effect on my entire financial future. I've learned that too much information can be just as useless as not enough!

You don't have to go to the effort I have in creating a financial planning application to enjoy the benefits of doing your own financial planning calculations. Whether you hire a financial planner, use a simple spreadsheet, or write a custom program, mapping out how your current plan plays out in the coming years is a powerful way to ensure that you are building the future you want.

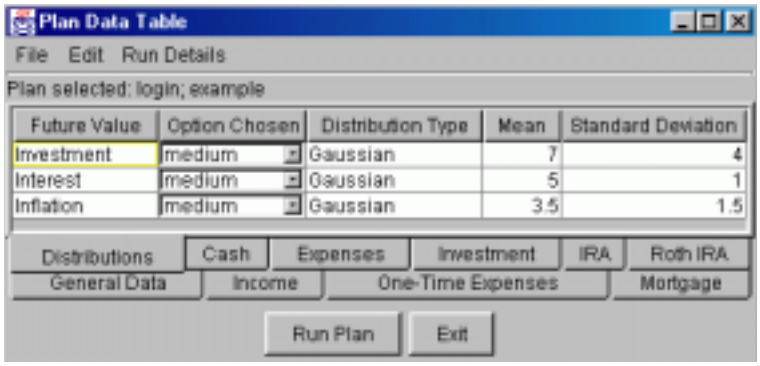


Figure 3: FinPlan Distribution

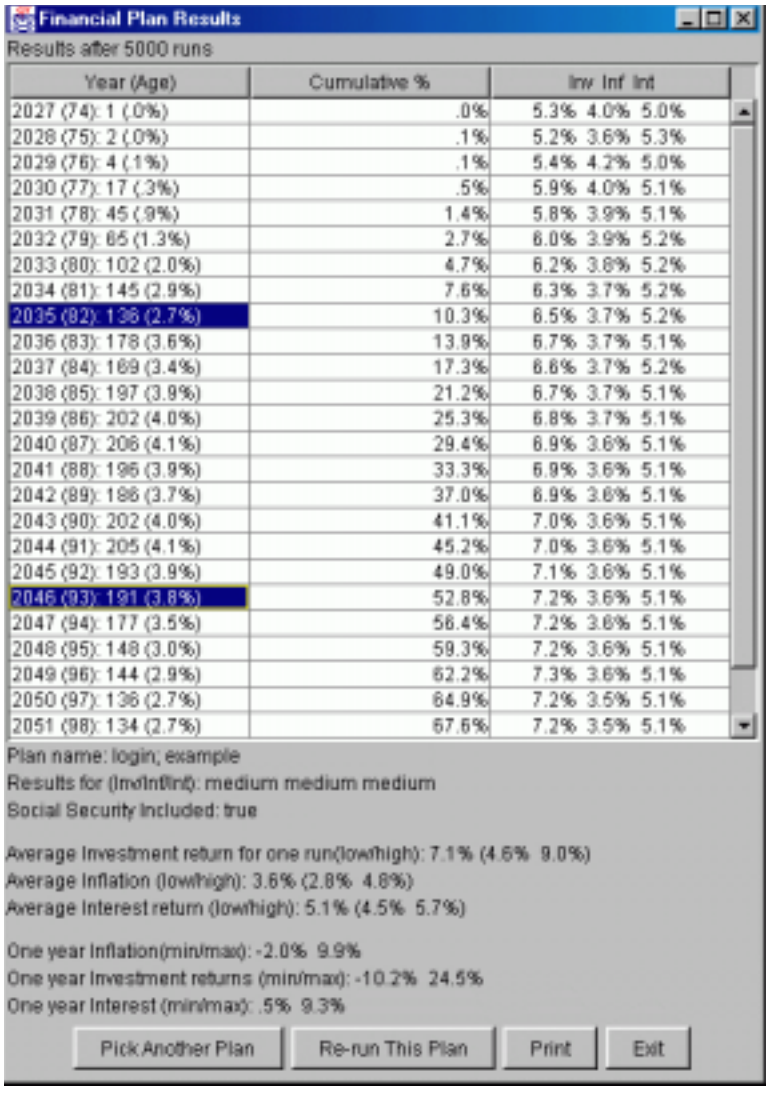


Figure 4: FinPlan Output Summary

data privacy

Life Just keeps Getting More Complicated

by John Nicholson

John Nicholson is an attorney in the Technology Group of the firm of Shaw Pittman in Washington, D.C. He focuses on technology outsourcing, application development and system implementation.

John.Nicholson@ShawPittman.com

Not too long ago, the public started worrying about "privacy" related to the Internet.[1] They weren't really sure what it meant, or how it differed from privacy in the real world, but they had an opinion, anyway.

Sensing an opportunity/need for regulation, governments started getting involved. The US government enacted laws like the Children's Online Privacy Protection Act (COPPA), the Gramm-Leach-Bliley Act (GLBA), and the Health Insurance Portability and Accountability Act (HIPAA), and the European Union implemented its own Privacy Directive.

Over time, companies also put up "privacy policies" on their Web sites. Sometimes these policies were written by lawyers, sometimes by a marketing intern, and sometimes they were just copied from another Web site. Frequently, they were posted and ignored by the business itself. Surprisingly, however, a large number of companies do not yet have a privacy policy. The purpose of this article is to discuss what should be in a privacy policy and to discuss some of the recent developments in privacy regulation that will make your life more complicated.

Privacy Basics

The essence of data privacy involves four fundamental principles: notice, choice, access, and security. If your company doesn't already have a privacy policy, it should develop one that is accessible from your home page (preferably with a clear and easily identified link on your home page and also on each page where users can provide information to you). The purpose of this section is to provide examples of the type of language that should (and, occasionally, should not) be in your privacy policy.

Privacy Policy Part 1

Companies frequently like to include some type of introduction in their privacy policy, and this is the first place where you can get yourself into trouble. For example, one Web site opens its privacy policy with the following:

"We will ensure that our relationship is as exclusive as you want it to be."

This is a fairly sweeping statement and is the kind of language that marketing executives use to give their lawyers high blood pressure. From a legal perspective, "ensuring" something is a very high standard, especially when what you have promised to "ensure" is that you will comply with the potentially subjective and variable wishes of a consumer. Other privacy policies have promised to use "best efforts" to protect consumer information. When technical people say "best efforts," they mean that they will try to do something, and if they can do it – great. From a legal standpoint, that actually describes what are known as "reasonable efforts." On the other hand, in legal terms, "best efforts" means that you will use ALL efforts, regardless of cost. This type of distinction is one of the big reasons why your privacy policy should be reviewed by your legal department.

The introduction should also specify that your company's privacy policy can be modified at your discretion without prior notice to consumers, and that any changes to the policy will apply both to all information gathered after the change and retroactively to

all information already in your company's possession. Although it has not been settled whether or not such language will be effective, having it at least provides notice to consumers that retroactivity is your policy.

An example of good introductory language is:

> [*Web site*] supports a general policy of openness about personal data collection and use. We have adopted and implemented this Privacy Policy as part of our commitment to protecting your personal information from misuse. Although this Privacy Policy is not intended as a contract or as creating any legal rights, it does represent [*Web site*'s] current policies with regard to personal data collection and use. We reserve the right to modify this policy at any time without notice, and any changes to this policy will apply to all information then in [*Web site*'s] possession or acquired after the date of such change.

PrivacyPolicy Part 2: Contents

The policy should explain your company's policies with regard to:

NOTICE: WHAT INFORMATION YOU COLLECT

Early in your privacy policy, you should tell users in a broad sense what information you collect. If you collect personally identifying information such as name, email address, mailing address, or even demographic data such as zip code, age, income, etc., you should notify the user.

If you use tracking technologies such as cookies or Web bugs, disclose it, but explain what they are and why they are used. The following text is an example:

> From time to time, we may use the standard 'cookie' feature of major browser applications that allows us to store data about your visit. Cookies help us learn which areas of our site are useful and which areas need improvement. We do not set any personally identifiable information in cookies, nor do we employ any data capture mechanisms on our Web site other than cookies. You may configure your Web browser to prevent cookies from being set on your computer.

Some Web sites even actually explain, at a very basic level, the difference between session cookies and persistent cookies in order to educate users.

Do not try to make these technologies sound more palatable to consumers by using euphemisms that may end up being technically inaccurate. For example, one Web site tells consumers that it uses "pixels," by which it means Web bugs, to track how consumers move through the site. This kind of technical inaccuracy could lead the Federal Trade Commission (FTC) to believe that the owner of that Web site was trying to mislead consumers. If you (or your marketing department) feel strongly about not using the term "Web bugs," then at least be accurate and call them "single pixel images" or "clear GIFs" and explain how they work.

NOTICE: WHAT YOU DO WITH THE INFORMATION

You should clearly explain what you are going to do with the information. If you plan to share the information with third parties or other partners, say so. If you plan to use the information for marketing purposes, say so and say how. Just as importantly, if you may use or disclose the information, do not say you won't. Failure to comply with a

Some Web sites even actually explain, at a very basic level, the difference between session cookies and persistent cookies in order to educate users.

The user should have the option to decide whether the information will be used or shared.

published privacy policy is exactly the kind of action that would attract the FTC's interest.

Here is a sample of some language used by one Web site that only collects automatic traffic data:

> In general, our site automatically gathers certain usage information like the numbers and frequency of visitors to [*Web site*] and its areas. We only use such data in the aggregate. This aggregate data helps us determine how much our customers use parts of the site, so we can improve our site to ensure that it is as appealing as we can make it for as many of you as possible. We also may provide statistical 'ratings' information, but never information about you personally, to our partners about how our members, collectively, use [*Web site*].

Children under 13

You should confirm with your lawyers that your Web site is in compliance with the regulations related to the Children's Online Privacy Protection Act of 1998 (COPPA).[2] If any portion of your Web site is directed toward children under 13, including providing products or services directed to children under 13, then you need to comply with the COPPA regulations. On the other hand, if your Web site is not intended for children under 13, the following is an example of the type of language you could include:

> [*Web site*] respects the sensitive nature of children's privacy online. We are a general audience site and do not direct any of our content specifically at children under thirteen (13) years of age. We have implemented procedures so that if we obtain actual knowledge of a child's personal information, we will take steps to delete that information. In addition, if [*Web site*] knows that a user is under age 13, we will NOT:
>
> 1. deliberately collect online contact information from that user without prior parental consent, except where used to respond directly to the child's request;
> 2. deliberately collect personally identifiable offline contact information from that user without prior parental consent;
> 3. deliberately give that user the ability to publicly post or otherwise distribute personally identifiable information without prior parental consent;
> 4. distribute to third parties any personally identifiable information of that user without prior parental consent;
> 5. entice that user, by the prospect of a special game, prize or other activity, to divulge more information than is needed to participate in the activity.

Be sure, however, that if you use this type of language you comply with it. Failure to comply with provisions like this when posted on your Web site could lead to unwanted attention from the FTC. So far, the FTC has taken action and levied fines against several Web sites that have violated COPPA and/or their own privacy policies:[3] the fines have ranged from $30,000[4] to $100,000.[5]

Choice

The user should have the option to decide whether the information will be used or shared.

You should include directions for how users can prevent you from providing their information to a third party, to remove their information from your database, or to stop receiving communications from you. You can set up an email address, URL or

other contact method. The following is an example of language you can use related to opting out:

> Opting Out
>
> [*Web site*] gives you the following options for contacting us to (i) prevent your information from being shared with third parties; (ii) stop receiving communications from us; (iii) remove your information from our database (which may prevent you from receiving our service); or (iv) stop receiving our service:
>
> 1. Send email to [opt out email address]
> 2. Visit [opt out URL]
> 3. Send postal mail to [opt out address]
> 4. Call [opt out phone number]

ACCESS

The user should have the ability to see what information you have collected and to correct it, if necessary.

You should include directions for users to review and modify information from your database. You can set up an email address, URL, or other contact method. The following is an example of language you can use related to users accessing and modifying information:

> Updating Your Information
>
> [*Web site*] gives you the following options for changing and modifying information previously provided:
>
> 1. Email [*corrections email address*]
> 2. Visit [*corrections URL*]
> 3. Send postal mail to [*corrections address*]
> 4. Call [*corrections phone number*]

Finally, you should refer any questions related to the privacy policy to a contact person:

> Contacting [*Web site*]
>
> If you have any questions about this privacy statement, the practices of this site, or your dealings with [*Web site*], please feel free to contact: [*contact information*].

SECURITY

You should have in place protections appropriate to the nature of the information collected; for example, health care or financial information would be expected to be better protected than an address list.[6]

Your Web site should specify that you have taken "reasonably appropriate security measures" to protect information in your possession and should generally describe the technologies you use for security (e.g., SSL). You are not required to (and should not) discuss in detail the security practices on your Web site. You should recognize, however, that in the event of a use or disclosure by an unauthorized person of information stored by you, your security policies and practices could come under scrutiny. Even if your practices are reasonable by current industry standards, in the event of a security breach, you could be criticized for not doing enough to protect your information.

The user should have the ability to see what information you have collected and to correct it, if necessary.

So far, dealing with privacy in the US has been complicated and expensive.

Security and incident response are areas where you should have a team of technology, legal, operations, and marketing personnel developing policies and procedures. Unless your operations and marketing people understand the implications of security, including potential legal/regulatory issues, and unless those same people believe that you understand their concerns about the impact of security on operations and customers, they will not cooperate with you, and your security efforts may be circumvented.

Life Gets More Complicated

So far, dealing with privacy in the US has been complicated and expensive. If your site caters to children, you have to deal with COPPA. If you are in the financial industry, you have to deal with the requirements of the GLBA.[7] If you are in the medical or heath insurance fields, you will be subject to the regulations being developed associated with HIPAA.[8] On top of these domestic laws, if you operate in or receive data from Europe, you may have to comply with the European Union Privacy Directive.[9]

So far, it is not clear how these various laws and regulations will evolve or interact. For example, what happens when US and EU privacy regulations conflict? Is a bank that provides outsourced billing for an HMO subject to HIPAA? What about credit card companies who process payments for medical services – do they have to determine whether a payment is medically related and treat that information differently from all other data? Should health insurance companies that provide financial services be subject to GLBA? The HIPAA regulations have not been finalized yet, so there may be even more surprises in store.

As if all that uncertainty weren't bad enough, dealing with privacy is getting worse. Other countries are beginning to pass privacy legislation similar to the EU Privacy Directive – notably Canada[10] and Australia.[11] In the US, furthermore, FTC Chairman Timothy Muris stated in a recent speech at the Privacy 2001 Conference in Cleveland, Ohio, that "privacy promises made offline should be held to the same standard as online privacy promises" and indicated that ensuring the security of sensitive information is fundamental to privacy, whether that information is collected online or offline.[12]

In a December 5, 2001, speech to the Promotional Marketing Association, the director of the FTC Bureau of Consumer Protection, Howard Beales, stated that a company's online privacy policy also applies to offline collection and the use of such offline information, unless the policy clearly limits its promise to the online realm. The FTC has begun looking more closely at whether companies are complying with their own privacy policies with regard to information collected online and offline, and whether companies are implementing security procedures that are sufficient to protect the information they collect.

It is not clear what the new FTC policy means for businesses, but it could make life very difficult – both operationally and in terms of technology. For example, you may need to create the ability to move all data collected offline into the area where your online-collected data is stored, so that consumers will be able to access that offline-collected data electronically. Your company may have data-sharing agreements in place regarding data collected offline, and those agreements may conflict with the privacy policy on your Web page. How will you make the people providing data offline aware of your privacy policy? Until the FTC's policy becomes clearer, your privacy policy on your Web page should clearly limit the application of the privacy policy to information gathered via the Web page.

Conclusion

Regardless of how some of the current issues in privacy law turn out, you should give a great deal of thought to how your company deals with information and data privacy. Your privacy policy is no longer something that you can just slap together (or poach from another Web site) and stick on your Web site. Your privacy policy needs to be carefully crafted, in cooperation with legal, operations, and marketing, to deal with your operations and your industry, and failure to pay attention could result in unwanted attention from the FTC, foreign regulators, and/or the media.

Your privacy policy is no longer something that you can just slap together.

Notes

1. This article provides general information and represents the author's views. It does not constitute legal advice and should not be used or taken as legal advice relating to any specific situation.

2. See Title XIII of PL 105-277, 15 USC 6501. See also, "Taming the Wild West: Laws and Regulations Governing the Technology Industry," *;login:*, vol. 25, no. 5, August 2000, pp. 43-49.

3. *http://www.ftc.gov/privacy/index.html.*

4. *http://www.ftc.gov/opa/2001/10/lisafrank.htm.*

5. *http://www.ftc.gov/opa/2001/04/girlslife.htm.*

6. HIPAA regulations, for example, include specific language and requirements related to the protection of health information.

7. For a discussion of the requirements under the GLBA, see the FTC's Web page at *http://www.ftc.gov/privacy/glbact*.

8. PL 104-191, Aug. 21, 1996.

9. See Gary Rothenbaugh and John D. Woodward, "Fact Sheet on the European Union Privacy Directive," *http://www.dss.state.ct.us/digital/eupriv.html*, and the US Department of Commerce's page at *http://ww.export.gov/safeharbor*.

10. Privacy Act 1980-81-82-83, c.111, Sch.Ii "1". See also, the Canadian Privacy Commissioner's Web site at *http://www.privcom.gc.ca/legislation/index_e.asp*.

11. Privacy Amendment (Private Sector) Act 2000 (Act No. 155 of 2000). See also, the Australian Attorney General's Web page at *http://www.law.gov.au/privacy*.

12. Timothy J. Muris, "Protecting Consumers' Privacy: 2002 and Beyond," October 4, 2001, *http://www.ftc.gov/speeches/muris/privisp1002.htm*.

the bookworm

by Peter H. Salus

Peter H. Salus is a member of the ACM, the Early English Text Society, and the Trollope Society, and is a life member of the American Oriental Society. He is Editorial Director at Matrix.net. He owns neither a dog nor a cat.

peter@matrix.net

BOOKS REVIEWED IN THIS COLUMN

TELECOMMUNICATIONS ESSENTIALS
LILLIAN GOLENIEWSKI
Boston: Addison-Wesley, 2002. Pp. 582.
ISBN 0-201-76032-0.

WIRELESS INTERNET
MARK BEAULIEU
Boston: Addison-Wesley, 2002. Pp. 632.
ISBN 0-201-73354-4.

RUBY IN A NUTSHELL
YUKIHIRO MATSUMOTO
Sebastopol, CA: O'Reilly & Associates, 2002.
Pp. 204. ISBN 0-596-00214-9.

WEB CONTENT MANAGEMENT
RUSSELL NAKANO
Boston: Addison-Wesley, 2002. Pp. 238.
ISBN 0-201-65782-1.

MAC OS X: THE MISSING MANUAL
DAVID POGUE
Sebastopol, CA: O'Reilly & Associates, 2002.
Pp. 583. ISBN 0-596-00082-0.

TROUBLESHOOTING TECHNIQUES OF THE SUCCESSFUL TECHNOLOGIST
STEVE LITT
NP: Troubleshooters.Com, 2001. Pp. 313.

In the February column, I lauded Rich Morin's publication effort, DOSSIER (Documenting Open Source Software for Industry, Education, and Research). At that time, I'd looked at the *Mail and Sendmail* volume. I've now looked at *Text* (which covers both RedHat and FreeBSD commands, as well as the standbys: groff (troff), eqn, pic, and tbl) and *Kernel* (which covers the FreeBSD kernel). Keep 'em comin', Rich!

Collect 'em all from: *www.ptf.com/ptf/dossier/*.

Telecoms

Goleniewski's book was a disappointment to me. It is nicely written but superficial to a point that I began looking for sections about stuff I knew, only to be disappointed. Its glibness can result in distortions, as in the brief paragraphs on the history of the Internet. There are no references nor is there a bibliography. Each chapter ends with a reference to a Web site. When I went there, it demanded registration information. I consider this a blatant marketing ploy (Ms Goleniewski runs a "seminar and e-learning" business) and didn't bother. Despite its title, the volume contains too few "essentials" for me to recommend it to anyone below the level of corporate vice president.

Unwired

Beaulieu's volume on wireless, on the other hand, was both interesting and valuable. Beaulieu discusses the concept of the wireless Internet, the language used in the discussion, and the basic concepts involved. His exposition of WAN, LAN, and PAN standards is quite good. He also talks about e-commerce and m-commerce (mobile commerce) servers. Read together with Wheeler, Milojicic, & Douglis, *Mobility* (1999) and Solomon's *Mobile IP* (1997), this yields a really good picture of where we're likely to go.

Ruby

I read *Ruby in a Nutshell* with great interest. I thought that Ruby was an interesting attempt at creating a language that, like the Second City's WFMFMT had "the best of everything worthwhile." Matsumoto has pulled together the features of a number of other scripting languages. Unfortunately, I think that the history of programming languages has shown us that being good is not the same as being successful. So, I think that Ruby most likely won't make it, because it's not enough better to make users of Java, Perl, and/or Python want to switch. (Ray Schneider differs with me here, and I have requested a full review of O'Reilly's's Ruby book from him to give readers a more "balanced" view.)

Web Stuff

Nakano's volume is highly nontechnical. But I think that it will prove useful to those who are trying to manage their Web sites at the same time the content is waxing at a furious pace. My major problem with the book was that so many items in the "executive summary" sections sounded like platitudes. (This may be my fault, for expecting content in something containing "executive"; it may be a real benefit, as Nakano recognized, that executives read only summaries; I'm not sure.) However, I think that the section in Chapter 3, on versioning, needs references to SCCS, RCS, and their friends.

book reviews

MacOS X

About 15 years ago, in Phoenix, I complained to Steve Jobs that I had found Mac "manuals" fairly useless where trying to actually locate information was concerned. Steve told me I wasn't supposed to "look things up" but to "just mouse around."

Well, if you are using a Mac and running OS X, you didn't find a book in the box. Pogue has written a big, fat manual that a user will find indispensable. I have a devoted colleague, running a cube, who nearly tore the book from my hands when it arrived.

I think it's really a fine job.

Troubleshooting

My guess is that no matter what aspect of computing you're involved in, you end up spending time troubleshooting. Litt has produced (in some sense) a really fine volume to serve as a guide.

Litt points out that most technologists are employed to solve technical problems and/or design new technology. Solving problems is "pure troubleshooting." Right. We all know that. But Litt goes on, outlines a respectable methodology, and provides a system which should enable the intelligent troubleshooter to work efficiently.

I enjoyed reading the book. Its single drawback is that it is 8.5 x 11 inches and stapled. Maybe there's an opportunity for a "real" publisher here. It's $42.50 and available from *http://troubleshooters.com/bookstore*.

A Concluding Note

When I look at the number of books I get about the Web or XML or Java or Perl or . . ., it makes me wonder just why most publishers are afflicted with rampant me-too-ism. In a more sensible industry, neither Morin nor Litt wouldbe in the publishing or distribution industries. And there might be fewer books on how to design Web pages or use Office 2000.

AGILE SOFTWARE DEVELOPMENT

ALISTAIR COCKBURN

Boston: Addison-Wesley, 2001. Pp. 304.
ISBN 0-201-49834-0

Reviewed by Raymond M. Schneider

ray@hackfoo.net

Get ready, we are going to start hearing more and more about agility when it comes to software development processes. *Agile Software Development* is one of two books anchoring an up-and-coming series.

Have you ever looked at your development process and thought it was bloated? Have you ever noticed people in projects suffering from argh-minutes? What is an argh-minute? Cockburn introduces us to the term "as the unit of measure for frustrating communication sessions." *Agile Software Development* brings the reader up to speed about adapting agility in development methodologies.

Agile Software Development is seemingly broken down into two distinct parts. The first three chapters acquaint the reader with the terminology and concepts that are being discussed. Levels of listening, failure and success modes of individuals, and convection currents of information are all core ideas developed in the first part of *Agile Software Development*. There are three levels of listening: following, detaching, and fluent. These levels are referred to by Cockburn throughout *Agile Software Development* in sections where readers' focus may differ depending on their listening level.

Methodologies, sweet spots, self-adaptation, and Crystal methods are discussed in the second half of *Agile Software Development*. Methodology is defined and explained in depth in Chapter 4. Extreme programming is examined, and ways to adjust it to the needs of a project are introduced. "Sweet spots" are characteristics of agile methods that work best. Cockburn gives examples of five different sweet spots that affect an agile process. How to become self-adapting is another area in which Cockburn offers ideas about how a group might adapt processes to their projects. A starter methodology is given and ways to adapt it are discussed. The Crystal family of methods is offered by Cockburn in the final chapter. These methods exemplify how he has solved problems in methodology design in the past.

Most chapters contain a concluding section called, "What Should I Do Tomorrow?" This section is used by Cockburn as a place to make suggestions to the reader about how they might notice the subjects covered in that chapter in their group, in the work place.

Agile Software Development is excellent and proved to be fun to read. Cockburn has done a wonderful job introducing the reader to adaptation of agile methods of software development.

"Selling" System Administration

by David Parter

President, SAGE STG Executive Committee

parter@sage.org

I am a professional system administrator.

From the founding of SAGE, building System Administration as a recognized profession has been an important part of our mission. At the time, many sysadmins did not get the respect they deserved – both as individuals and as professionals. Many stories were told of sysadmins with totally erroneous job titles, with the predictable difficulties when it came to hiring, training, and performance evaluations.

Since publication of the first edition, many SAGE members have said that the SAGE Jobs Descriptions booklet is the most important thing SAGE has done. In many organizations, SAGE members have used the Jobs Description booklet both to create appropriate job descriptions, and to educate management and the personnel department as to what sysadmins really do. Compared to other groups, those organizations have had two advantages: 1) they had already hired system administrators, and 2) their sysadmins were already members of SAGE.

During the recent boom – both in the dot-coms and in other computer-dependant companies – the market for system administrators was very good. It seemed to many that the need to educate (or agitate) for system administration as a profession had dissipated. Younger sysadmins that I met while doing SAGE outreach seemed surprised at the importance we placed on building credibility for system administration as a job title, and as a distinct job that should be done by someone with appropriate training, with appropriate management and expectations.

Yet I still find myself in the awkward situation of having to "sell" people on the idea that they need a sysadmin. Over the past few years I have had several situations that left me wondering about this. Why do I feel like an insurance salesman when I talk to managers about their system administration needs?

A few situations in particular convinced me that we have a long way to go, and need to give real thought to how to make the case for system administration in organizations that lack it, and need it.

A few years ago, I became involved, as a friend, in a modest, prudent software startup. This company was co-founded by a computer science professor. Most of the staff are either graduates or former staff of the university CS department, and all quickly learned (as they put it) "how good they had it" with a fairly large professional system administration staff to support their needs. Several told me of their surprise at how difficult it was to work using "stock" Unix/Linux systems — without all the software, configuration, and patches that they took for granted when professionals took care of their computers. The company was small, but growing. Engineering was slated to double (from 10 to 20 staff) in one year. Wisely, they hired a director of engineering to handle this. In fact, they hired my brother (yet another CS graduate, who had been working for several years for a rather large software firm with a reasonable sysadmin staff).

As far as I know, my brother respects me, and respects my work. He often asked me for advice about his new situation. Yet he didn't think that they needed a sysadmin, and I couldn't convince him. If I can't convince my brother, I thought, something is really wrong here!

Three engineers – hired and paid to produce the product that generated revenue for the company – "enjoyed" doing some sysadmin tasks. Since they are very bright, and had a good model to copy (the CS Department), they did fairly well at it. But the bottom line was that they were amateur sysadmins, and none of them had the time or inclination to focus on the "complete" system administration needs. And (of course) system administration always took a back seat to software development priorities.

The result was predictable – inconsistent methods, inconsistent software installations, a lack of planning, and a rapidly growing mess.

After a while, they decided that they needed a sysadmin. There were several factors that contributed to this decision: they realized that schedules were slipping because engineers were diverted from development to sysadmin crises, and some flaws in their current system administration practices became obvious.

I guess they learned the hard way. But is that the only way we can convince management that they need a professional sysadmin – by having them fail trying to do without? There must be a better way. If you have any ideas, please let me know.

SAGE Retreat

David Parter

President, SAGE STG
Executive Committee

parter@sage.org

The SAGE executive committee held a training and planning retreat on the weekend of February 22-24. The retreat was very valuable, and we are looking forward to a very productive and exciting year. The next regular meeting of the Executive Committee will be at the USENIX Annual Technical conference this June in Monterey. We will also host a SAGE BOF at the conference. As always, please send comments, suggestions and feedback to *sage-exec@sage.org*, or directly to me or to any member of the executive committee.

SAGE Certification Update

Stacy Gildenson

Managing Consultant

stacy@sage.org

cSAGE RUN: SAGE Certification Is Live

By the time you read this, the first level of SAGE certification, called cSAGE, will be available worldwide. This first level is geared toward junior-level system administrators and will require a candidate to agree to a basic statement of a candidate's applicable system administration experience prior to taking the exam. cSAGE testing will then consist of two exams, a core plus a module, which can be completed in the same sitting, or completed independently of one another within a given period of time. The core exam is neither platform nor vendor specific, while the modules are platform-specific exams.

SAGE Certification began testing junior-level candidates worldwide on March 26. The program has received press from *SysAdmin Magazine, IT Contractor, Certification Magazine, GoCertify.com*, and *IT World*. The list is growing daily. For more information on who's talking about us, see *http://www.sagecert.org/news*. Another source available for updated information on the program are opt-in mailing lists which are accessible from the Web site.

Tanya Alexander-DeBuvitz, the program's marketing consultant, and J.D. Welch, the Webmaster, worked in conjunction with the staff and boards of SAGE Cert, SAGE, and USENIX developing *http://www.sagecert.org* in record time. In addition to standard content development, the site contains discussion boards, surveys, and a unique customer service interface developed by Galton Technologies. The service, called CertManager, allows a candidate to review and update his or her contact information remotely through a dynamic Web-based interface. In the future, SAGE Certification plans to build or locate a customer service and reporting interface that better conforms to our commitment to universal accessibility and privacy.

Both the core and the UNIX module ran operational exam betas through VUE testing that was completed on February 17. The core beta was extended past LISA in order to accumulate more exam results from lower-level candidates. SAGE Certification appreciates the patience of those who had to wait a few weeks longer for their LISA score reports in order for us to more accurately analyze the test items.

Finally, SAGE Certification is seeking training and publishing experts to assist us in developing accurate content for our program. If you work within an organization that may be interested in committing to our knowledge management standards for training and publishing, please contact Stacy Gildenston at *stacy@sage.org*.

A New Career!

by Hal Miller

Major, US Air Force
Chief of Maintenance, 143 Combat Communications Squadron

halm@sage.org

So, here I sit, not knowing from minute to minute what comes next, nor how to prepare and plan my home life.

I was laid off the week after the September 11th catastrophe, the day after I'd had surgery. Not to worry, I was then "mobilized" (I'm an Air Force/Air National Guard reserve officer), and have been on active duty ever since. My job keeps changing, duty location keeps changing, and "they" keep talking about sending me (well, lots of us) to many parts of the world where SAGE doesn't yet have a presence.

I can't help thinking about how well my sysadmin experience has trained me for my current role. I'm continually asked to do things I've never seen nor heard of, or have no clue how to tackle. I'm asked to do things I know could be better dealt with in other ways. I'm asked to fix symptoms rather than analyze and fix/prevent problems. Everything is critical and must take precedence over everything else I'm doing. Sound familiar? Seems that the same skill set I use on a computing site also works here: people-handling skills to calm them down, show confidence in my own abilities, show competence and professionalism, then

distract them while I frantically flail about trying to figure out what's really wrong and what to do about it. All without enough tools, training or budget, of course.

My past military experiences were in days of another kind of war. Some people back then refused to call or treat that one as a "war" because it didn't look like the previous one. Seems like that scenario (differences) has played out again. As has always happened in history, nations have been preparing to fight "the last war", instead of "the next war", and we (as a nation, if not globe) are scrambling to get on track now.

I've noticed something novel about this new war. The key players aren't the Marines hitting the beach, or the pilots with scarves flying in the wind, or tank jockeys traversing the countryside, or the admirals and generals directing it all. This time the key participants are the police and firefighters at the station down the street, the medical research technicians at the lab, and yes, the sysadmins at every computing site in the world. Suddenly I'm on two lists as a combatant, one in which "we" understand combat, and the other in which "we" (different we) understand the current war. The trick is going to be to get these two groups to work together.

Frankly, I can think of few groups more diverse in nature than the military and the sysadmin community. This is going to be "interesting", to say the least. I've been grappling with this problem for a number of years already (hah! prescience, just like a sysadmin) without success. Still trying.

The military now has "aggressor" units, where a bunch of self-appointed experts sit and try to crack into other units, to show how non-secure our networks are. We know that already, and don't need further proof, but that seems to be as far as they've taken it. Of course, military use of computing tends to be pretty limited (I still maintain that a word processor function is not a computer, but a fancy typewriter.) There are exceptions, including, for example, the mainframes chunking out flight track data for every aircraft, radar control from those mainframes, communication satellite tracking to maintain connectivity, etc. In nearly all cases, the critical machines are not networked, and generally also physically secured. Those machines that might be doing something critical for the military forces (such as battlefield/theater command and control systems) are being dramatically underutilized because of the poor quality of software and systems engineering. Things we in the SAGE and USENIX community could easily make better, if we could just get all of us past our prejudice against each other . . .

Thoughts and suggestions more than welcome!

USENIX news

The Long Wave

by Daniel Geer

President, USENIX Board of Directors

geer@usenix.org

A high percentage of the lay press and people in general have some idea that computer horsepower keeps getting better, i.e., they've heard of Moore's law and its halving of price per unit performance every 18 months. Economists, such as the American Economic Association's Dale Jorgensen, commentators, such as the *American Spectator*'s George Gilder, and senior government officials, such as Senator Ron Wyden, each in their own way point to the interaction of tax law and Moore's law as *the* source of wealth expansion over the last decade. Who am I not to concur?

But Moore's law has another important aspect – it predicts the future. Predicting the future well is a great source of capital, monetary and intellectual, as well as a guide on how to spend your monetary and/or intellectual capital. Thinking of Moore's law as a "curve," I'd like to add two other curves to the mix and hazard a prediction or two. Both curves are, like Moore's, more approximations based on observation rather than laws of physics. All three are imperfect, but let's ignore that imperfection for the moment.

The first curve is the price of storage. For the purpose of this article, it shows a similar form to the curve for computing, i.e., a halving in price per unit volume every cycle where, in this case, the cycle is 12 months rather than 18. An approximation to be sure, but a well-studied approximation (see, e.g., Clayton Christensen's *The Innovator's Dilemma* for a book-length treatment).

The second curve is the price of bandwidth – raw communications bandwidth. Again for the purpose of this article, let's say it, too, shows a halving in price per unit volume every cycle – in this case, 9 months rather than 12 or 18. Another approximation to be sure, and one that in a full econometric model would have to be adjusted to take into account both regulatory and monopoly supply issues. For the moment, though, I am talking just about raw costs – system inputs, in other words.

Now even if the numbers 18, 12, and 9 are off, these power curves have a relationship to each other that is at least ordinal, i.e., computing horsepower increases pretty awesomely but for storage it is even more awesome while for bandwidth it is more awesome yet. So what might it mean if we had, say, another decade of increases in bang for the buck along each of these fundamental axes?

Taken over a decade, computing's cost effectiveness would increase by a factor of 100, storage by 1000, and networking would be 10,000 times more cost effective. What I am interested in, however, is not these raw numbers but the relative change in what a networked computer might be like 10 years from now as compared to today. Within a decade, the CPU would be facing 10 times as much data per available cycle and 100 times as much bandwidth; yet draining the entire storage of such a machine over the net would take only one-tenth the time, even though the overall data volume had gone up three orders of magnitude. Putting it differently, this is not the world with which we are familiar and for which our tools, or thought processes, are familiar.

The very existence of the "distributed computing" model is a reaction to the last big wave, the one where the impact of these sorts of curves moved the sweet spot in computing-for-competitive-advantage from the mainframe data center to the desktop. UNIX happened to be, by a combination of good luck and rare foresight, in exactly the right place at exactly the right time, and so UNIX was a central player in that "revolution." The pendulum is swinging again, and it is the three curves, or more precisely, the economic implication of the three curves that is powering the swing.

Last August I heard Jack Valenti, head of the Motion Picture Association of America, boast that he could re-ignite the economy with only two ingredients, two ingredients he was challenging his audience to provide: 20 million homes on broadband and a reasonably viable solution to digital piracy. With that he could open a fixed-subscription-price video-on-demand service and use up the glut of bandwidth now in the ground. He might be right about what those two things would give him and about the market opportunity; Excite reports that already 15+% of its total bandwidth consumption is due to Kazaa. Boastfulness aside, Valenti is looking at near-term viability of a model where it is the bandwidth that is cheaper than the storage, which is cheaper than the processing power; prices are falling with a rapidity that only a couple of years ago all but a few would have found laughable. This is the three curves in action, as interpreted by an entrepreneurial schemer of the first rank.

As a different example, peer-to-peer is just beginning, and it is these curves that will drive it. Speaking to the system administrators in my audience: what do you think it will mean to have little computationally smart network nodes popping up everywhere, talking to each other, thinking nothing of exchanging vast quantities of data that appear to have zero cost to everyone except the infrastructure's cost of reliability? Security people: what do you think a network with no perimeter and impossible synchronization means to the data integrity and confidentiality constraints that more or less are the constants in every problem statement you get to solve? Database administrators: IBM Research expects 85% of all storage at IBM to be on SANs within five years – do you see the same thing coming and, if so, what are you going to do about organization much less naming?

I don't think we can be content to get better and better at what we do. I think we have to work ourselves out of a job in the sense that the essentialness of people like us just is not scalable. You've probably seen the study that suggests, given current growth rates (those three curves) in volume and complexity of computing, that half the world's population will have to be in computer operations, broadly defined, within the next 10 years. Similar predictions several decades ago removed telephone operators from the scaling limits of the telephone system through the introduction of direct dial. Before that, 100,000 elevator operators were made redundant by the automatic elevator, and it's a good thing as we now have four orders of magnitude more elevators today than we did then. We have to work ourselves out of a job or face becoming, ourselves, a limit to productivity growth.

It's a sobering thought. If not us, who?

Fifteen Years Ago in ;login:

by Peter H. Salus
USENIX Historian
peter@matrix.net

Nowadays, OSes seem confined to the UNIX model (in which I subsume Linux) and the VMS model (which includes Windows). But once upon a time in a universe alien to this one . . .

The January/February 1987 issue of *;login:* carried, among other things, an article by Matt Bishop on "How to Write a Setuid Program" and an article on the "Sprite Project," by John Ousterhout, A. Cherenson, Fred Douglis, M. Nelson, and Brent Welch.

The March/April issue contained articles on MINIX, by Andy Tanenbaum, and DASH, by Dave Anderson and Domenico Ferrari.

For those of you who have been living in the DOS/Windows/NT universe, Sprite was a network-oriented operating system, MINIX was (is?) a UNIX clone (compatible with V7) that would run on the IBM PC. The internal structure was quite different from UNIX in that it "is a message-passing system on top of which are memory and file servers." DASH was a very large distributed system, potentially involving "thousands or millions of connected hosts . . . [which] may be thousands of miles apart."

I consider these truly remarkable landmarks. Without MINIX we would (most likely) not have Linux, and Beowulf, SETI, etc. owe their bases to Sprite and DASH.

But these weren't all. In January/February, Marc Donner reviewed *The C Programmer's Handbook* and in

March/April, Lou Katz reviewed the very first "nutshell handbooks" from O'Reilly.

And there was the Weirdnix competition, too.

Quirky Cows & Computing Challenges: The USA Computing Olympiad

by Gary and Steven Sivek

[*Identical twins Gary Sivek and Steven Sivek have been participating in the USA Computing Olympiad for three years. Both have qualified twice for the summer training camp; last year, as a member of the US team that competed at the International Olympiad in Informatics, Steven earned a bronze medal.*

In this article, Gary and Steven explain why this contest keeps them coming back and why other teen programmers should join them. Ed.]

Bessie the Cow wants to use a pogo stick to travel along a cow path of integer length L. Bessie starts at point 0 and proceeds in integer pogo-jumps to land exactly on point L. Bessie's velocity, V, starts out at zero and is always nonnegative. At the beginning of each bounce, she can change her velocity by -1, 0, or +1. The velocity is the distance Bessie travels along the path during the bounce. Bessie can be traveling at any nonnegative velocity when she lands on point L. Bessie wishes to avoid jumping on any of the cow pies that happen to be located at various integer points along the path (not including location 1 or location L, of course). Your job is to determine the smallest number of bounces to reach exactly the end of the path, point L.

Thus begins a problem entitled "Cows on Pogo Sticks" from the USA Computing Olympiad's 2000 Fall Open. Accompanied by three other programming tasks, this formed part of a five-hour contest designed to challenge the best high school programmers from around the world. This was typical of the contests conducted since USACO's inception in 1993: problems differing in difficulty and solution methods, a wide range of scores among participants, and, of course, bizarre bovine humor.

These USACO contests simply wouldn't be the same without cows. These animals, abundant in the olympiad's home state of Wisconsin, the site of its annual training camp, are featured in nearly every problem as they engage anthropomorphically in a number of activities: attending "kinergarten," planning an L-shaped pool for their forest, and jogging around their pasture, among other things. Sometimes the benevolent Farmer John plays a pivotal role in the problems: trying to give cows gifts according to their wish lists, dividing the herd into a set of fields, or just calling the cows home for dinner. If you understood the "kinergarten" pun – kine is an archaic plural of cow – you'll love the bovine touch that helps USACO stand out among the other science olympiads. These contests have another purpose beyond simple Holstein hijinks and Guernsey glorification, of course: each year, they ultimately determine the top

l. to r.: Steven Sivek, Gary Sivek

15 programmers (using C, C++, or Pascal) in the United States, who win a trip to the University of Wisconsin-Parkside as finalists. There, after a week of intense

training and contests, the four-person team is selected that will represent the US in the annual International Olympiad in Informatics (IOI). The US team has historically excelled in competition. Last year's team of Reid Barton, 18; Tom Widland, 18; Vladimir Novakovski, 16; and Steven Sivek, 17, won the top gold medal, two silver medals, and a bronze medal, respectively.

The 2001 training camp was a nonstop flurry of activity for the finalists, selected after an unusually grueling US Open in which only one competitor received even half of the points possible. After arrival and introductions on the first day, every day began with either one of four three-hour experimental contests or one of two exhausting five-hour "Challenge Rounds" used to determine the team of four. Afternoons included discussions of solutions from the morning's contest, planning for the next day's contest strategy, lectures in different techniques, and, of course, time for relaxation activities such as ultimate Frisbee, disc golf, an annual business simulation game, and various excursions to movies, museums, and water parks. Nights included presentations by the coaches on hot research topics in computer science such as operating systems, network security, and mapping the Internet, which all showed interesting applications of computer science outside of these contests.

The 2001-2002 season has had a strong start, drawing almost 500 competitors in recent contests. The contests are scored by a unique grading algorithm that spreads scores over the full range of 0 to 1,000 possible points by carefully weighting programs and individual test cases according to relative difficulty; thus, a score of "only" 500 would actually indicate a strong performance, and a perfect 1,000 is exceptionally difficult to obtain.

Two divisions allow for a wider range of skill levels, with an orange division for those just beginning and a green division for more experienced students. Only green division participants are considered for the training camp, but students can switch to green mid-year and even earn an invitation from a strong performance in the U.S. Open alone! The U.S. Open is the final contest of the year, after the Fall, Winter, February, and Spring contests, and is administered by proctors in schools over a period of five hours.

Of course, it's easier to be interested in these contests than it is to excel in them. If you were confused by the above problem, you're not alone. But there is something you can do about that! Ever since the Winter 2000 contest, the coaches and some top-performing students have worked to provide an interactive training site online (*http://ace.delos.com/usacogate*) where you can try your hand at these problems and learn techniques to solve them. The page provides resources on such topics as the ever-popular dynamic programming (affectionately called "DP"), shortest path algorithms, greedy algorithms, network flow, effective search techniques, minimal spanning trees, computational geometry, and any other problem type that contestants might encounter, including the mysterious "ad hoc" category. It also makes students work completely through even the toughest challenges, not allowing anyone to move forward without completing preceding problems, but coaches are available for hints when needed. The solution to "Cows on Pogo Sticks" . . . well, we won't spoil the fun for you. Learn about DP in the training pages and solve it for yourself!

The USA Computing Olympiad would not be possible without the hard work of many people, including director Don Piele; head coach Rob Kolstad; coach Greg Galperin; coaches/former participants Hal Burch, Russ Cox, and Brian Dean; and USACO's generous sponsor, USENIX, which provides funding for the Olympiad each year. Beyond them,

though, USACO would not exist without its participants. So visit the USACO Web site for more information, and start competing!

USACO Contests

The USACO offers multiple Internet contests in which individual precollege students have three to five hours to solve three to five problems. Students are encouraged to, but do not have to, participate in all of these contests before entering the US Open. The US Open will be given on April 11, 2002, at local high schools.

Based on their performance in any contests they complete as well as their work on training materials, 15 students are selected for the USACO training camp. Four students from the training camp will form the US team that will travel to the International Olympiad in Informatics in August, to be held this year in Korea.

Visit *http://www.usaco.org* for more information, including past problems and instructions for joining the mailing list.

Research Exchange Program (ReX) Update from the Field

by Sabine Buchholz

buchholz@kub.nl

A report on the ReX exchange program between Tilburg University, the Netherlands, and the Natural Language and Information Processing (NLIP) Group at the Computer Laboratory, University of Cambridge, UK.

Since not all readers of *;login:* might be familiar with the research field of computational linguistics, which forms the scientific background of this report about my four-month exchange stay at Cambridge University, I will start by introducing some of the most important concepts. Computational linguistics studies the combination of computers and natural (i.e., human) languages. It aims at developing and implementing models of how natural languages can be processed. Applications include text-to-speech, machine translation, question answering, and natural-language interfaces. A common subtask in many applications is parsing: determining the syntactic structure of a sentence. Although to a certain extent parsing can be done on the basis of knowledge about the based on part-of-speech (like verb, noun, preposition) of words, it is widely acknowledged that information about specific words (lexical knowledge) is advantageous. One of the most important pieces of lexical information is subcategorization(subcat), especially of verbs. This tells us, for example, that a verb like "give" preferably takes two complements (the di-transitive frame): "to give somebody something"; whereas "invent" takes one complement (transitive): "to invent something"; and "sleep" takes none (intransitive): "to sleep" but not "to sleep something." This information helps the parser in disambiguating sentences that would otherwise be ambiguous, like "She gave/invented Tim water." As parsers should be applicable to all kinds of texts, from all domains (for example for applications on the Internet), and extensive subcategorization information is not readily available for all verbs, it can best be acquired automatically. It is this subcat acquisition problem that I worked on during my time at Cambridge.

Ted Briscoe is a reader in computational linguistics in the Natural Language and Information Processing (NLIP) Group at Cambridge University. He had previously developed an automatic subcat acquisition system that works by parsing large amounts of texts (parsing based on part-of-speech information only), recording the frequency with which each frame occurs with each verb, and filtering out combinations that did not occur sufficiently frequently (and thus probably result from parser errors). Those verb-frame combinations that pass the filter, together with their associated frequencies (converted to probabilities), can subsequently be used for better probabilistic parsing.

To improve the performance of this last filtering step, PhD student Anna Korhonen developed a method for smoothing the acquired frequency distributions of new verbs by backing-off to semantically related known verbs, and for filtering based on the maximum likelihood estimation (MLE) of the resulting frequencies. As her method presupposes knowledge about semantic classes of verbs that is not easily available for all verbs, my task for the project was to explore alternative filtering approaches using machine learning.

I was a fourth-year PhD student at Tilburg University, the Netherlands. Since pPart of my research concerns concerned finding grammatical relations between verbs and their complements, which is related to parsing and subcat acquisition, . I thus already knew several of Ted's and Anna's publications on the subject when Ted asked my supervisor Walter Daelemans whether one of Walter's students would be interested in the project. Walter had developed a machine-learning algorithm called Memory-Based Learning (based on the k-Nearest Neighbor algorithm) which I had also used for my thesis research, so I had the necessary background for the project and also liked the idea of spending some time in a foreign country. On the one hand, Cambridge with its beau-

tiful and famous university, dating back to the 13th century, is very different from the "modern industrial city" Tilburg with its 75-year-old university. On the other hand, everybody cycles there tooin Cambridge, and the landscape is conveniently flat and the city small, so I immediately felt at home. I arrived in August, which is a good time for getting to know the city, the river, and the surroundings but a bad time for arriving at a university since half the staff is on holiday or attending conferences or summer schools. My first weeks were complicated by the fact that the entire computer laboratory, of which the NLIP Group is a part, was moving to a new building at the western edge of the city (an event which should have happened long before I arrived but which had been postponed several times). So I started by (re)reading the available literature, most notably Anna's nearly finished thesis, and by discussing a lot with Ted and Anna. Once I got my own office and computer in the new building, I started to locate all of the corpus resources, acquisition system modules, and evaluation software I had been reading about, and to use them myself. I also had a look at the source code, reviving my knowledge of Lisp, C, and shell scripting on the way. I then worked on three the following topics:

After the subcat acquisition system has parsed a text, tokens of frames together with verbs can be extracted. These must then be classified into types of frames. For example, "He invented the telephone" and "The telephone was invented" instantiated the same kind of (transitive) frame. I adapted the classifier to accomplish this passive-to-active conversion, so that all the tokens would contribute to the frequency count of their mutual type.

To evaluate how well the subcat acquisition performs, a so-called gold standard had been created manually. This means that some verbs were chosen at random, people looked at a representative number (mostly 300) of sentences in which these verbs occur, and noted how often they occur with which frames. Performance of the system is then computed in terms of precision (how many of the verb-frame pairs that the system proposes are also in the gold standard), recall (how many of the pairs in the gold standard are found by the system), and rank correlation (how similar is the order of pairs if gold standard and system results are ordered by frequency).

However, there are more types of frames that should be distinguished on theoretical grounds than the subcat acquisition system is able to do on the basis of part-of-speech information alone. Therefore the output of the system frequently was not a list of frames for each verb but a list of frame disjunctions.

These disjunctions complicated the computation of precision, recall, and rank correlation. In addition, they made the results of evaluation of the machine-learning experiments hard to judge, since the learner tended to predict all possible disjunctions containing common frames. I therefore developed a variant of the classifier that in which it wasis forced to return a list of single frames (no disjunctions). It is an open question what would be the best way to make such a forced decision. At the moment, the most general or frequent frame of a disjunction is chosen.

I used a supervised machine-learning algorithm. This means that one part of the gold-standard material was used for training the algorithm and another part for testing it. For each verb-frame pair acquired by the subcat acquisition system, the learner has to make a binary decision: keep it or reject it. As a first step, I had to create a machine-learning instance for each such pair. Features of the instances correspond to pieces of information from system output or from external information sources (like the semantic classes used by Anna). I could then study the influence of various (combinations of) features on filter performance. After machine-learning filtering, instances need to be converted back to the initial format of lexical entries. Results are that the influence of features depends heavily on the sort of verbs tested. In general, a combination of type of frame and observed frequency performs well, and adding additional information about semantic classes helps a little. For a special group of verbs, however, the type of frame feature alone is sufficient and adding frequency information degrades performance. An experiment that still needs to be performed is to combine Anna's back-off smoothing with the machine-learning filtering.

I devoted much of the last part of my exchange into documenting my software so that research into the method can be continued after the official end of the exchange project. The documentation will form part of a larger technical report that should describe the subcat acquisition system and related modules. I will also use my new knowledge to make comparisons between the subcat acquisition system and parts of my thesis work.

I had a very pleasant and informative stay and wish to thank all the people who made my exchange visit possible.

For more information about this exchange, please contact:

Dr. Sabine Buchholz
S.Buchholz@kub.nl
Dr. Ted Briscoe
Ted.Briscoe@cl.cam.ac.uk

USENIX and Stichting NLnet jointly support the ReX program. For more information about ReX, see *http://www.usenix.org/about/rex.html*.

DON'T MISS OUR FREENIX SESSIONS DEDICATED TO *EVERYTHING OPEN SOURCE*

2002 USENIX Annual Technical conference

June 10-15, 2002

Monterey Conference Center

MONTEREY, CA USA

TECHNICAL TUTORIALS
June 10 - June 12, 2002

TECHNICAL SESSIONS
Thursday, June 13 - Saturday, June 15, 2002

FREENIX SESSIONS
Thursday, June 13 - Saturday, June 15, 2002

FREE VENDOR EXHIBITION
June 13 - 15, 2002

Get practical, in-depth, technical training, explore the latest, ground-breaking research, develop new ideas and solutions, & connect with your colleagues at the USENIX Annual Technical Conference.

KEYNOTE

LAWRENCE LESSIG, *Professor of Law at the Stanford Law School and founder of the school's Center for Internet and Society*

"The Internet's Coming Silent Spring"

ERIC ALLMAN,
Sendmail, Inc.
"Taking an Open Source Project to Market"

STEVE BELLOVIN,
AT&T Labs - Research
"Internet Standards"

BRUCE SCHNEIER,
Counterpane Internet Security
"Fixing Network Security by Hacking the Business Climate"

FEATURED TUTORIALS:

- **FreeBSD Kernel Internals**
- **Inside the Linux Kernel**
- **Advanced Solaris Sysadmin**
- **Topics in Unix & Linux Administration**
- **System & Network Performance Tuning**
- **Building Honey Pots**
- **Cisco's Security Features**
- **Introduction to Computer Security**
- **Practical Wireless IP**

REGISTER ONLINE BY MAY 17 AND SAVE UP TO $400!

www.usenix.org/events/usenix02

USENIX

CONNECT WITH USENIX & SAGE

MEMBERSHIP, PUBLICATIONS, AND CONFERENCES

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710
Phone: +1 510 528 8649
FAX: +1 510 548 5738
Email: <office@usenix.org>
<login@usenix.org>
<conference@usenix.org>

WEB SITES

<http://www.usenix.org>

<http://www.sage.org>

EMAIL

<login@usenix.org>

COMMENTS? SUGGESTIONS?

Send email to <ah@usenix.org>

CONTRIBUTIONS SOLICITED

You are encouraged to contribute articles, book reviews, photographs, cartoons, and announcements to *;login:*. Send them via email to <*login@usenix.org*> or through the postal system to the Association office.

The Association reserves the right to edit submitted material. Any reproduction of this magazine in part or in its entirety requires the permission of the Association and the author(s).

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

;login:

PERIODICALS POSTAGE
PAID
AT BERKELEY, CALIFORNIA
AND ADDITIONAL OFFICES

USENIX Association
2560 Ninth Street, Suite 215
Berkeley, CA 94710

POSTMASTER
Send address changes to *;login:*
2560 Ninth Street, Suite 215
Berkeley, CA 94710