;login:

THE MAGAZINE OF USENIX & SAGE

August 2000 • volume 25 • number 5

inside:

USENIX & SAGE

The Advanced Computing Systems Association &
The System Administrators Guild

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>

Needs

In the late 1960s, when the psychological world embraced behaviorism and psychoanalysis as its twin grails, Abraham Maslow proposed a hierarchy of needs. This hierarchy was brought to mind because I am hosting the Polish computing champion for a short visit, and he often asks questions of the sort, "Why would anyone need such a large car?"

Maslow listed, in order:

- Physiological needs, including air, food, water, warmth, shelter, etc. Lack of these things can cause death.
- Safety needs, for coping with emergencies, chaos (e.g., rioting), and other periods of disorganization.
- Needs for giving and receiving love, affection, and belonging, as well as the ability to escape loneliness/alienation.
- Esteem needs, centering on a stable, high level of self-respect and respect from others, in order to gain satisfaction and self-confidence. Lack of esteem causes feelings of inferiority, weakness, helplessness, and worthlessness.

"Self-actualization" needs were the big gun of the thesis. They exemplified the behavior of non-selfish adults and are, regrettably, beyond the scope of this short article.

As I think about the context of "needs" vs. "wants" or "desires" in our culture, economy, and particularly among the group of readers of this publication, it seems that we're doing quite well for the easy needs (physiological needs and safety). I know many of my acquaintances (and myself!) are doing just super in their quest for better gadgetry, toys, and "stuff" ("whoever dies with the most toys wins").

As I try to answer my visitor's sincere questions, I find myself trying to differentiate between "needs" and "neat stuff that's fun to have." I'm delighted that I am in a position to focus purchasing power and time on "neat stuff" instead of my next meal.

But he's not the only one to say, "Why would anyone need three refrigerators?" (The answer is: one for the kitchen/food, one for drinks and entertaining, one for film and paper in the darkroom.) I think the answer to the question lies more in the question than in the answer. I think that in our well-compensated part of the economy, we often don't focus on needs like food and safety (and I'm sure some of us don't focus on the needs for love, esteem, and self-actualization) when it is so simple to trade money for time and convenience. I like reclaiming time (the true nonrenewable resource from each person's point of view) and making my life easier (e.g., a self-propelled lawn mower, an attachment to the TV that enables me to press pause to take a phone call and then un-pause to resume a real-time program where it left off).

This causes me to reflect. Gadgets and toys are fun. What about love, esteem, and the elusive self-actualization? These are harder (and dramatically more difficult to purchase in any sincere way)!

I haven't looked at this list for years. Now that I've performed the research for this article, I'm going to think about it a bit to see how my life is doing for those other needs. I think it's OK but I want to make sure that I don't fail to seize the day.

apropos

by Tina Darmohray
<*tmd@usenix.org*>

Tina Darmohray, co-editor of *;login:*, is a consultant on Internet firewalls and network connections and fre-quently gives tutorials on those subjects. She was a founding member of SAGE.

A Negative in the Zero Commute

Judging from the subject lines of my SPAM-mail, the concept of "firing the boss" and "working from home" are hot topics for a lot of folks. Apparently the concept of working from home conjures up images of an ideal work environment with all sorts of side-benefits built into the scenario. I've been working exclusively from a home office for the last six years, and there are certainly some benefits to it (personally, I get a lot of laundry done in the time I would have been distracted at the water cooler), but it's not without some drawbacks.

I was talking last week to a fellow consultant who also works at home. We've been good friends for a long time, and very few topics are off-limits for us. The topic for this particular day was home offices and families and how they sometimes just don't mix. We were in the mood to vent, and we did. Later I thought that our conversation might be of interest to others who may be considering working from home, or are envious of those who do. In either case, this should help put some perspective on the realities of work-from-home scenarios.

Our observation is that a home office doesn't seem to command the respect that an office located in the corporate compound does. Our beef is that our family members sometimes don't recognize that it should. We went through the list of atrocities that happen in home offices that just don't happen "at work":

- a continual migration of office supplies away from their expected location in-my-office-where-I-need-them: ruler, tape, stapler, mechanical pencils, scissors
- someone else's leftovers left between the edge of the keyboard and the mouse pad (a particularly unwelcoming start to the day)
- the Web page changed from where you last left it
- specially sorted stacks of paper are re-sorted, covered up, or (maybe the worst), knocked off the desk into the trash
- people walking into the room talking loudly when you're on a business call, e.g., "MOM? Where are my baseball socks?" (My fellow consultant has started locking the door when he's on a call.)
- other family members accidentally answering the business line, or, more likely, using it because the family phone is in use at the time
- other computer-capable family members sysadmining your machines, e.g., default printer, default router, etc.
- competition for keyboard time, e.g., Junior wants to play computer games.

Don't get me wrong; there are many up-sides to working from a home office; I just haven't touched on those here. But our collective experience suggests there are a few innate problems with a zero commute as well.

the tclsh spot

by Clif Flynt
<*clif@cflynt.com*>

Clif Flynt has been a professional programmer for almost twenty years, and a Tcl advocate for the past four. He consults on Tcl/Tk and Internet applications.

The previous Tclsh Spot articles discussed building a stock quotegathering robot, saving the data and using the BLT graph widget to display a stock's history. This article will describe more details about the BLT graph widget and discuss using Tcl's associative arrays.

The first step in any sort of data analysis is getting the data. In this case, that means reading the stock data from the file created by the stock quote—retrieval robot.

The stock-quote robot produces a datafile with lines that resemble this:

```
955127760 SUNW {95 3/8} {2 11/16} 2.9 13:00 {93 7/8} 96 {93 1/32} 6,938
955127780 INTC {134 13/16} 5 3.9 13:00 {131 3/8} {136 17/32} {131 5/16} 15,590
```

These are deliberately formatted (by the robot) to be Tcl lists, to make reading them a bit simpler. But the data isn't quite as ready-to-use as it might be.

The problems to address are:

- The prices are given as fractions instead of decimal numbers.
- The trade volumes are written with commas.
- The data for many companies is mixed into one file.

The first two problems are data-representation issues. We can deal with them with a single data-conversion procedure to convert fractions to decimals and strip out commas.

For a task this simple — only two processing options, and a simple test to figure out how to process the data — I prefer to put both the selection and the conversion intelligence in the subroutine, instead of writing two conversion procedures and putting the selection intelligence in the loop. My reasons are:

- Loops tend to get long, and anything that simplifies them is good.
- If the selection logic becomes complex as I better comprehend the problem, I can rewrite the logic in a procedure more easily than I can rewrite the spaghetti code inside a loop.
- Conversion code is likely to be useful for another project, and it's easier to grab a procedure than extract code from a loop.

This procedure converts a fraction like 2 1/2 to a decimal, using the regexp command to convert the number to an arithmetic expression (2 + 1/2.0) and then evaluating the arithmetic expression to get the floating-point number 2.5.

Converting 1/2 to 1/2.0 looks strange, but there is a reason for it. By default, the Tcl expr command leaves numbers in their native state when it does math operations. Thus, it won't convert from integer to float before doing a division and will return an integer result (1/2 == 0).

However, if one operand in an expression is a float, other operands are promoted before the operation is performed. Thus, the operation 1 / 2 returns the integer 0, while 1.0 / 2 returns the floating-point value 0.5.

We used the `regsub` command in the robot as part of the page-parsing logic. We can also use the `regsub` command to strip unwanted commas from the volume data.

Here's a procedure that will convert a number in one of the two unwanted forms (fraction, or including a comma) into a decimal:

```
proc normalize {val} {
  # Only do fraction to decimal conversion if a fraction exists.

  if {[regexp {([0-9]* +)*([0-9]*)/([0-9]*)} $val m whole num denom]} {
    set val [expr $whole + ($num / $denom.0)]
  }
  # Delete any commas that might be in the value
  regsub -all "," $val " " val

  return $val
}
```

That leaves us with the third problem, separating the data for multiple companies into the appropriate lists.

The Tcl interpreter supports three data structures:

- simple variables like numbers or strings
- lists
- associative arrays

The associative array was first introduced to Tcl in the TclX extension and was quickly merged into the Tcl core. An associative array is an array structure in which the indices are strings instead of numbers. This concept is perfectly obvious if you're familiar with Awk or Perl, or perfectly bizarre if you've always worked with languages like FORTRAN or C.

Associative arrays look and act just like normal Tcl variables, except that the array name is followed by a pair of parentheses that enclose the index value. For example, `os(unix)` represents the index `unix` in the associative array `os`.

The associative array is the most powerful data structure in Tcl. Using some do-it-yourself naming conventions, you can emulate most of the complex data structures that other languages support.

For instance, using an associative array lets us write code like:

```
set fruitPrice(apple) 0.5
set fruitPrice(banana) 0.25
```

instead of the C equivalent:

```
struct fruit {
  char *name;
  float cost;
} fruitPrice[2];

  fruitPrice[0].name = "apple";
  fruitPrice[0].cost = 0.5;
```

```
  fruitPrice[1].name = "banana";
  fruitPrice[1].cost = 0.25;
```

If we want to track more data about our fruits, we can define a naming convention to use. We might decide that the indices will be the fruit name followed by a data description. For example:

```
set fruitInfo(apple.price) 0.5
set fruitInfo(apple.inventory) 500
set fruitInfo(apple.color) red
set fruitname "banana"
set fruitInfo($fruitname.price) 0.25
set fruitInfo($fruitname.inventory) 1000
set fruitInfo($fruitname.color) yellow
```

When it comes to graphing the stock data, we want a list of timestamps and a list of prices for a given company. So, to make life (or at least this example) simple, we declare that these lists will be saved in an associative array with an index naming convention of `StockSymbol.Description.` For example,`Data(SUNW.price)` will have a list of selling prices, and `Data(SUNW.date)` will have a list of timestamps for Sun's stock.

The data in the file is a set of lines, and each line is a list consisting of TimeStamp, Stock Symbol, Selling Price, Absolute Change, Percent Change, Time of Quotes, Opening Price, High Price, Low Price, and Volume.

The second item in the list is always the stock symbol. We can extract that value from the list with the lindex command. (The first item in a list is at position 0.) Once we know the symbol name, we can parse the rest of the data in the line using a foreach loop.

One of the features of foreach is that it can iterate through multiple lists simultaneously, retrieving the first item from each list, then the second item, and so on. The syntax for this is:

```
foreach variable1 list1 variable2 list2 ... variableN listN {...}
```

This code will read the lines from the file and extract the stock symbol from each line. It then iterates through the list of descriptive names and a list of values, and appends the current value to the appropriate list within the associative array:

```
proc readData {infl} {
  global Data

  while {[set len [gets $infl line]] >= 0} {
    set id [lindex $line 1]
    foreach n {date id price change pct dt o high low vol} v $line {
      lappend Data($id.$n) [normalize $v]
    }
  }
}
```

After this procedure has run, the associative array Data will have lists of all the data we've collected, indexed by stock symbol and type of data.

Now, we can get back to making graphs.

Just on general principles, I don't want to create our graph in mainline code and pack it on the main window. Putting this code into a procedure, with the parent frame as an argument, makes it easier to merge this graph into a larger application when we have multiple frames to deal with.

This procedure will create an empty graph with just a label and will return the name of the new graph widget:

```
proc makeGraph {parent name} {
  global Data
  :blt::graph $parent.g_$name -title "Stock Data for $name" -width 600 -
height 400

  return $parent.g_$name
}
```

The graph widget displays lines as graph elements. A graph element is an object that contains a list of X, Y values and some options to define how the line should be drawn.

Syntax: *widgetName* `element create`

?option value?

We can draw a line showing price versus time by adding this code to the makeGraph procedure after the graph command:

```
set name "SUNW"
$parent.g_$name element create "$name Price" -xdata $Data($name.date) \
  -ydata $Data($name.price) -symbol none
```

This procedure will create a graph like this, with a label and legend, and with the X axis tics listed as seconds since the epoch.

This is better than nothing, but not much. However, the BLT package has lots of facilities for customizing graphs. For instance, the first thing I want to change on this graph is the tic labels. The BLT widget command axis can be used to configure a graph's X and Y axes.

Syntax: *widgetName* `axis configure` *axisName -option1 value1 ...*

widgetName The name of the graph object that contains this axis
`axis configure` Identifies this command as configuring the axis
axisName Identifies which axis is being configured. The default axes are:
- X The bottom X axis
- X2 The top X axis
- Y The left-hand Y axis
- Y2 The right-hand Y axis

-option value An option name and the new value to associate with that option.

Options include:

- -fmt The name of a function to use to format the tic labels
- -max The maximum value to display
- -min The minimum value to display
- -hide 1 to hide an axis, 0 to display. By default, the X and Y axes are displayed, and the X2 and Y2 are hidden.

So, to show the X axis tics as Month/Day, we can use the `Tcl clock` command to convert the seconds since the epoch into a MM/DD format with a procedure like this:

```
proc fmt {graph sec} {
  return [clock format $sec -format {%m/%d}]
}
```

and tell the graph to use this procedure to format the tics by adding these lines to the `makeGraph` procedure:

```
  # Format the x-axis tick labels as Month/Day
  $parent.g_$name axis configure x -command fmt
```

This is better, but we've still collected a lot of data we aren't viewing. For example, it might be good to know the trade volume.

We could make another line on the graph to show the volumes, but lines imply that the there is continuity between values, and there is no connection between yesterday's and today's trade volume.

A bar chart is a more appropriate way to display this info. We can generate barcharts on our BLT graphs. In fact, there is a whole other widget (`barchart`) designed for barcharts, but for this application it makes more sense to put the bars on the price graph.

The command to build a bar is:

Syntax: *widgetName* `bar create` *label ?-option value?*

`bar create`	Create a new set of bars on the graph
label	A label that describes this barchart (this string will be displayed in the legend)
-option value	Option and value pairs to describe this barchart. Options include:
-xdata	A list of values for the X axis
-ydata	A list of values for the Y axis
-mapy	The axis to map this data against (defaults to the lefthand Y axis)
-mapx	The axis to map this data against (defaults to the bottom X axis)
-fg	The foreground color for the bars
-bg	The background color for the bars
-barwidth	How wide to draw the bars (defaults to a single pixel)

One of the tricks in putting two sets of data on the same graph is how to scale the data. Since a high price for a stock is around 100, while a low volume is around 10,000 shares, we really can't plot both of these against the same Y axis.

This is where the Y2 axis and the -mapy option come in. We can map the prices on a 0—100 scale on the left axis, and volume on a 0—100,000 scale on the right axis.

In fact, we don't have to declare the sizes of the axes. The graph widget will automatically scale the axes from the minimum to maximum value in the data set.

So, to generate a volume barchart on the graph with our price graph, we add this line to the `makeGraph` command:

```
$parent.g_$name bar create "$name Volume" -xdata $Data($name.date) \
  -ydata $Data($name.vol) -mapy y2 -fg green -bg green \
  -barwidth 2000
```

Since the dates on the X axis are the same for the price and volume graphs, we don't need to use separate axes for that data. The `-barwidth 2000` makes the bars a little wider than a normal single-pixel line (2000 seconds wide, if you are counting units).

What this doesn't do is display the values. By default, the X2 and Y2 axes are hidden. But adding this line will display the Y2 axis:

```
$parent.g_$name axis configure y2 -hide 0
```

That leaves the high and low data. It would be nice to see the range of values in a day, and whether our stock closed at the top of the range or bottom.

Another feature that BLT supports is markers. Markers are things that you can put at location on a graph. They can be text messages (like "This is when the SEC cancelled trading"), or bitmaps (like a happyface when the stock splits), or polygons, or lines.

In this example, we'll use line markers to show the high and low prices for a stock, similar to the error lines in the graphs from your old physics lab.

The syntax for creating markers is:

Syntax: *widgetName* `marker create` *type ?-option value?*

widgetName	The name of the graph widget
`marker`	Create a marker.
type	The type of marker being created (valid types include text, line, bitmap, image, polygon, and window)
option value	An option/value pair. The options available vary depending on what type of marker is being created. Options for line markers include:
-outline	The color for the line
-coords	A list of coordinates to define the line

To create these markers, we need to tell the graph widget to draw a vertical line from the low price to the high price at each vertex. We can do this with the`create marker` command, and another loop with multiple lists and variables.

But while we're doing that, we might as well track the maximum and minimum prices in the high and low dataset. Since the graph was scaled to the maximum and minimum closing prices, there may be highs and lows outside that range.

Like most of Tcl, the BLT graph widget is introspective. You can query it to find out such things as what current configuration values are using the same configure and cget subcommands that are supported by native Tk widgets.

Our code can query the Y axis to find out what the max and min values in the price data are before we start looking at the high and low data.

The problem is that the graph widget hasn't looked at the data yet. When we invoked the graph and create element commands, the interpreter didn't actually create a graph. It placed events on the event queue to create the widgets as soon as the interpreter isn't busy doing something else (like evaluating our procedure). The event queue won't be checked until after our process finishes the `makeGraph`procedure.

The solution to this problem is to use the `update` command. The `update`command will cause the event queue to be processed before returning. The`update` command comes in two flavors, `update`, which will process all events, and `update idle`, which will only process events that are in the idle loop. Updating graphics objects is an idle-loop task, so that's the flavor we should use for this application.

This code will force the idle loop to be processed, find the starting maximum and minimum prices, draw high/low lines at each price, and then reconfigure the axis to show the new range:

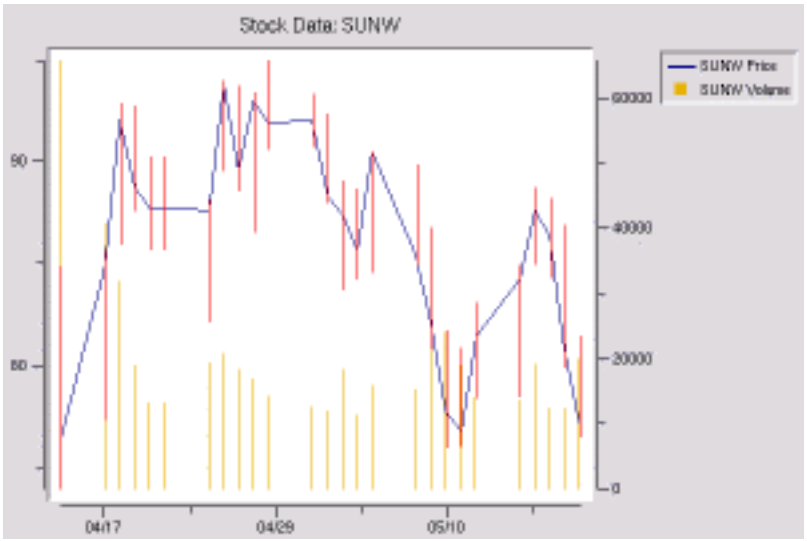
```
# Create vertical high/low lines at the vertices, and find max & min.
foreach d $Data($name.date) h $Data($name.high) l $Data($name.low) {

  $parent.g_$name marker create line -coords [list $d $h $d $l] \
    -outline blue

  if {$l < $min} {set min $l}
if {$h > $max} {set max $h}
}

# Now expand the Y axis to the real min/max range.
$parent.g_$name axis configure y -max $max
$parent.g_$name axis configure y -min $min
```

The code we've discussed in this article will generate a display that looks like this from a command like `wish stockShow.tcl SUNW`:



This code, and code from other Tclsh Spot articles, is available on my new Web site <*http://www.noucorp.com*>.

This is plenty of info, but running a new command-line task for each stock is too much finger work. The next article will discuss ways to display multiple stocks in this application.

```
#!/usr/local/bin/wish

package require BLT

set Graph(dataName) stock.data
set name $argv

#############################################################
# proc normalize {val} - -
#  Convert fractions to decimals and remove any commas
# Arguments
#  val    A numeric value
#
# Results
```

```
#  Returns a legal floating point or integer value.
#
proc normalize {val} {

  # Only do fraction to decimal conversion if a fraction exists.
  if {[regexp {([0-9]* +)*([0-9]*)/([0-9]*)} $val m whole num denom]} {
    set val [expr $whole + ($num / $denom.0)]
 }

  # Delete any commas that might be in the value
  regsub -all "," $val " " val

  return $val
}

#################################################################

# proc readData {infl} - -
#  Reads data from a stock data file
# Arguments
#  infl A channel to the data file
#
# Results
#  Creates lists of values in the Data global associative array,
#  sorted by StockSymbol.DataType
#
proc readData {infl} {
  global Data

  while {[set len [gets $infl line]] >= 0} {
    set id [lindex $line 1]
    foreach n {date id price change pct dt o high low vol} v $line {
      lappend Data($id.$n) [normalize $v]
    }
  }
}
#################################################################
# proc fmt {graph sec} - -
#  Formats a time-since-epoch time into MM/DD
# Arguments
#  graph The name of the graph which includes this tic mark.
#  sec Seconds since the Epoch.
# Results
#  Returns the appropriate MM/DD value.
#
proc fmt {graph sec} {
  return [clock format $sec -format {%m/%d}]
}
#################################################################
# proc makeGraph {parent name} - -
#  Makes a stock graph.
# Arguments
#  parent  A parent frame to hold this graph
#  name;    The stock symbol to use as a key to access the data
#          to display in this graph.
# Results
#  Creates a new graph widget, and returns the name of that
```

```
#  widget.
#
proc makeGraph {parent name} {
  global Data

  if {![winfo exists $parent.g_$name]} {
      # Create the graph
      ::blt::graph $parent.g_$name -title "Stock Data: $name"
        -width 600 -height 400

      # Format the x-axis tick labels as Month/Day
      $parent.g_$name axis configure x -command fmt

      # Create a line showing the stock price when
      # the robot ran.
      $parent.g_$name element create "$name Price" -xdata
        $Data($name.date) \
        -ydata $Data($name.price) -symbol none

      # Generate a bar chart for volume, and display the second
      #   Y axis that the bar chart references
      $parent.g_$name bar create "$name Volume" -xdata
        $Data($name.date) \
        -ydata $Data($name.vol) -mapy y2 -fg green -bg green \
        -barwidth 2000

      $parent.g_$name axis configure y2 -hide 0

      # Do an update to force the graph to run through the data
      # and calculate the min and max values.

      update idle;
      set max [$parent.g_$name axis cget y -max]
      set min [$parent.g_$name axis cget y -min]

      # Create vertical high/low lines at the vertices,
   # and find max & min.
      foreach d $Data($name.date) h $Data($name.high)
        l $Data($name.low) {

    $parent.g_$name marker create line -coords
      [list $d $h $d $l] \
          -outline blue

        if {$l < $min} {set min $l}
        if {$h > $max} {set max $h}
      }

      # Now expand the Y axis to the real min/max range.

      $parent.g_$name axis configure y -max $max
      $parent.g_$name axis configure y -min $min
    }
  return $parent.g_$name
}
```

```
set infl [open $Graph(dataName) r]
readData $infl
close $infl

set w [frame .graphs]

pack $w -side bottom

pack [makeGraph $w $name]
```

java performance

Using Java Reference Objects to Implement Caching

by Glen McCluskey
<glenm@glenmccl.com>

Glen McCluskey is a consultant with 15 years of experience and has focused on programming languages since 1988. He specializes in Java and C++ perfor-mance, testing, and technical documenta-tion areas.

When we hear the term "cache," we often think of hardware, such as L1/L2 caches used with a CPU to implement very fast memory access. We also might think of operating-system caches, for example, a cache of recently used disk blocks. Or we might think of customized caching implemented in an application, such as keeping the most recently accessed record from a database around in hopes of avoiding a costly future lookup.

The Java standard libraries and runtime system offer another style of cache support, one that rests above the hardware and operating-system levels, but that requires support within the Java virtual machine (JVM). The underlying mechanism is a simple one. Suppose I have an arbitrary object of some type, and I say:

```
Object obj = ...
SoftReference ref = new SoftReference(obj);
```

`SoftReference` is a class whose instances are used to wrap other objects. In other words, a object of type `SoftReference` has within itself a reference to another object; this is known as the "referent." If I want to get my referred-to object back out, I can say:

```
obj = ref.get();
```

Reference objects are useful in implementing caches, because the garbage collector in the JVM knows about them and implements specific semantics for such objects. For example, the referent of a `SoftReference` may be cleared by the garbage collector if it needs additional memory and the referred-to object is not actually used anywhere in the application. In such a case the referent is said to be "softly reachable" rather than "strongly reachable."

Suppose that at some point in my program, I have:

```
ref = new SoftReference(obj);
```

with `obj` as some memory-resident object, and then at a later point I say:

```
obj = ref.get();
```

and `obj` is null, that is, the garbage collector has cleared the reference because it needs the memory. In this example, using a `SoftReference` object as a wrapper means that the referred-to object will be kept around if possible, but if memory is required elsewhere, then the object may be discarded if it's not being used in the program.

To see how `SoftReference` works in practice, here is a program that implements a file cache. If a requested file is in memory, a byte vector with its contents is returned; otherwise, the file is read from disk. A hash table is used that maps pathnames to `SoftReference` objects.

```
import java.util.HashMap;
import java.lang.ref.SoftReference;
import java.io.*;

public class FileCache {

  // hash table that maps pathnames -> SoftReference objects

  private HashMap map = new HashMap();

  // read a file into a byte vector and put it in the table

  private byte[] readin(String fn) throws IOException {

    // open the file

    FileInputStream fis = new FileInputStream(fn);

    // read all its bytes

    long filelen = new File(fn).length();
    byte[] vec = new byte[(int)filelen];
    fis.read(vec);
    fis.close();

    // put the (pathname, vector) entry into the hash table

    map.put(fn, new SoftReference(vec));

    return vec;

  }

  // get the byte vector for a file

  public byte[] getFile(String fn) throws IOException {

    // look up the pathname in the hash table

    SoftReference ref = (SoftReference)map.get(fn);
    byte[] vec;

    // if the name is not there, or the referent has been cleared,
    // then read from disk

    if (ref == null || (vec = (byte[])ref.get()) == null) {
      System.err.println("read " + fn + " from disk");
      return readin(fn);
    }
    else {
      System.err.println("read " + fn + " from cache");
      return vec;
```

```
    }
  }

  public static void main(String args[]) throws IOException {

    // set up the cache for files

    FileCache cache = new FileCache();
    byte[] vec;

    // read in big files the first time

    vec = cache.getFile("big1");
    vec = cache.getFile("big2");
    //vec = cache.getFile("big3");

    // read the second time from cache or from disk

    vec = cache.getFile("big1");
    vec = cache.getFile("big2");
  }
}
```

This program reads three large files called `big1`, `big2`, and `big3`. You create them using this program:

```
import java.io.*;

public class MakeBig {

  public static void main(String args[]) throws IOException {
    if (args.length != 2) {
      System.err.println("Usage: filename length(K)");
      System.exit(1);
    }

    // open output file

    FileOutputStream fos = new FileOutputStream(args[0]);
    int len = Integer.parseInt(args[1]);
    byte[] vec = new byte[1024];

    // write out 1K chunks to the file

    for (int i = 0; i < vec.length; i++)
      vec[i] = (byte)'*';
    for (int i = 0; i < len; i++)
      fos.write(vec);

    fos.close();
  }
}
```

You create the files by saying:

```
$ java MakeBig big1 5000
$ java MakeBig big2 5000
$ java MakeBig big3 5000
```

When the cache program is run on my machine, the results vary depending on whether the third `cache.getFile()` line in the first group is commented. If this line is commented, then the second set of `getFile()` calls will read from cache, but if the line is uncommented, then the second set reads from disk again.

In other words, changing the numbers and sizes of files that the program reads affects memory usage and garbage collection, and because of this, the referents of `SoftReference` objects are affected as well. Your results will vary, depending on what JVM you're using, how much memory you have available, sizes of files, and so on.

There are several other reference types similar to `SoftReference`. For example, the class `java.util.WeakHashMap` is based on `WeakReference` and is a hash table with the property that unused keys are discarded over time, via an interaction between the garbage collector and `WeakHashMap`.

Active Content in Java

by Prithvi Rao

<prithvi+@ux4.sp.cs.cmu.edu>

Prithvi Rao is the co-founder of KiwiLabs, which specializes in software engineering methodology and Java/CORBA training. He has also worked on the development of the MACH OS and a real-time version of MACH. He is an adjunct faculty at Carnegie Mellon and teaches in the Heinz School of Public Policy and Management.

Drawing and animation are integral parts of writing Java applications. Applets frequently contain various AWT (Abstract Windowing Toolkit) Components that are part of a graphics context. When they need to be redrawn, the AWT starts with the top Component in the hierarchy and works its way down to the bottom.

The purpose of this article is to expose the reader to various capabilities that are part of the "Active Content" in Java. I'll focus on issues related to the drawing model, drawing shapes, the graphics context, drawing text, measuring text images, and loading them both synchronously and asynchronously. I'll also touch briefly on eliminating flashing and animating images. This information provides for a richer knowledge base from which to venture into writing more interesting Java programs with active content.

The Drawing Model

Redrawing is performed when the AWT starts with the topmost `Component`in the hierarchy and works its way down to the bottom `Component`. Given that containers are Components as well, this applies equally to applets and panels.

Each `Component` draws itself before it draws any of the C`Component` that it contains. This ensures that a Panel's background is visible only where it isn't covered by one of the Components embedded in it.

Programs draw only when the AWT tells them to do so. For instance, this can happen when a `Component` becomes uncovered by the user. Another example is when there is a change in the data being reflected by the`Component`.

The AWT requests that a `Component` draw itself by invoking the`Component`'s `update()` method. The default `Component` implementation of`update()` clears its clipping area in the current background color and then calls the `paint()` method. The default implementation of `paint()` does nothing.

Another way to look at this is that a programmer causes a refresh that's due to some change in state of the control by calling `repaint()`, which in turn calls `update()`. It is usually the case that the `paint()` method is overridden to provide component-specific drawing. It is also possible to override`update()` for more sophisticated results. However, it is possible to call`paint()` directly without calling `update()`, so `paint()` must always be implemented.

Note that `update()` sets the clipping region, and so if the object being drawn changes between drawings, it is possible to end up with half of one and half of the other.

As can be anticipated, the `paint()` and `update()` methods must execute very quickly to ensure acceptable performance. (Note: This is where it becomes advantageous to use threads with applets to achieve better performance. My December 1999 "Using Java" article, "Using Threads Within Applets," discusses this topic.)

The Graphics Context

The argument to the `paint()` and `update()` methods is a Graphics object that represents the context in which the Component can perform its drawing. The Graphics class provides for drawing and filling rectangles, arcs, lines, ovals and polygons, text and images.

It is also possible to "set" and "get" the current color, font, or clipping area, and to set the paint mode.

The `Color` class encapsulates an "r.g.b" triplet, and color is set by:

```
Graphics.setColor(Color color);
```

How text is drawn is defined by the Font class that encapsulates a font (e.g., Times Roman or Helvetica). It also supports point size and a style, although currently only bold, italic, and plain are supported. An example of setting the font is:

```
Graphics.setFont(Font font);
```

Drawing Shapes and Text

The Graphics class has a multitude of methods that permit the drawing of various shapes, including:

```
Graphics.drawRect(int x, int y, int width, int height);
```
for drawing rectangles
```
Graphics.fillRect(int x, int y, int width, int height);
```
for filling a rectangle
```
Graphics.drawRoundRect(int x, int y, int width, int height, int arcwidth, int
archeight);
Graphics.fillRoundRect(int x, int y, int width, int height, int arcwidth, int
archeight);
```

When drawing text it is better to first consider whether it is possible to use a text-oriented Component such as the `Label`, `TextField`, or `TextArea` class. The alternative is to use `drawBytes()`, `drawChars()`, or `drawString()`. For example you can "draw" a string as follows:

```
Graphics.drawString("This is an example of drawing a string", x, y);
```

where x and y specify the coordinates for "drawing" the text.

Measuring Text

In order to determine if text can fit inside a certain area, it is necessary to query the characteristics of the Font using `getFontMetrics()`, which returns a `FontMetrics` object corresponding to a given Font:

```
FontMetrics foo = Graphics.getFontMetrics(Font f);
```

Some of the most commonly used methods that are part of the `FontMetrics`class are `charWidth()`, `getHeight()`, `getAscent()`, `getDescent()`, and`stringWidth()`.

Images and Loading Images Asynchronously

All images are encapsulated by the image class. There are two ways to load images, `Applet.getImage(URL)` and `Toolkit.getImage(filename/URL)`. Both of these load an image, and return an Image object. The method`getImage()` returns immediately without checking whether the image data exists or whether it's been successfully loaded. This is done to improve performance, since it is not necessary to wait for an image to be loaded before going on to perform other functions in the application. Some examples of loading images are:

```
Image image = Applet.getImage(getDocumentBase(), "Image.jpeg");
Image image = Toolkit.getDefaultToolkit().getImage( new/URL
   ("http://www.sample.com/images/sample.gif"));
```

In order to draw the image it is necessary to use one of the Graphics.drawImage() variants. In other

words, `drawImage()` allows you to specify the image to draw and the positioning and scaling of the image. It also allows the specification of the color to draw underneath the image. This is useful if the image contains transparent pixels.

The `ImageObserver` parameter specifies an object that implements the`ImageObserver` interface. This object will be notified whenever new information regarding the image becomes available. The `Component` class implements the `ImageObserver` interface to invoke the `repaint()` method as the image data is loaded. The method `drawImage()` returns immediately even if the image data has not been completely loaded; this results in the image being displayed partially or incrementally. The easiest way to track the loading of images and to make sure that `drawImage()` draws only complete images is to use the `MediaTracker` class. The sequence is:

1. Create a MediaTracker instance.
2. Tell it to track one or more images.
3. Ask the MediaTracker the status of images.

The following code examples elucidates this point:

```
tracker = new MediaTracker(this);
images = new Image(num_images);
for (int i=0; i
 images[i] = getImage(this.getDocumentBase()."image" + i);
tracker.addImage(images[i],i);
}
```

and then later in the program

```
for (int i=0; i
  this.showStatus("Loading image: "+i);
 tracker.waitForID(i);
 if(tracker.isErrorID(i)) {
   showStatus("Error loading image"+i);
   return;
 }
}
showStatus("Loading images done.");
```

Animation and Thread Management

Animation is perceived motion accomplished by sequencing rapidly through frames. Frames can be incrementally different images or graphics operations. A good rule of thumb is that animation should run on a separate thread in order not to adversely affect event handling. The typical sequence is:

1. Implement the `run()` method to increment a frame.
2. Perform a `repaint()` operation.
3. Perform a `sleep()` operation to delay the frame.

Incorporating threading with animation permits the suspension and resumption of animation with button or mouse events.

To suspend or resume animation in an applet when the user leaves the page, it is necessary to reimplement the applet's `stop()` and `start()` methods. The following code example shows how to handle a button event:

```
public boolean handleEvent(Event e) {
 if (e.id == Event.ACTION_EVENT) {
   if (e.target == start)
    start();
  else
```

```
   if (target == stop)
     stop();
    }
    return super.handleEvent(e);
}
private Thread thread = null;
public void start() {
 if (thread == null) {
   thread = new Thread(this);
  start.disable();
  stop.enable();
  }
}
public void stop() {
  if ((thread != null) && thread.isAlive())
    thread.stop();
  thread = null;
  start.enable();
  stop.disable();
}
```

A common problem with animation is "flashing," which manifests itself as animation with jitter. One solution to this is to use "double buffering." This involves performing multiple graphics operations on an undisplayed graphics buffer and then displaying the completed image on the screen. Besides preventing incomplete images from being drawn to the screen, double buffering improves drawing performance. This is because drawing to an offscreen image is more efficient than drawing to the screen.

Animating Images

It is likely that loading images will take a long time regardless of whether`MediaTracker` is used or not. In other words, whenever you load an image using a URL, it will be time-consuming. Most of the time is taken up by initiating HTTP connections. Each image file requires a separate HTTP connection, and each connection can take several seconds to initiate.

One way to avoid this performance degradation is to include multiple images in a single file. Performance can be further improved by using some sort of compression scheme, especially one that is designed for moving images. A simple way to do this is to create an image strip, which is a file that contains several images in a row. To draw an image from the strip it is necessary to first set the size of one image, then perform a draw operation on the image strip shifted to the left so that only the image desired appears within the clipping area.

Conclusion

Writing Java programs that contain active content can be very frustrating but also very rewarding. While it may be simple to rapidly prototype an applet with animation and images, fine-tuning it to meet stringent performance requirements requires more than a cursory knowledge of Java.

The JDK provides a versatile and powerful collection of packages to facilitate writing Java programs with active content. The common theme is always to work with the current graphics context and to use classes such as the `FontMetrics` class to determine sizing of text regions.

the network police blotter

by Marcus J. Ranum
<mjr@nfr.net>

Marcus J. Ranum is CTO of Network Flight Recorder, Inc. He's the author of several security products and of a book on computer security (with Dan Geer and Avi Rubin) and is a part-time sysadmin.

Some More Results

My contest on "Where do packets go when they die?" was a complete failure. I got only one response, which means that my concept was clearly unfunny and uninteresting. After the spectacular success of the Haiku contest, I think I'll be hard-pressed to come up with another exciting avenue for you to show off your creative juices, but check the end of the column for details. By next issue's column I hope I'll have some printable results from the "dirty tricks" contest.

Enough Waffling, Already

For the last year I've been dancing around a set of issues that have really been bugging me: the social problems behind computer security. These have to do with media attitudes, corporate attitudes, and, most important, the attitudes of security professionals. All the pieces of the jigsaw puzzle finally clicked into place for me a couple of months ago and, as a result, I've been taking a much less forgiving position on a number of issues, primarily those related to how security bugs are discussed and how security professionals share information. Now I'm officially throwing my hat in the ring as a hard-liner on professional ethics for security practitioners — and I'm going to urge you to join me in saying "enough is enough."

A few weeks ago, I had the honor of giving a keynote talk at The Internet Security Conference (TISC) in San Jose. As usual, I wrote my presentation the night before, and allowed my talk to follow the outline of my viewgraphs, while accreting free-associated thoughts on the way. The results were unexpectedly strong, amounting to a prediction that eventually we security professionals (backed by civil litigators and law enforcement) will declare unrestricted war on hackers. I've made an MP3 version of the talk available on my USENIX Web page <*http://pubweb.nfr.net/~mjr/usenix/index.shtml*> for those of you interested in the talk. So, now I'm officially out on a limb, and I'm looking forward with distaste to the usual mature high-level debate I have come to expect from the hacker community: overflowing email boxes, Web site denial of service, problems with phone bills and my voicemail.

The Three Phases of Internet Security

In the first phase of Internet Security, we pretty much ignored the problem. This era was typified by an "It's OK, it's just fun-loving hijinks and curiosity" attitude. Back in the first phase, we had things like the gnu.ai.mit.edu server, which provided what amounted to anonymous shell accounts to the Internet at large. Not surprisingly, a lot of hostile network activity originated from that site. But in those days "dotcom" madness had not yet set in, the Internet was not yet a gold rush, and the only people who were suffering were relatively unimportant: system administrators and network managers. The attitude of the day was playful fun-loving curiosity and willingness to forgive.

In the second phase of Internet Security, we built firewalls, vulnerability-assessment tools, and intrusion-detection systems. We're still there today. This phase began in the mid to late 1980s and peaked in the early "dotcom" madness of 1992—1994, when everyone finally started installing firewalls. Network and system administrators had to become like urban homeowners: constantly checking on the state of their locks, burglar alarms, and barred windows. Today's attitude is typified by a mentality of hunkering down behind walls and defenses and hoping the bad guys will go away.

In the third phase of Internet Security, we will lash out in anger, seeking retribution. The firewalls will be supplemented with tort lawyers, and the intrusion-detection systems will become sources of evidence. I think we'll be there within a year, and it'll peak in five years. Law enforcement's efforts will fall by the wayside as people realize that they can hurt their tormentors much faster and more effectively by suing them for damages than by trying to put them in prison. Ever see *The Omega Man*? The attitude of the future will be typified by a mentality of hunkering down behind a wall with a loaded sniper rifle and the phone number of a good ambulance chaser on your speed dialer.

I'm not sure the third phase will be particularly fun, but having lived through the first two, I don't think it's going to be a whole lot worse. It might even be better, since a lot of the casual script kiddies will dry up and blow away, leaving us with only the cadre of dedicated troublemakers to deal with. Fear of litigation will also cause a lot of sites to tighten up their acts. I'm glad I'm not the system administrator at a university or anyplace with deep-pocket financing and a large body of uncontrolled and unmonitored users.

The kind of draconian monitoring requirements that lawyers will start recommending could get really ugly. The downstream-liability lawsuits you'll face if you can't deflect liability onto your users will be uglier. Other folks who are going to have problems with downstream-liability lawsuits are all those sites that distribute security exploits and hacking tools, or teach hacking techniques. Look at the kind of troubles publishers who sell "mayhem manuals" have had, multiply that by a few thousand, and that should give a rough approximation of the magnitude of pain that may be felt.

I'm a big fan of the various low-numbered amendments to the constitution, especially the first. Censorship's a bad idea. However, we've shown time and again that as a society we are willing to apply the screws to people who use their freedom of speech irresponsibly. And I'm now of the opinion that the hackers have garnered themselves enough negative attention that they're about to be on the receiving end of a little good-natured crushing. Too bad it's come to this, but I'm going to save my sympathy for someone who deserves it.

Ethics for Security Professionals

A distressingly large number of security professionals have come up through the ranks by starting as "black hat" hackers who have subsequently "gone legit." I used to believe that this happened because they saw the error of their ways and changed sides in an attempt to help improve the situation — to be part of the solution rather than the problem. Now, however, I'm convinced that the main reason they're doing so is because they've realized that they can make a ton of money, disclaim responsibility for their actions, and continue to do pretty much the same kinds of things they were doing before.

"Ethical Hacking" has been widely marketed as an essential tool in information security. Former "black hats" in several cases have managed to parlay their old collections of attack tools into millions (and even tens of millions) of dollars. Hackers whose claim to fame is a criminal record for cyber-crime can make six figures as "Ethical Hackers" working for audit firms. At the same time, their buddies are releasing a veritable flood of new tools designed to exploit vulnerabilities in commercial products, ostensibly to "illustrate the seriousness of the problem" or to "promote vendors taking security seriously." Some individuals who work as engineers for security companies spend their free time writing and distributing the very attack tools their corporate masters sell you products to protect against. Conferences like Interop, SANS, and TISC are offering classes in "how to hack" that basically train analysts in how to assess and exploit vulnerabilities in systems and networks. They know better, but the classes are sold-out successes and the money's too good to refuse.

I don't know about you, but I'm thinking this is one highly messed-up situation.

I'm not a believer in standards bodies, but I think someone needs to propose a standard of ethics for security practitioners. I'll give it a shot:

1. If I have been involved in cyber-crimes in the past, I have renounced and will denounce such activities in the future.
2. If I know individuals or groups who are involved in cyber-crimes, I will not encourage them, share information with them, or in any way aid or abet their activities.
3. I will never produce or distribute attack tools or tools designed to evade or bypass security systems.[1]
4. I will never teach others how to compromise system security; rather, I shall strive to teach them how to defend systems against compromise.

5. I will never publish designs for new attacks or new attack tools, nor will I speculate about potential vulnerabilities in future or existing systems except in the context of offering ways to prevent them.
6. If I discover a vulnerability, I will work with the author or vendor in a responsible manner to ensure that it is fixed with minimal impact on the user community. If the vendor does not act responsibly in addressing the vulnerability, I will use appropriate channels to inform the user community of the presence of the vulnerability but will not provide exploitable information except to the vendor and responsible parties.
7. If I develop security-critical software I will, at all times, be prompt in responding to or fixing vulnerabilities found in my software.
8. I will not employ individuals I believe might be cyber-criminals or ex—cyber-criminals unless I am certain that they are adhering strictly to this standard of ethics.

I welcome comments on the thoughts above! If you're in the security business, and you agree with me, please share them with others.

My belief is that if the security community doesn't clean up its act, a flurry of lawsuits will. Maybe that's wishful thinking. But I know that if I were so unwise as to develop and release an attack tool, the last thing on earth I'd do is sign my handiwork as many tool-writers do, especially not the denial-of-service tools. Guys, you're painting targets on your backs that could attract years of lawsuits downstream. At the very least, they might someday cost you a job.[2]

If I ran a security conference, the last thing I'd teach attendees is how to hack people's sites. Conferences have deep pockets. I know that if I were running a security-related Web site (actually, I do . . .) the last thing I'd do is redistribute attack tools and exploit information. Not only is that a huge disservice to the community, but I bet any lawyer you told about it would faint. You don't have a lawyer yet? Don't worry, you will.

U Help Me I Lame

Since I authored the original Internet firewalls FAQ and have posted a lot of security-related messages in the last 13+ years, I get emails about once a week from people asking for help in compromising the security of this, that, or the other. The one below is a real sample I got recently:

```
From: [Deleted]@hotmail.com
To: mjr@nfr.net
Subject: hello sir ! ! ! ! ! !
dear sir,

I have read u r "article" at http://www.hackingexposed.com/ I find it very
nice,plz. send u r next papers also to me, if possible. I am also in the
hacking,i was caught 7 times,I want Do my Carrier in HACKING if possible.I
want u to suggest me some way to a big Bully.

thanking u in an anticipation ,

u r's faithfully,
```

Lame, isn't it? This is what we're up against. Caught seven times and he still doesn't get it.

Next Up

In my next column I plan to attack a concept that has been very widely accepted in the security community — the notion of "full disclosure" in bug/vulnerability reports. It's an old idea, and a lot of people place considerable stock in it. I'm sure my popularity will hit an all-time low. In fact, I suspect that a number of people will disagree strongly with the opinions I've voiced in this column. So I'd like to make this issue's contest the "Disagree with Marcus Contest." I'll give a cool Network Police windbreaker to the person who emails me the best brief rebuttal to my position. I'll include it (or a summary if it's not brief enough) in a future column. Mail to <*mjr@nfr.net*> with the subject line "column rebuttal" if you've got something to say.

Forgive me if I'm passionate about this stuff; I've been working in this field a long time, and as far as I can tell things are getting worse faster than they're getting better. So: no more Mr. Nice Guy.

NOTES

1. I'm kind of red-faced on this one. When I was an undergraduate in 1984 I wrote a program called cloak which was designed to let me disappear from our UNIX system so I could play empire without getting caught breaking the university's "no games" policy. There are probably copies of it still out there.

2. You won't work for me, anyhow. My company is the only security-products company I know of that has a policy of not hiring cyber-criminals or even ex-cyber-criminals.

Austin Group Status Update

by Andrew Josey
<a.josey@opengroup.org>

Andrew Josey is the director, server platforms, for The Open Group in Reading, England, and the chair of the Austin Group.

This article contains a brief status update for the Austin Group, the joint technical working group established to consider the matter of a common revision of ISO/IEC 9945-1, ISO/IEC 9945-2, IEEE Std 1003.1, IEEE Std 1003.2, and the appropriate parts of the Single UNIX Specification.

There are now 245 participants in the Austin Group from a wide mix of interest groups.

The fifth meeting of the joint technical working group was held at the Open Group offices in Reading on May 15—19, 2000. The objective of this meeting was to review the Draft 3 specifications.

Over 1140 change requests were processed at the five-day meeting, at an average of 43 requests per hour. This was down slightly from 47 requests per hour at the previous plenary in Montreal.

Summary status is as follows:

- Draft 1 published June 1999.
- Draft 2 published October 1999.
- Draft 3 published February 2000. (3196 pages).

Draft 3 was the result of much teamwork, with many members working around the clock for several weeks to get the draft out on time. Altogether, over 1,100 editorial "aardvarks" (aardvark is the commenting mechanism used by the group) were submitted by the team to enable the merge of the new documents.

CURRENT BALLOTING STATUS

The IEEE ballot invitation has gone out.

At the ISO/IEC level, it has been approved for concurrent CD registration and final CD ballot by WG15, but this has to be forwarded to SC22 for approval.

Of the over 1,100 aardvark comments received against Draft 3, many were for C99 alignment. It was observed that the quality of the aardvarks submitted by some reviewers could be improved and that future plenary meetings would have to reject malformed (such as wrong line numbers, and missing or incomplete actions) requests as nonresponsive. The next draft (Draft 4) will be produced at the end of July and will fold in the changes identified by the review of Draft 3, fold in 1003.1q, and produce the rationale volume (XRAT). Draft 4 will be feature-complete — no new features are expected to be added after that point. Draft 4 is scheduled for August 1, 2000, and will be available for an eight-week review period closing September 26, 2000. The formal review period for Draft 4 will include concurrent IEEE and ISO/IEC balloting. The next plenary meeting will be October 9—13, 2000, to review aardvarks arising from Draft 4. This meeting will be held in the USA at a location to be confirmed. To participate in the revision, see <*http://www.opengroup.org/austin/*>.

musings

by Rik Farrow
<*rik@spirit.com*>

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*.

Memorial Day in the U.S. has come and gone, with the usual ceremonies to remember war dead as well as the survivors. I tacked a visit to the Vietnam War Memorial onto a family visit in Washington, D.C., not so much because of Memorial Day, but rather because I had forgotten it was Memorial Day, and I had never seen the complex.

At the time of the Vietnam War, I was adamantly opposed to sending troops there. It was their civil war, and we wound up fighting our own WWII ally, Ho Chi Minh, under the illusion that if Vietnam became Communist, other countries would fall under Communism like dominoes. In no time, the Australians would have been fighting guerrillas hiding within the dense jungles in their arid interior. Right. Instead, millions of Vietnamese have suffered, and many have died, along with millions of American casualties.

What really struck me, as I walked with thousands of others past the Wall, was just how many names there were. Over 50,000 Americans died, but you don't really know how many people that is until you watch the Wall grow taller, with more names in each column, going on and on until I was numb. Rolling Thunder, a group that demonstrates for the return of Vietnam Missing in Action, was there. Their parade on Sunday consisted of over a hundred thousand motorcyclists rolling on a thunderous parade route around the Capital mall. Yet the antipathy that had existed between the vets and the anti-war protestors during the seventies appeared to have vanished.

My own feelings today about armed forces are that I am glad they are on my side. Their job is still a necessary one, and often a dangerous one, at that. If you want to get a better understanding of modern warfare, what it is like to be under fire, as well as to comprehend a failure in foreign policy, read *Black Hawk Down*, by Mark Bowden. I often wondered at the silence of vets when asked about their combat experiences, and this book explains, through the eyes of combatants, why people just don't talk about these things. My heart goes out to any veteran of any conflict anywhere, with the prayer that someday we can eliminate organized death and mayhem.

DNS, the Forgotten Story

Speaking of heroes of less violent conflicts, DNS has become a casualty quite frequently of late. Or perhaps I should say that DNS has become one of the most common ways to break into some UNIX and Linux systems remotely. And, in an embarrassing turn of events, one of the security extensions to DNS paved the way for this buffer overflow.

The latest buffer overflow is not the only one. Up until two years ago, most Linux systems were distributed with BIND version 4.9.6, with FAKE-IQUERY enabled. While many other versions of UNIX used the same version, they did not support FAKE-IQUERY. The exploit for this works by sending a request that contains a value that overflows an internal buffer in named, providing a root-owned shell. CA-98.14 covers this issue in BIND, as well as several other denial-of-service attacks, and cache corruption.

The newer releases of BIND are much more resistant to cache corruption, as they only accept responses that match outstanding requests unlike the old version, which would accept an unsolicited response. The newer cache-corruption tools must first collect a request ID, send a request for the site to spoof to the victim DNS server, and then send the phony answer with the correct request ID (based on the previously collected request ID). The OpenBSD version of named randomizes request IDs, making this much more difficult to accomplish.

What has been causing a lot of problems is a buffer overflow in BIND 8.2. If you have installed patch 5 and are running BIND 8.2P5 (or earlier than 9.2 or later versions), you do not have this problem. You can determine the version of BIND that you are running by executing named -v locally, or using dig to query a nameserver remotely:

```
dig bear.spirit.com version.bind chaos txt
```

Look in the ANSWER section of the result for VERSION.BIND, and hope to see something like "8.2.2-P5."

There has been a lot of confusion around one particular exploit (apparently written by a small group called ADM crew), as it takes advantage of a new resource record (RR) named NXT. Until recently, I had no idea what NXT meant, although I correctly assumed that you could pronounce it "next." NXT is part of the DNS security extensions, or DNSSEC, and you can find out more about this by reading RFC 2535 (<*http://www.faqs.org/rfcs/rfc2535.html*>). I am looking for someone who can provide more detailed information about DNSSEC, but I will share some of what I learned from the RFC and from some email exchanges.

No More Corruption

The only sure way to end DNS cache corruption is to create DNS servers that can include digital signatures with their answers, and servers that can validate those signatures when they are received. BIND 8.2 includes these features, supported through the use of three new RRs — SIG, PUB, and NXT. SIG RRs provide digital signatures for named resources. The signatures are multiline data structures that define what is being signed and how, and include the signature itself.

The PUB RRs provide the public key for the DNS server itself. For this to really work, a server's PUB key must be signed by a DNS server in a super zone, and the receiving server must be configured with public keys for the root servers, so that the chain of public keys can be authenticated. Note that this is a little like a PKI. There is a project going on in Europe to set up DNSSEC in Sweden, Germany, and the Netherlands by the middle of 2001, and I am very interested in seeing the results. One thing I find myself wondering about is all of those asymmetric encryptions being done to verify signatures, and how much load this will add to a busy DNS server.

The NXT RR actually stands for "non-existence record." I had to read through this section carefully, because it is not intuitively obvious (at least for me) how you could cover all non-existing records in your DNS database. The concept is actually much simpler, as it appears that instead of sending back an error when information is requested that does not appear in the zone database, two NXT RRs are sent back. The NXT RRs provide proof that the requested record really does not exist. Please read section 5 of RFC 2535 for details.

So, if you don't even know about NXT records, how can you be vulnerable to an attack using them? Well, that's easy enough. Someone will send you a NXT record with a buffer overflow in one of the fields. That way, you don't need to know anything about NXT RRs to be vulnerable. Note that vulnerable versions of BIND were shipped not only by Linux distributors but also by several BSDs, Sun, IBM, HP, SCO, and Data General. (See<*http://www.cert.org/advisories/CA-99-14-bind.html*>.)

I had to find a version of the exploit, plus RFC 2535, before I could begin to understand all of this. You can find the version I examined at<*ftp://ftp.technotronic.com/unix/nameserver-exploits/t666.c*>. This version has been modified so that it will not work without some changes to the shellcode, unless you attack a system that you control and copy a shell to `/adm/sh`. This version also does not explain how to tickle the remote DNS server into querying your evil server either, but I imagine that cache corruption tools like `erect` or `jizz` would help here.

An interesting aspect of t666.c is that the comments claim to be able to break out of a `chroot` jail. The `chroot()` system call sets a string in the process's user area that changes the process's root. Once this system call has been executed by root, all paths appear relative to the new root. If you are interested in how you would set this up for Linux and the patched version of BIND, you can read <*http://metalab.unc.edu/pub/Linux/docs/HOWTO/Chroot-BIND-HOWTO*>. Although I was not willing to decode most of the shellcode, the usual technique to escape a chrooted environment is to find an open file descriptor for a directory outside of the jail, and to use that to escape. I believe that BIND 8.2P5 fixes this by closing all file descriptors opened before the change root.

Ubiquitous

One of my favorite words, ubiquitous means "ever-present." The vulnerable version of BIND, 8.2, also appeared to be nearly ubiquitous. A survey of DNS servers, using the inverse in-addr.arpa tree by Bill Manning

(<*http://www.isi.edu/~bmanning/in-addr-data.html*>) shows that as of second quarter 2000, 13% of all DNS servers — about 19,000 of them, that is — were running 8.2.

But CERT pointed out in CA-2000-03 that this is only the tip of the iceberg. Many default installations of Linux, BSD, or UNIX will start up `named`. The unconfigured `named` will function as a local server, as well as be remotely detectable by scanning for an open port 53. Just try to imagine how many people have installed Red Hat 6 (with BIND 8.2), and left `named` running, just waiting for someone to notice the open port 53, verify the BIND version (if they even bother), and run the exploit.

CERT also pointed out another thing I find sadly funny. Besides the usual tricks done after a completed break-in (install rootkits, set up backdoors, edit logs), attackers have taken to installing patched versions of BIND. I do not consider this an act of goodwill, but merely another way to hide the intrusion (as well as making the system less likely to be "owned" by yet another attacker).

Sysadmins, sweep your networks! Commercial tools will find the buggy version of BIND, as well tools like `nmap` set to scan for an open port 53. The use of a properly configured firewall (one that, at the very least, limits incoming connections to known servers) will also do a lot to contain the damage. Note that FireWall-1 v4 (with no patches) permitted access to port 53 TCP/UDP by default, and you should change this.

VMware

I finally got a chance to install the 2.0 version of VMware on my little notebook. VMware allows me to run NT4 under Linux, and the new version does run faster. It also has a suspend feature, so you can suspend NT and resume it later without having to go through the whole reboot process. I like this: until now, before I could start teaching I had to boot first Linux, then VMware and NT4.

I have also been playing around with Java again. I am not particularly fond of PowerPoint, and I've wanted to include animations in my presentations for a long time. I downloaded a version of Java 1.1 from Blackdown.org and started writing my own display tool. Right now it runs simple animations (labeled boxes move across the screen, allowing me to demonstrate the infamous TCP three-way handshake, as well as scanning), and I can display images. Now I need to add keyboard control, as well as the ability to display text, and I might have something fun to use.

On the Microsoft front, not much has happened. Well, there are about 150 class-action lawsuits on behalf of people who had to pay for a Windows license when they bought their computer to run Linux, *BSD, or BeOS. And there have been some wonderful worms and viruses floating about, such as ILOVEYOU! and the resume virus. Microsoft had released its thirty-ninth security advisory of the year (already heading for a record-breaking year), which leads me to suggest the following to the Clinton administration:

- Microsoft should be broken up into at least two separate companies for national-security reasons.
- My argument is simple. In its present state, Microsoft can continue to embed features in its operating system that support not only the applications it writes, but copious exploits as well. Who could have imagined that an operating system and its related applications would, in the Internet age, make it so simple to download and execute code, jeopardizing the security of most desktop systems? Breaking up Microsoft should not be considered punishment, or bad for the U.S. economy, but an action taken in the hope that Microsoft will be forced to start designing secure application-language interfaces with published specifications.

I know, fat chance, but I felt I had to say something.

POSIX System Services Working Group — Realtime (SSWG-RT)

by Joe Gwinn
<gwinn@res.ray.com>

Joe Gwinn is the chair of the POSIX System Services Working Group — Realtime (SSWG-RT).

The POSIX System Services Working Group — Realtime (SSWG-RT) is considering amendments to the realtime POSIX profiles IEEE Std 1003.13-1998, reflecting implementation, and field experience with 1003.13. People wishing to offer their experience with and ideas for improvement of 1003.13 are encouraged to contact the chair of SSWG-RT, Joe Gwinn <*gwinn@res.ray.com*>.

IEEE Std 1003.13-1998 is a family of four related realtime profiles ranging in size from the very small through a full-featured platform conforming to essentially all of 1003.1, 1003.1b (realtime), and 1003.1c (threads), and the parallel Ada binding 1003.5b, with realtime options chosen. The smaller profiles specify just that subset of POSIX interfaces needed to "clothe" widely used small kernels such as pSOS (from ISI), VxWorks (from Wind River), and VRTX32 (now from Mentor), and the ORKID interface standard (from VITA), which, although very similar in approach and function, differ greatly in interface details. (As a matter of interest, there are more of these small kernels in UNIX systems than there are UNIX kernels because, for instance, many I/O controllers and peripherals themselves use one of these small kernels.)

Standardization of these interfaces will yield the same benefits for embedded and realtime as standardization of UNIX did for workstations. In addition, the P1003.13 interfaces are chosen to allow multi-computer distributed systems to be built, such as those used in factory automation. Such systems are typically set up as a hierarchy, with a few large-profile machines at the top, and a large number of smaller profile machines at the bottom controlling this or that piece of machinery, perhaps with an intermediate layer of supervisory machines between top and base, and all communicating with peers, superiors, and subordinates, as needed.

To order hard copies of standards and draft standards by phone, call the IEEE at 1-800-678-3444 from inside the US and Canada, and +1-908-981-1393 from elsewhere. To order by fax, call +1-908-981-9667. Payment may be made by credit card. In general, standards and draft standards are not available electronically, as the IEEE supports its standards activities by selling these documents.

From the President

by Daniel Geer
<geer@usenix.org>

President, USENIX Board of Directors

If there is anything that attendance at a USENIX general conference gets you, it is a good sampling of what is out there. I mean people, ideas, tools, skills, outré gizmos, clever adaptations, reminders of your own inadequacies, how fast things are changing, how much there is to know, and how much just plain fun it is to breathe that kind of air.

I've been attending for a long time — obviously, or I wouldn't be the damned President. I can remember coming to my first one. Wandered into a marketing talk by Rob Kolstad — Wow! Could he ever sell supercomputers! Hmmm; this is an interesting place. Went to the Reception. Wow! That's Bill Joy! Went up and shook his hand, thanked him for all that BSD stuff I was just learning how to use. Blathered on and made an idiot of myself, I'm sure. Later, and only later, I find out that it really was Dennis Ritchie wearing a badge that said Bill Joy. Never forgave him, but that's why I keep coming. Where else can a story like that happen?

That's my point. People come the first time to see what the fuss is, but they stay for reasons that often are idiosyncratic and have on the face of it little to do with The Mission Statement. This is for all of you who are first- or second-timers — keep coming. Yeah, that sounds lame, but I challenge you to find something better. If you do, tell me, 'cuz I'm serious about excellence without hype. But until you do, come to USENIX and watch for that moment when you say to yourself, "USENIX is only a month away and I can't wait to see my colleagues again."

N Years Ago: Anniversaries

by Peter H. Salus
<peter@Matrix.Net>

USENIX Historian

We emphasize certain anniversaries: lustra (five-year periods) and decades (ten-year periods) especially.

In 1975, on June 18, Mel Ferentz ran a meeting of the "Unix Users' Group" at the City University of New York. There were about 40 attendees.

June 17—20, 1980, saw over ten times that number gather at the University of Delaware. Steve Daniel distributed "A News" there. The first real news reader. NETNEWS had been written by Steve Bellovin and revised by Daniel and Tom Truscott. There were 15 sites on the UUCP network. That Delaware meeting saw Armando Stettner and Bill Munson recruit Bill Shannon to DEC.

June 10—14, 1985, there were 1,730 attendees at the Portland, OR, meeting. Vic Vyssotsky was the keynote speaker. Other speakers were Gordon Bell, Eric Allman, Bjarne Stroustrup, Rob Pike, Dave Presotto, Andy Koenig, Tom Ferrin, Dave Yost, Dennis Ritchie, Don Libes, Mike Hawley, Ed Gould, Sam Leffler, Dave Korn, Kirk McKusick, Mike Karels, Brian Redman, and . . . Tektronix sponsored a truly memorable reception.

June 11—15, 1990, saw about 1,200 gather in Anaheim to listen to Dennis Ritchie talk about "What happens when your kid turns 21?" There were a number of memorable papers (Cheriton, Ousterhout, Farmer, and Spafford, etc.) and a really stunning session on music by Peter Langston and Mike Hawley. Though it may seem absurd, they were even better than they had been in Atlanta in 1986!

USENIX has always been a place where real users got together to pool their knowledge and to renew acquaintance. It's hard to think that in 20 years we've seen the rise and demise of store-and-forward networks; that I can recall V and Mach and Sprite and a host of other systems. The first papers on C++ and on Tcl. The announcement of 3BSD; of 4.4BSD.

Give credit to IBM: Big Blue sponsored the meetings that gave rise to SHARE from 1955 on. Software exchanges are nothing new (Mike O'Brien, take a bow); hardware tips aren't, either (your turn, Tom Ferrin). But from May 1974 to now, USENIX has been a focal point.

Postlude

Tom Duff sent the following:

> I just read your mention of the "10th anniversary" USENIX meeting in Portland. I hadn't thought about that meeting in a long time.
>
> I met my wife Susan for the first time two days before that meeting.
>
> I can't help but remember the fireworks after the outdoor banquet and my visit to the Portland municipal rose garden.
>
> Also, the first (non-Bell Labs) UNIX users' meeting may have been in 1974, but the first International meeting was in NYC, summer of 1975. A delegation from Toronto including Mike Tilson (I think), Ron Baecker, and me drove down in Bill Reeves's orange Datsun micro-car. Surely the meetings before I started attending aren't counted.
>
> Anyway, cheers, and pardon me for interrupting your nap.

The 1999—2000 USACO Report

by Rob Kolstad
<kolstad@usenix.org>

Head Coach, USACO

For several years, USENIX has sponsored the USA Computing Olympiad, an organization that fosters pre-college computing by offering contests whose winners ultimately travel to exotic foreign locales to represent the USA in international competitions.

I am the head coach and work with a fabulous set of people that includes Director Don Piele at the University of Wisconsin, Parkside; Greg Galperin from MIT (now on grad school leave at a startup); Hal Burch (Carnegie Mellon, interning at a startup); Russ Cox (Harvard undergrad); and Brian Dean (MIT grad student). Together we create programming tasks, grade them, evaluate performance, offer feedback and so on.

The 1999—2000 Contests

The USACO offers programming contests over the Internet throughout the school year. These contests are "open entry" (any pre-college student in the world can enter, as long as they have Internet access to see the problems). This year we continued to offer two divisions (one much less challenging than the other) into which students self-classified themselves. International participants often account for more than half of the total entries! Here's a quick table that summarizes participation for the 1999—2000 USACO season:

	Fall 1999		**Winter 2000**		**Spring 2000**		**U.S. Open**
	easy	***hard***	***easy***	***hard***	***easy***	***hard***	***hard***
Total	38	149	46	159	29	150	373
U.S.	9	64	22	54	16	57	221
	(24%)	(43%)	(48%)	(34%)	(55%)	(38%)	(59%)
Non-U.S.	29	85	24	105	13	93	152
	(76%)	(57%)	(52%)	(66%)	(45%)	(62%)	(41%)
Countries	15	31	16	32	9	32	35

The participation from the USA in the first contests was of great concern to me. I knew I was recruiting one to two dozen new programmers for each contest, but the total number of USA contestants was not increasing. I finally went to investigate to see which contestants were returning for future contests and to see if any pattern was recognizable.

The pattern was instantly and obviously clear. Those who score very, very low in a contest (e.g., scoring 80 out of 1,000 points) were quite likely to refrain from returning. Since this amounted to 10—25 contestants each time, the recruiting effort

(magazine articles, newsgroups) was required to yield a couple dozen new entries just to stay even!

Like a good marketer, I polled some members of the departing group and quickly found that they felt they need better training resources in order to compete with the world-class stars in the upper division.

In January, Hal Burch and I set out to create the "USACO Training Pages." Using material from the previous camp, problems from a myriad of sources, and Russ Cox as a problem analyst, we have created roughly 200 hours of training material for our competitors. If you know pre-college programmers who would like to improve their contest programming ability, please send them to <*http://ace.delos.com/usacogate*>.

The system includes sequencing, task grading, and a host of instructional materials on various algorithms. These pages are currently targeted at the student with a year or so of programming.

We're working to extend them "down" to the novice level this year.

The Training Pages have been discreetly announced through the USA since January. Over 300 pre-college students have tried at least some of the pages and the grader has evaluated 8,393 solution attempts submitted in both C/C++ and PASCAL. So far, it appears that the training pages are a super success and are creating students that, if this year's training camp is any indicator, are the equivalent of two training camps ahead of the game!

Regrettably, I cannot attribute the improved U.S. Open attendance results solely to the training pages. We improved our public relations program for announcing the U.S. Open so much that determining the reason for the dramatic increase is not possible.

The 1999—2000 Winners

Ranking the winners across four separate contests, some of which were more challenging than others (e.g., the U.S. Open) is a daunting task.

Consistency is an important factor; sometimes contestants "get lucky" once. Furthermore, the contests span the school year and the skill levels of the contestants change through the year.

The 1999—2000 season saw a spectacular performance by Reid Barton, a home-schooled junior from Boston. He won two of the contests (one of them outright). John Danaher (one of the many talented students at the Thomas Jefferson High School for Science and Technology located in Virginia) placed fifth, first, second, and fifth for a spectacular final year of eligibility. Sophomore Vladimir Novakovski, also a TJ student, earned a first and two seconds for the best performance in his age group. Veteran Percy Liang, a New Mexico student also in his final year, earned a first, second, and third.

The U.S. Open yielded some surprising results. Senior Jack Lindamood, a contender throughout the year, was the only U.S. student to achieve a perfect score and thus earned the U.S. Open championship crown. In his spare time, Jack programs DSP chips for Texas Instru-ments. Only nine points behind, Thuc (sounds like "took") Vu captured second place. A junior, Thuc has been in the U.S. only eight months and is still mastering the English language!

Below is a list of the students chosen to attend the USACO training.camp, an eight-day affair held just south of Milwaukee at the University of Wiscon-sin's Parkside branch.

- Reid Barton, Arlington, MA
- Jacob Burnim, Silver Spring, MD
- Kevin Caffrey, Oakton, VA
- Tomasz Czajka, Stalowa Wola, Poland
- Adam D'Angelo, Redding, CT
- John Danaher, Springfield, VA
- Richard Eager, Alexandria, VA
- Percy Liang, Phoenix, AZ
- Jack Lindamood, Dallas, TX
- Yuran Lu, Presque Isle, ME
- Vladimir Novakovski, Springfield, VA
- Gregory Price, Falls Church, VA
- Gary Sivek, Burke, VA
- Steven Sivek, Burke, VA
- Thuc Vu, Westminster, CA
- Tom Widland, Albuquerque, NM

The list contains a whopping seven students from TJHSST in Virginia, including identical twins Gary and Steven Sivek. We finally have a finalist from California, which is underrepresented in all our contests for no reason I can determine. Danaher, Liang, and Novakovski were our returning veterans. Four campers (Barton, Danaher, Liang, and Lu) earned a head start on this year's international competition at the Balkan Olympiad a couple of months ago. (See the July *;login:* for an account of that competition.)

Tomasz Czajka, a Polish veteran who won several USACO contests, was invited as our international representative this year. He provided stellar competition to our contestants in both formal and informal competitions.

Training Camp

Training camp has two direct goals: improve the skills of our top programmers and identify our four best candidates for gold medals at the international events throughout the rest of this season.

Indirectly, the camp and its rewards provide goals for all US contestants throughout the competitive year.

Contestants arrived at camp on Tuesday, June 20, and departed early on June 28. Two five-hour contest days highlighted the experiences, with dozens of instructional modules, problem attacks, and less formal exercises in between. This year we incorporated several parallel sessions with one coach/four students in an effort to increase personalized instruction and decrease the "Well, Joe knows the answer so I don't need to think" phenomenon. I think our effort was quite successful, and we'll continue this trend.

We implemented Dynamic Program-ming Day, during which Russ Cox organized lectures, drills, lab problems, and even a set of eight different student presentations on dynamic programming. The presentations were stunning and the students nailed the DP problems on the first contest.

The "training notebook," begun in earnest last year, grew to almost 150 pages of tutorials, drills, and programs. We're proud of this resource and believe it helps students throughout their competitive careers.

This year's innovations included not only written solution analyses for the contest problems but also in-depth graphical analyses of the test data and its creation.

Informal events included the Nine Men's Morris competition in which campers wrote programs that played against each other in the traditional English (and other) drinking game. Polish contestant Tomasz Czajka's program was undefeated. A (Frisbee-style) Disc Golf Competition was won by Jacob Burnim, while the traditional USACO Quiz Show was won by Gary Sivek, who edged out his twin brother Steven 2700-2400 in the final round.

The first round of competition at the camp was an easier round with very high scores (mostly due to success on dynamic programming problems). For the second round, we cranked up the difficulty (see below).

After Russ Cox graded all the problems and the group of coaches reviewed those scores and the finishes throughout the year, four students and one alternate were chosen for the international USACO traveling team. They are:

- Reid Barton
- John Danaher
- Percy Liang
- Gregory Price
- Jacob Burnim (Alternate)

L. to R: Gregory Price, Percy Liang, John Danaher, Reid Barton

The next big contest is slated for late August in Cluj, Romania (in the state of Transylvania); the t-shirts will be fabulous. The international finals (IOI — International Olympiad on Informatics) will be held the last week of September in Beijing, China. The U.S. team carries with it high hopes for multiple gold medals.

The Future

It's just amazing how much time an endeavor such as this can take, especially once a publicly available set of Web pages is available. We plan to continue the set of individual contests on the Internet next year with the same style of divisions (or maybe even pseudo-divisions that let contestants evaluate themselves better in appropriate subgroups).

This next year will be a milestone for the competitions, because we are finally leaving behind the 16-bit DOS-based compilers that have been de rigueur since the IOI's inception. These dinosaurs allow contestants access to only 640KB of memory and generate terribly slow code. I have chaired (I think I was the chair, anyway) a committee that is in charge of updating the technology. The current plan is to switch to GNU C/C++ and Free Pascal on the day after the IOI ends in China.

The new compilers enable: automatic problem grading without the problems of rebooting DOS, file systems that might or might not become corrupted after each program execution, lack of multitasking, and lack of memory protection. This changes the landscape for available contest types and dramatically eases grading (i.e., IOI grading for 300 competitors should take closer to a fraction of an hour than the current 0.5 to 2 days).

We're looking to grow the level of our competitions and the number of competitors throughout the next year.

In order to foster computer programming as a more recognized sport, I'm looking at holding a set of dual meets (just like track, tennis, swimming, or football) through the school year. Each meet will pit one school's team against another's — even if the teams are proximal only via the Internet. It's an easy matter to conduct Web registration, offer a variety of scoring mechanisms (like swimming? tennis?) and team organizations (singles, doubles, etc.) to make some exciting competitions (probably more so for the contestants than the spectators).

The parallelism that enables all the competitions to be just the same except for the competitors will be our little high-leverage secret.

The problem-grading system used by the Web training pages has proven very reliable and amazingly efficient. Email responses are returned to the submitter often within just a couple of seconds.

Opportunities

A couple of items really need a large group focus in our effort. First of all, we need publicity. If you know students who might be interested in competing, please send them to <*http://www.usaco.org*> and<*http://ace.delos.com/usacogate*>. The USACO page shows some of the accomplishments of our team over the years and always contains up-to-date info on the latest contests and events.

Second, we need additional programming tasks. These are a very special niche of problem type that require months (half a year? more?) to learn how to create. They suffer from the "Goldilocks Syndrome": they must not be too hard; they must not be too easy; they must be "just right." If you would like to help contribute to our "problem budget" (which could easily exceed 100 problems for the upcoming year), please contact me at<*kolstad@ace.delos.com*>.

Acknowledgments and Thanks

The USENIX Association's sponsorship of USACO is invaluable. Our time is spent creating contests, software, and educational opportunities instead of dressing up and begging for money. We're working hard to improve our program.

Director Don Piele implements a huge number of tasks, including: the USACO Web page, administering the budget, acting as USENIX liaison, serving as camp operations manager, planning all physical plant items for camp (including transportation), and meals. He is like the genie in Aladdin's lamp: wish for something (a three-hole punch, for example) and it appears within hours.

The four coaches (Brian, Greg, Hal, Russ) work with me throughout the year on problem creation, solution creation, test data creation, grading, and answering student questions from the training pages.

All of our assistants donate their time (almost ten days at UW-Parkside just for training camp and its setup). It's a great group to work with and continues to create new and exciting ways to improve our camp.

Thanks!

Two Challenging Tasks

Here's an easier problem (three stars out of four) created this year by Hal Burch. It is cowified to keep with the spirit of attending camp in Wisconsin:

> **Cow Talk**
>
> Like the natives of Africa and their drums, the cows moo to each other over long-distance networks of relay-cows. Cow A moos to Cow B; Cow B passes that message to Cow C, and so on. Cows only moo to specified partners, even if some other cow happens to be close by; and when two cows are partners, it is always the case that either can moo to the other to relay messages.
>
> The speed of sound being finite, some relay links take longer than others. In this communication area, we can calculate the relay time seconds by dividing the distance between the cows by 100. Let's call the set of cows-who-relay along with the list of specific, relaying cow-partners a cow-network. The cow-network is always set up so that it is possible for any cow to talk to any other.
>
> Some cow-networks are better than others. A cow-network's merit is measured in terms of the "average communication time between cows." This number is the sum of the total communication times between all possible pairs of cows divided by the number of pairs of cows; thus a lower number is better. When two cows wish to communicate, they always talk using the relay-cows which give the quickest possible moo transmission.
>
> Given a set of cows, their (x,y) grid locations, and their communication partners, determine the link between two (hitherto unconnected) partners which, when added, creates a cow-network with the smallest possible figure of merit.

This year's hardest problem (four stars) was "Kinergarten" (kine, you know, is an archaic plural for cow). This problem was couched in terms of a teacher taking the young calves on a field trip in which they each held on to a set of strings in order to avoid separating. After a good amount of verbiage, the problem devolves to:

Kinergarten [Burch & Galperin, 2000]

> Given a set of nodes (no more than 150) and weighted arcs between some of them, count the total possible number of different minimal spanning trees. A given arc weight appears no more than 10 times. 640KB memory limit. Five seconds CPU time limit on a 16-bit compiler on a P2/300.

The problem, already quite challenging, is complicated by the fact that the total number doesn't fit in just a few bits, thus requiring the implementation of parts of a "bignum" package just to keep track of the final result.

Conclusion

The USACO is alive and well. Our camp participants thought the organization must have at least several dozen affiliates, given the Web pages, contests, and general "look and feel" of the public interface.

This year was our most successful for increasing participation and raising the bar. The upcoming challenges are dominated by continuing to gain affiliates, maintain our quality, and prepare for the 2003 IOI, which is to be held in the USA. Our biggest challenge for 2003 is finding sponsors for the IOI.

system and network monitoring

Using Big Brother

by John Sellens
<jsellens@gnac.com>

John Sellens is Associate Director, Technical Services, with GNAC in Toronto. He is also proud to be husband to one and father to two.

This article will outline, review, and express opinions on the Big Brother System and Network Monitor. I'll share an overview of Big Brother, relate some of my experience with it, and describe it in terms of the evaluation criteria outlined in the first article (*;login:*, June 2000) in this series on system and network monitoring.

Stealing a sentence from the Big Brother FAQ, "Big Brother is a Web-based Systems and Network monitor written by Sean MacGuire <sean@bb4.com> and Robert-Andre Croteau
<robert@bb4.com>," and it is available at <http://www.bb4.com/>.

Big Brother (BB) was first released in October 1996. As of this writing, I've been running Big Brother version 1.4c on an experimental basis for a little over a month. During that time, the current version of Big Brother has progressed to 1.4g, which, according to the README.CHANGES file, has added a few fixes and a few new features, including a likely to be useful "failover" ability for when your monitoring or paging host fails.

Overview

Big Brother is a collection of Bourne shell scripts and C programs. It consists of four main components or services:

```
BBDISPLAY:   A machine running a Web server that generates and provides the Big
             Brother status pages.
BBNET:       A machine that tests the availability of various network services
             on hosts of interest.
BBPAGER:     A machine that accepts notification requests, interprets the
             notification rules, and sends email or pager messages.
Client:      A machine running the Big Brother client software that performs
             various local tests and reports the results to the BBDISPLAY and
             BBPAGER servers.
```

Big Brother is primarily a monitoring system for UNIX hosts, but it can be used to monitor network services on just about any network-connected device. BB client implementations for Windows NT and Novell are available.

Configuration of Big Brother is done through a number of text files contained in the BB build/installation directory hierarchy. Once configured, the primary status interface to BB is through Web pages that are regularly rebuilt by the Big Brother daemon.

The Web interface provides a comfortable, understandable, and portable interface to status, history, and other reports. Through various configuration and HTML fragment files, Big Brother can be configured to report in multiple groups, pages, and hierarchies. And it's possible to report status summary information to other BBDISPLAY servers to create a server hierarchy.

The design and implementation of Big Brother make it easy to extend. Other monitors or probes can use the BB tools for reporting and logging by sending simple one-line messages to the BBDISPLAY and BBPAGER servers.

Evaluation Criteria

SIZE AND COMPLEXITY

Big Brother is intended to be a simple, relatively lightweight monitoring system that handles the most common needs in a reasonable fashion. It generally succeeds in its goals, with a few exceptions.

I found the configuration of Big Brother to be overly convoluted, with too many options in too many places. (More on this below.) Additionally, installing the BB client on a new host apparently can't be done by just copying from an existing host — the instructions require using the bbclient command to create a host-specific tarball for each new client.

The fact that Big Brother is implemented in sh and C is both good and bad. It means that Big Brother is fairly easy to port and will run on just about any (UNIX) machine with a minimal software base, but it also means that the resultant implementation is sometimes more convoluted and inefficient than it might otherwise be. For example, the `bb-network.sh` script uses about 20 lines of commands to look up network services in the `/etc/services` file, each time through, for each service, and for each host being monitored. A different implementation could allow the use of services' file-entry caching and the`getservbyname()` library routine, which will work better in an environment that relies on NIS.

SCALABILITY

Because of its implementation, its multiple configuration mechanisms, and its use of a nonfilterable Web interface for display, my expectation would be that Big Brother won't scale very well beyond perhaps 50 or 75 systems or devices. While I'm sure that there are BB installations larger than that, my impression is that Big Brother is best suited for small- to medium-sized installations.

RELIABILITY

With proper configuration and installation, Big Brother can be fairly reliable. If BBDISPLAY and BBNET are actually different hosts, the net effect is that they (mostly) watch each other — BBNET will send notification messages if BBDISPLAY is down, and BBDISPLAY will complain about stale information if BBNET is down. With careful planning, and a certain amount of replication, you can do fairly effective monitoring of all your hosts, including the monitoring hosts themselves.

Some of the Big Brother ancillary scripts seem to be a little fragile. For example, the `bbchkcfg.sh` script (which checks certain BB configuration files for errors) seems to work only if you run it from the BB etc directory and fails in curious ways if you don't. Similarly, the `install/bbclient` script needs to be run from the install directory. And, as a final example, the `runbb.sh` start/stop script fails in an ugly way if you ask it to stop Big Brother when BB isn't already running:

```
% ./runbb.sh stop Stopping Big Brother... cat:
/home/jsellens/bigbrother/bb14c/tmp/BBPID: No such file or directory usage:
                        kill [-s signal_name] pid ...
                        kill -l [exit_status]
                        kill -signal_name pid ...
                        kill -signal_number pid ...
rm: /home/jsellens/bigbrother/bb14c/tmp/BBPID: No such file or directory
```

Admittedly, these kinds of errors don't have much impact on BB's ability to operate correctly under normal circumstances, but they did make me wonder about the overall reliability of the Big Brother implementation.

COST

Unless you're reselling monitoring services or the Big Brother software itself, there is no charge for Big Brother. You can choose to buy a support contract or support the Big Brother project by purchasing a license, but otherwise the cost for Big Brother is limited to the time you take to install and configure the software.

NUMBER AND TYPE OF PROBES

Big Brother comes ready to monitor the most common system attributes and network services: network connectivity, ftp, http, https (with lynx), telnet, ssh, nntp, disk space, UNIX process existence, etc. The ease with which Big Brother can be externally extended makes adding "internal" services possible without too much trouble.

CONFIGURATION COMPLEXITY AND FLEXIBILITY

I had a certain amount of difficulty with Big Brother's configuration mechanisms — things sometimes seemed nonintuitive, and there seem to be more places to configure things than I would have expected.

My first surprise was with BB's `etc/bb-hosts` file — it uses # (a.k.a. hash, pound, octothorpe) both to indicate comment lines and to act as a separator on configuration lines. The `bb-hosts` file lists the hosts to be probed and specifies certain output formatting information for display. Some typical lines might look like this:

```
192.168.1.12 hostname          # ftp smtp telnet ssh
192.168.1.27 monitorhost       # BBNET telnet ssh ftp
192.168.1.39 webhost           # BBDISPLAY http://webhost/ telnet smtp
192.168.1.49 pagerhost         # BBPAGER telnet ssh  group

<h3>Workstations</h3>
192.168.1.51 work1             # telnet ssh
192.168.1.52 work2             # telnet ssh
192.168.1.53 work3             # telnet ssh
```

The documentation states that the bb-hosts file is "identical to the standard `/etc/hosts` file," with a few additional directives — I suspect that that is the original reason for using # as a separator and not just as a comment indicator. But with the addition of the various formatting and summary commands, this no longer seems to be a reasonable choice.

Additionally, the documentation warns against simply commenting out entries in some configuration files — I think that this is because some of the Big Brother programs search (with `grep`) for keywords in the file, blindly ignoring any comment indicators that might exist. There is no obvious way to continue lines in the `bb-hosts` file, which means that the definitions of some hosts can get a little ugly. Finally, the services for Big Brother to check are all specified as service names from the services file (ftp, telnet, etc.), except for http and https, which require a complete URL. This is an inconsistency I found somewhat problematic.

I had trouble understanding what all of the BB config files are for, and I kept finding out about new and different config files. A somewhat complete list of config files is:

```
etc/bbdef.sh
        various shared settings
etc/bbsys.local
        OS specific paths for commands and logs files
etc/bbwarnsetup.cfg
        paging and notification setup
etc/bbwarnrules.cfg
        who to notify, how, for what, and when
etc/security
        which hosts/networks can connect to the Big Brother daemon;
        no comments are allowed
etc/bb-proctab
        local override for the PROCS and PAGEPROCS variables from bbdef.sh
etc/bb-dftab
        local or shared disk warning configuration, overriding DFWARN
        and DFPANIC from bbdef.sh
www/notes/hostname.html
        additional per-host notes that can be accessed from the generated Web
        pages
```

Each of the various files seems to have a somewhat different syntax and rules about comments. Tools such as bbchkcfg.sh and bbchkhosts.sh are provided to check for certain configuration errors, but they don't notice some of the possible errors in the files and are somewhat fragile in their operation. Some of the files in the etc directory are intended to be shared, some are local, and some can be shared if desired.

EXCEPTION REPORTING STYLE

The Big Brother Web displays always provide a "full" status display — every device is always listed, and there's no way to have it display just those devices that have a nonnormal status.

This can be modified somewhat by the use of "summary" lines (which aggregate the information from another BBDISPLAY host), but that adds some level of "convolution" to your configuration, and may end up being harder to manage.

BB provides a simple mechanism to "acknowledge" a particular problem report, which suppresses additional trouble notifications for however long the user requests. A similar mechanism allows an administrator to disable notifications on a group of hosts for a specified number of minutes.

EXCEPTION-REPORTING TOOLS

Exceptions identified by Big Brother are reported in three standard ways: Web interface, email, and pager (via kermit, qpage, or sendpage). A mechanism allows the notification destinations to be set based on problem host and time of day. There's not really a useful command-line or curses interface — the lynx text browser is "usable" with BB, but painfully.

Overall the notification mechanism is more advanced and more flexible than I was expecting — it does, however, seem to require a certain amount of thought and staring at the documentation to get things set just right.

LOGGING AND DATA STORAGE

Big Brother logging, history, and status information is stored in a number of text files, most of which are separated into different files or directories by hostname and service type. BB typically logs only state changes, rather than every contact, so the logs tend to grow in a reasonable fashion. There doesn't seem to be a mechanism for pruning old information, but a simple script with a few `finds`, `rms`, and `mvs` can probably take care of most situations.

One thing that seems a little discomfiting to me: the files in BB's logs directory have modification times that are some number of minutes in the future. I think this is done in order to keep track of when state changes or renotifications should happen, but manipulating modification times makes me wonder if there isn't some better way achieve the same result.

REPORTING MECHANISMS

Big Brother's primary strength is near-realtime monitoring and reporting — some historical reporting is available, but it's somewhat limited. Trend information (such as disk-space growth over time, or load averages) is not available through Big Brother.

Other Notes

Overall, I thought that the Big Brother documentation could benefit from a little consolidation and elaboration. For example, when trying to determine what all the parameters on the `bb-hosts` file "summary" lines are for, I had to examine the source code before I really understood.

If you remove a host or service from a `bb-hosts` file line, you are required to manually remove the various log and history files related to those hosts or services, because otherwise you get various trouble indications and alerts for the now-removed service. This means that temporarily disabling the polling of a host may force you to remove history information you might otherwise wish to keep.

Similarly, if you've got various messages in your syslog files that Big Brother is warning you about, there's no way to tell Big Brother to stop telling you about particular problems unless you modify or remove the log files themselves or temporarily disable all warnings for those files, neither of which is a particularly appealing choice.

I happened to find the default Web display a little busy for my taste, but Big Brother provides ample opportunity to change and customize the displays, so I can easily make them just as plain as I want them to be.

One other thing I found surprising: I initially installed Big Brother on a machine that didn't have the lynx text browser installed. Big Brother didn't notice or report that lynx was missing; it indicated instead that the http service had failed. It took me a few tries to figure out just what was going wrong.

Opinion

In the world of monitoring software, Big Brother is fairly well known and, by all appearances, is widely used in small- to medium-sized installations. Lots of people find it very useful, and it certainly seems to fill a niche.

But even so, I've ended up in a sort of "like-hate" relationship with it. I wanted to like it and expected that I'd be able to recommend it enthusiastically for smaller sites. But I kept running into things that seemed overly complicated or more annoying than they had to be. Perhaps I was lulled into a false sense of "very simple" just because Big Brother has a Web interface. But as time went on, I kept seeing other little nagging things about it, and I kept finding out about yet another configuration file or option.

I was disappointed, but not surprised, by the lack of use of SNMP in Big Brother. To be honest, if I had started to implement a monitoring system in 1996, there's a pretty good chance that I wouldn't have used SNMP either. But with the advent of tools such as MRTG and Cricket, it's almost a necessity to have an SNMP agent running on every device and system. It seems a bit of a shame to have a separate client and network protocol and not make use of the SNMP agent that you probably want to have running anyway.

I think my conclusion is that Big Brother would benefit from a substantial rewrite, perhaps in some other more effective language, and a general consolidation and reconsideration of the configuration structure and syntax.

As the heading above indicates, this is just my opinion, and I suspect that large numbers of people will disagree with me and think I'm being overly critical. Some of my dissatisfaction with Big Brother is likely simply due to laziness or impatience on my part, or perhaps because I came to Big Brother with unreasonable expectations.

Clearly, Big Brother is effective for a lot of sites and fills a need. And, honestly, what more can one ask from a piece of software, or the kindness of software authors who make their considerable efforts available for free to the community at large?

The Big Brother home page, documentation, samples, and distribution are all available athttp://www.bb4.com/, along with information on support and commercial licensing, courtesy of the MacLawren Group Inc. The two presentations from SANS '98 http://www.bb4.com/bbsans98.pdf and SANS '99 (http://www.bb4.com/bbsans99.pdf) are worth a read. Spong "is a simple systems and network monitoring package" by Stephen L. Johnson, written in Perl, that is similar in some ways to Big Brother. See http://spong.sourceforge.net/.

system administration research

Part 3: Challenges to System Administrators

by Mark Burgess
<*Mark.Burgess@iu.hioslo.no*>

Mark is an associate professor at Oslo College and is the author of cfengine and winner of the best paper award at LISA 1998.

The preceding two articles in this series tiptoed around the discipline of objective knowledge. How can system administration be kept within the boundaries of science and protected from the specter of personal opinion? The answer was straightforward: the scientific method, i.e., a mixture of inspired inquiry and exacting self-criticism. To round up the series, I would like to present a number of challenges to the system administration community, challenges that indicate a few of the important problem areas we have to examine in the coming years. This article will be of a more speculative nature than the previous pieces, with the aim of stimulating activity in the field; the list will not be complete, but you can fill in the blanks yourself. Speculation provides the creative impetus for a rational inquiry. System administration touches on a wide realm of topics; it is loaded with technical and social issues; it requires creativity and analysis; and we know very little about the empirical (verifiable, factual) basis for the subject. In short, it is a jungle waiting to be explored.

Theoretical

Perhaps the greatest challenge for system administration is the development of theoretical models. For some, "theory" is a dirty word, signifying the ivory towers of irrelevancy — charmless academics who have never gotten their hands dirty, preaching to the workers from their soapboxes. But this popular view is unfair. No field of study exists without a theoretical framework, a frame of reference — and you can be sure that no one gets into system administration without getting their hands a little dirty. The challenge of theory is to come up with ideas that can be tested by the kind of rational inquiry I have been discussing in this series. If one can build models, they can be tested by simulations or by experiment. Indeed, without a framework for inquiry, experiments are often meaningless.

As I explained in Part 2, there is no sense in collecting data and trying to guess what they might mean. You have to start with a question and then see what the data tell you about the question. Analyzing data blind is rather like Schiaparelli looking at Mars through a blurry telescope and seeing *canali*, which became even more blurrily interpreted as The Canals of Mars. It is easy to see things that are not there, if you don't have some theory to interpret what you see. I have repeatedly emphasized that theoretical work for system administration is about building models that link cause and effect. In some cases that is easy, in others it is complex. Computers are *dynamic systems*. One of the first tasks then is to develop models of competition for resources.[1]

Take, for example, the immunity model that I presented at LISA 1998.[2] This is based on the idea that a computer system has an ideal state. Recently I showed how such a state can be defined from a given system policy by mapping the average condition of a computer system onto a "state space," or lattice of possible states, where the origin was defined to be the ideal.[3] As faults build up in the system, the actual state gets farther from the ideal, moving through the lattice. This is now a precise model, not just words.

There is more to do. I did not show that the definition was the only possible definition. In order to detect anomalous or harmful behavior, we need to know how the system migrates through the space of states. Can certain regions of state space be classified into OK and dangerous? Is there a continuous blend, or are there rigid boundaries? The answers to these questions would be of enormous importance to anomaly detection, and thus also security. Is threshold behavior the answer, or something more subtle? Can immunological models be based on co-stimulation: a kind of lock-and-key signaling that switches on immune responses in the body only for a very specific signature (two or more complementary keys are required for confirmation)?

Empirical

Experiment and theory are symbiotes, feeding off each other in a constant dialog. Just looking for trends in data can lead to canali. To test theories one needs to collect data and see how well those data support or contradict the theories. The data feed back into models and refine them. An important issue here is scale — finding the right scale for the right problem. As in the examples from Part 2 in this series, the first step in analyzing a problem is to determine the time-scale at which it occurs. Problems therefore need to be classified by scale. There is no sense in measuring variations once a week if you are trying to see how users behave, since the time-scale relevant to users is minutes, not weeks.

It sometimes happens that problems arise interpreting data. If the measured data are troublesome, measure something different. For instance, many researchers have measured process lifetimes, or the lifetimes of network events. These fall into divergent distributions, since some processes or events hang or never die. This causes a problem. Solution? Try looking at a different variable, for example, the number of events, a quantity that cannot easily diverge.

Experimentation is already tied to many issues such as anomaly detection or time-series analysis. Use of neural networks is common, but neural networks are not well understood. Try to keep within the bounds of what you understand. Studies of reliability, effectiveness of certain policies, determination of what the relevant quantities are to measure in specific instances, dependency in causal webs: the possibilities are almost endless.

Psychological

The behavior of users is central to the behavior of computer systems. Users are as much a part of the system as are the programs they manipulate. Can user behavior be understood in any convenient way? There is almost certainly a relevant literature on this subject in connection with user-interface design and other issues. It needs to be traced and examined. It will not be completely relevant to understanding how users interact in network communities, but it will pave the way and inspire new work.

What shall we measure? Can user behavior be classified into groupings with particular characteristics that are useful when defining system policy? Psychological behavior can be useful in several circumstances, such as predicting user behavior in a crisis or deciding on adaptive security measures. Do users fit profiles? Is it possible, and, if so, useful to put users into categories? Should the categories be different for different system policy models? What are the ethical issues here? Can profiles be kept and stored in a form that is meaningful only to the computer system and which cannot be examined or used by humans for scurrilous purposes?

Conflicts of interest develop in any community where limited resources have to be shared by several players. How do such conflicts develop? What are ways of extinguishing them before they escalate? What is the effect of trust and privilege between users and the system? Do increased trust and privilege lead to fewer conflicts, or do they simply open the system to abuse? I do not know whether studies of this kind have been performed in other fields of research. Certainly the results have not been applied and made available in the field of system administration. Note how this final point is related to the issue of "security through obscurity" — if one attempts to conceal something, it is like stamping that item TOP SECRET in flashing neon lights.

Technological

Research into smart/safe algorithms that implement aspects of theoretical models will lead to an improvement in basic technology. For example, I have personally worked on the idea of "convergence to the ideal state," using cfengine as an embodiment of the immunity model. Load balancing or resource sharing can be implemented algorithmically. Network middleware already creates technological layers that help to standardize these issues for network services. Can similar middleware technologies (like cfengine) assist with the core problems of system administration?

Many problems in system administration would be solved more readily if there were standard log-file formats or mechanisms that allowed programs to share log data without having to reparse the log files. Today, programs that want to know about the log output of other programs (e.g., intrusion-detection programs) have to parse each other's log files. This is a waste of effort, made necessary by inadequate infrastructure. A signaling nexus for trusted processes to share their state information would assist in the task of anomaly detection and in dealing with security issues.

Changes in operating-system technology can have a huge effect on the ease with which system administration is conducted. Often system administration is a workaround for operating-system problems: fixing symptoms rather than cause. Operating-system technology at present pays little attention to the issues of system administration. If anything, UNIX-like operating systems are taking steps backwards in trying to emulate Windows (as if that were the standard for operating-system design), giving users privileges they should not have, in order to make things "easy." The careful analysis of operating-system technology in relation to system administration is long overdue.

How will encryption affect the issues of system administration? If system processes cannot detect threatening programs or data files because they are encrypted, they cannot protect against them. For instance, if system policy dictates that huge MP3 files should be deleted after 24 hours, but a user conceals these by encryption, the policy cannot be policed. This is reminiscent of the attitude of governments toward wiretapping. It is not as straightforward an ethical or technical issue as some would like it to be. Rather, it represents a conflict of interests.

Load Analysis

This is the area where the most work has already been done. Internet traffic, process lifetimes, capacity planning. High-performance and realtime computing issues for specialized problems. If you are going to analyze load, ask yourself why. What are you going to do with the data? What are you trying to show?

Databases

Bioinformatics is the new discipline in biology related to the creation and interpretation of databases of genetic and protein data. This a crucial sharing of important empirical data about the mechanisms of developmental biology, virus signatures, and disease markers. Creating and maintaining important public databases is equally important in the computing community (e.g., DNS, RBL, virus signatures). What new databases do we need? How will such databases be managed? What about trust and security issues relating to the database contents?

The very first database I would like to see created is a database of references to system-administration research. Without such a library of what has been done, we will be doomed to repeat earlier work. Such a database would be a valuable resource to the field. At Oslo, we have plans to set up such a database in BibTeX format.

The Shape of Windows to Come?

The day-to-day tasks of system administration change constantly, and we pay these changes little attention. However, improvements in technology always lead to changing work practices, as humans are replaced by machinery in those jobs that are menial and repetitive. The core principles of system administration will remain the same, but the job description of the system manager will be rather different. In many ways, the day-to-day business of system administration consists of just a few recipes that slowly evolve over time.

It is difficult to articulate just why the administration of computer communities is an exciting challenge, but if we are to succeed in pushing through programs of research that will bring about the level of automation we require, then it will be necessary to attract willing researchers. Fortunately, today a high proportion of system administrators have scientific backgrounds and the will and training to undertake such work. However, only the surface has been scratched. The tendency has been to produce tools rather than to investigate concepts. While the tools are necessary, they must not become an end in themselves. A clearer understanding of the problems we face, looking forward, will be achieved only with more analytical work.

In many ways system administration is like biology. Animals are machines, just billions of times more complex than our own creations, but the gap is closing and will continue to close as we enter into an era of quantum and biological computing techniques. The essence of experimental observation and of the complex phenomena and interrelationships between hosts is directly analogous to what one does in biology. We may have created computers, but that does not mean that we understand them implicitly. In our field, we are still watching the animals do their thing, trying to learn.

ACKNOWLEDGMENT

Parts of this article have been adapted from the author's new book, *Principles of Network and System Administration* (J. Wiley & Sons), by kind permission of the publishers.

REFERENCES

To keep this list short, I have given only a few pointers. For a full list of references, please look to the sources below. See especially<*http://www.iu.hioslo.no/SystemAdmin/scisa.html*>. At this location you will find many references and pointers. I would like to build a full list of key references to central issues. You can contribute to this database by sending references to work you feel to be important.

1. N. Glance, T. Hogg, and B.A. Huberman, "Computational ecosystems in a changing environment." *International Journal of Modern Physics*, C2:735, 1991. M. Burgess, "On the theory of system administration," submitted to*Journal of the ACM*.

2. M. Burgess, "Computer Immunology," LISA 1998.

3. M. Burgess, *Principles of Network and System Administration* (Chichester: J. Wiley & Sons, 2000).

toolman

Meet Ed Q.

by Daniel E. Singer
<*des@cs.duke.edu*>

Dan has been doing a mix of programming and system administration since 1983. He is currently a system administrator in the Duke University Department of Computer Science in Durham, North Carolina.

Modifying UNIX disk quotas can be a hassle. The command interface provided by major UNIX platforms[1] is quite cumbersome. Perhaps this issue will be addressed (fixed) one day, with appropriate command-line options or some other mechanism being added to the API.

In this article I'll discuss a tool that helps to work around this unnecessarily annoying system-administration duty. Some basic knowledge of the standard UNIX disk-quota system is assumed.

Background

In the course of system-administration activities, it's not uncommon to have to create or maintain user disk quotas as a component of user-account and disk-space management. It's nice to have a quick and efficient means for making these routine updates. And if you use any type of script or program for automating the setup of accounts, it's also nice to be able to automate quota setup as part of the process.

The facility provided by most UNIX platforms for editing user disk quotas in the standard file system is the `edquota` command. `edquota` allows you to create or modify user quotas one at a time, either via an interactive text-editor interface or via a prototype. The text-editor method allows flexibility, but it is time-consuming and not readily adaptable to automation. The prototype method can be automated (i.e., used noninteractively), but it requires you to have preexisting prototypes that can only be copied exactly; various problems are inherent in this. See `man edquota` for more details.

Moving Forward

The shortcomings of `edquota` make an alternative method seem desirable. One possibility is to write a C program using quota-related system calls. This is not a bad choice, but it involves the usual hassles of maintaining a compiled program, and possibly needing distinct binaries for each of multiple UNIX platforms. For us script-heads, a script approach is much more attractive.

Several years ago, a coworker mentioned having seen a USENET posting[2] that demonstrated a clever use of the EDITOR environment variable with `edquota`, and he suggested that this mechanism might be exploited to impose some automation on the quota-editing process. After some consideration, this approach did appear to me to be workable. You see, `edquota` invokes the editor of your choice to modify a quota, this *usually* being an interactive text editor.

So the basic plan followed is this: a Bourne shell script processes command-line arguments (user, partition, quota, options, etc.), sets the EDITOR environment variable to some program, then runs `edquota`. The program that EDITOR points to automatically (noninteractively) does the quota editing.[3] In this implementation, the script and the

EDITOR program are actually the same script, just called by different names (via a hard or symbolic link). So, `edq`, our script, calls `edquota`, which then calls `edq` (by a different, linked name) to do the quota editing.[4]

Execution

A typical use of `edq` (pronounced "ed cue") looks something like this:

```
# edq tina,,100000
```

This invocation sets the home directory blocks soft limit for the user account "tina" to 100,000 KB, the blocks hard limit to 125,000 KB, the inodes soft limit to 20,000, and the inodes hard limit to 25,000. Apparently some defaults were involved in this first example.

Here's a more explicit (and perhaps more general) example:

```
# edq adam,/work1,50000,10000 betti,/work2,80000,20000
```

Here we've set the quotas for two accounts at once, and we've also specified the partitions and soft limits for blocks and inodes explicitly. (The hard limits are still being set by hard-coded default ratios; see below.)

To summarize a bit, `edq` can noninteractively set one or more disk quotas for one or more users on one or more file systems, optionally employing various defaults. If the partition is omitted, the home directory is assumed. If the inode limit is omitted, it is computed from a ratio to the blocks limit. Hard limits are based on soft limits, and are also computed from ratios to the soft limits. Defaults for all of these ratios can easily be set in the script, but can all be overridden with environment variables:

- INODES_SOFT_RATIO: ratio of inodes to number of blocks
- BLOCKS_HARD_RATIO: ratio of blocks hard limit to soft limit
- INODES_HARD_RATIO: ratio of inodes hard limit to soft limit

(Run `edq -h` to see the defaults.)

Remote Control

What if the disk partition in question is on another host? Not to worry! Either the partition can be specified with a *host:/filesystem* notation, or the -r (remote) flag can be used. So, let's say you have a script that sets up accounts. The home directory quota could be set with a line similar to this:

```
INODES_HARD_RATIO=1.2 INODES_SOFT_RATIO=.3 \
BLOCKS_HARD_RATIO=1.2 edq -r $user,,$blocks
```

This sets the blocks soft limit for `$user` to `$blocks`, sets the inodes soft limit to `($blocks * .3)`, sets the blocks hard limit to `($blocks * 1.2)`, and sets the inodes hard limit to `($blocks * .3 * 1.2)`, assuming appropriate permissions to the possibly remote host housing the home directory partition for `$user`. (`rsh` could also be used for the remote access, but the remote host name would have to be provided; `-r` will figure out the right host.) So besides the user, the only other piece of hard information needed is the blocks soft limit.

Why this ratio approach? I don't want to have to figure out or maintain a table of all of the different values for each quota I might use for different kinds of accounts or for different partitions. So I just use ratios from what seems to me to be the most obvious figure, the blocks soft quota. For my purposes, having the inodes soft limit be 20% of the blocks limit, and then having the hard limits be 25% greater than the soft limits, seems about right, and these figures are hard-coded into edq.

Admittedly, the environment variable, ratios, etc., might seem a bit klutzy. But I've tried to keep the option list simple and allow for multiple quotas to be set with one call (the latter admittedly being a feature I don't use very often). Since it is a script, you can easily edit it and reset the default ratios to your preferences, and then never have

to worry about it again. I have it set to my preferences, and I never seem to have any need to deviate. Plus, it is designed so that use of the defaults can usually be leveraged, as in the very first example above. The use of commas for a compound argument delimiter also helps make it easier to omit components so as to go with defaults.

Syntax Variations

As I've mentioned, allowing for setting multiple quotas on the command line is perhaps dubious. I've considered whether it would make more sense just to allow for multiple account names or partitions, with the other data being specified only once. But this is not something one would need very often, except possibly for multiple account names if you are setting up multiple user accounts simultaneously, or when applying quotas to a new partition for the first time. But, then, I figure why bother, since these can be accomplished from the command line using the macro expansions supplied with many modern shells. For example, with the `csh`derivatives:

```
# edq -r {amy,barney,cathie},,60000
```

would automatically expand out to:

```
# edq -r amy,,60000 barney,,60000 cathie,,60000
```

And so it is already taken care of.

Other Languages

As I mentioned earlier, I'd prefer not to use a compiled language for this tool. Other scripting languages could probably accomplish this task with less code and more efficiency. For example, here's some Perl code provided by Seth Vidal of the Duke University physics department:

```
#!/usr/bin/perl
use Quota;
if (scalar(@ARGV) < 6) {
        print "\nUsage: setquota.pl mountpoint username blocksoft blockhard
filesoft filehard\n\n";
        exit(1);
}
my ($dir,$user,$bs,$bh,$is,$ih) = @ARGV;
$dev=Quota::getqcarg("$dir");
my @userstuff=getpwnam("$user");
my $uid=$userstuff[2];
Quota::setqlim($dev,$uid,$bs,$bh,$is,$ih,1);
```

In comparison, version 1.7 of `edq` is 837 lines long. But then, `edq` does contain some comments and a few more features. I chose Bourne shell because I like it, it's fairly portable, it's ubiquitous, and, well, because I wanted to try out that EDITOR environment variable trick suggested by that USENET posting. And, hell, it works!

Summary

The ability to easily update user disk quotas noninteractively has been overlooked for far too long by UNIX vendors and standards bodies. `edq` is one attempt to address this problem, providing a simple command-line interface for setting disk quotas both locally and remotely, with various details being rationally handled by practical defaults.

`edq` has been tested on fairly modern versions of Solaris and Digital UNIX, and also on SunOS. Some modifications will be required for other UNIXes. Drop me a line if you need help with a port, or if you want to provide one.

The `edq` Bourne shell script and its man page can be found at either of these addresses:
<*http://www.cs.duke.edu/toolman/*>
<*ftp://ftp.cs.duke.edu/pub/des/scripts/*>

NOTES

1. AIX, BSDs, D.U., HP-UX, Irix, Linux, Solaris, and SunOS, to name a few.

2. The posting in question — by the venerable Casper Dik in <*comp.sys.sun.admin*>, dated 14 Feb 1996 10:14:46 GMT — actually presented a method of adding a SUID wrapper to`edquota`; this particular issue is not addressed in this article.

3. The way this actually works is that `edquota` dumps some data into a temporary file, and then calls `$EDITOR` with the temporary file pathname as an argument; for example, vi /tmp/EdP.aXyyxV. After `$EDITOR` exits, `edquota` reads the temporary file back in and continues from there.

4. This is similar to the approach for using the EDITOR environment variable and the rc script as a revision-control wrapper around some other process. See the "Sleight of Hand" section in "Revision Control Revisited" in the October 1999 issue of *;login:*.

pithy sayings for the UNIX sysadmin

by William S. Annis
<*annis@biostat.wisc.edu*>

One day in late December I was thinking about the issue of sysadmin education. I had just hired a new student, and my previous two had just graduated. So with the new student I'd lose the normal tendency of students to instruct one another to the degree they are able. Of course, teaching the student myself would be time well spent, but rarely is enough time available.

There is, of course, no royal road to learning each site's peculiarities, any more than there is to learning the options to find, but I started to wonder if there was some way to distill bits of general UNIX admin wisdom into small, memorable phrases. Part of my inspiration for this is the many collections of Go proverbs that help beginners remember basic guidelines of good play. Unfortunately, the reasoning behind proverbs like "the monkey jump is worth eight points" isn't necessarily clear to the novice Go player, any more than "`cat /dev/null` into a large file to free space" is clear to the beginning UNIX admin. So, any collection of memorably pithy aphorisms was going to require some commentary.

After writing up and commenting a few basic aphorisms I already knew needed to be on the list, I sent email to other sysadmins and to two mailing lists with high sysadmin content, asking for more aphorisms. This caused a number of interesting discussions, and even a few new aphorisms. Interestingly, many of the aphorisms have more to do with social issues that technical ones, so "'s' is for sticky; 't' is for `/tmp`" sits geekily next to the more sober "Does the user think it's fixed?" Most of the aphorisms have turned out to be useful reminders for experienced admins.

A good aphorism should be both memorable and short; alliteration doesn't hurt, either. Many of the aphorisms in the collection are not exactly perfect examples of the genre. I'm a sysadmin, not a poet! But I hope the collective folk wisdom of the *;login:* readership can improve them.

Find the aphorisms at <*http://www.biostat.wisc.edu/~annis/aphorisms.html*>. Here are four from the middle of the list:

- Don't rewrite `cat -n`
- What have you changed? Whom did you tell?
- What was the return code?
- Where does your symbolic link point today?

Each aphorism comes with commentary. Here's the commentary for the second aphorism above:

- It is vitally important that the sysadmins you work with know what it is you're doing to the system. This goes beyond merely using your version-control system dutifully on any system files. It means telling your co-workers when you do things. This is especially vital when the admin staff is spread throughout a building, city, or the world and direct interaction isn't constant.
- Where I work we have an email alias to which admins are expected to send reports on system things they change that have any small chance of breaking things. This alias sends mail to all the admins, and further is logged into a file — one per year — in a directory devoted to these. Now, this logging may be objected to as being redundant if you're already using version control, but this alias gives the admin a chance to go into background that might be omitted in the change log. It's also a lot easier to grep for changes this way, not surprisingly.

New aphorisms and comments are welcome and encouraged.

[Editor's Note: These really are great! Check them out and share a few of your own. -RK*]*

taming the wild West

Laws and Regulations Governing the Technology Industry

by John Nicholson
<*John.Nicholson@ShawPittman.com*>

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, and other technology issues.

Back in the Old West, people lived in small enclaves of civilization that provided them with some protection from the dangers of the wild.[1] Outside of those small outposts, you took your chances. Inside them, people were protected by a few laws and the fact that everyone knew everyone else. Many laws and regulations that were commonly enforced in larger Eastern cities were unnecessary in these small communities. Instead, peer pressure enforced a general code of conduct and standard of behavior.

As news of the freedom and opportunity of the West spread, and as it became safer and easier to travel, more and more people moved to the West. As the West became more densely populated, more people interacted more frequently and more anonymously. Many of these interactions involved people who were transitory and/or unfamiliar with (or unwilling to comply with) the general code of conduct observed by those who already lived there. This caused increasing conflict. To avoid this increasing conflict, new laws were passed and old laws were better enforced. Those who lived in the once-small towns bemoaned the invasion of the horde of strangers and the loss of their "small-town" way of life.

Similarly, when the technology industry and the Internet were new, very little government involvement was required. The community was small enough and homogeneous enough that codes of conduct and unwritten rules governed behavior, enforced by peer pressure and public opinion within the community. Even as recently as the early '90s, one of the big rules was that the Net was not for business. How things have changed.

Just like the Old West, the technology industry and the Internet are victims of their own success. As people have realized the opportunities presented by the Net, and as technology has made getting access easier, more and more people have flooded into the "small towns" that used to populate the Net. AOL alone has millions of subscribers. Email once the realm of academics, researchers, and programmers is now a mission-critical application. The Web is now a major marketing and business tool. Until recently, there has been very little need for new legislation in this area, or even enforcement of many existing laws that apply outside of the cyber-realm. The increasing online population, however, combined with the anonymity and the relative lack of knowledge of many of these new users, has led to increasing conflicts.

Just as new laws were passed and old laws were enforced in the Old West, the same thing is happening to the technology industry. As a part of this trend, there have been several new developments that technology professionals should know about. This and my next few columns will deal with the Children's Online Privacy Protection Act (COPPA), the Uniform Commercial Information Transactions Act (UCITA), and the Digital Millennium Copyright Act (DMCA). Depending on what your company (and you) do, one or all of these new laws could have an impact on you.

The Children's Online Privacy Protection Act

The Children's Online Privacy Protection Act[2] was enacted by Congress to protect children's privacy by giving parents the ability to control what information is collected from their children online. COPPA directed the Federal Trade Commission (FTC) to draft rules that would control how Web sites gather, store, and disclose such

information. The rules enacted by the FTC[3] became effective April 21, 2000, and apply to the online collection after April 21, 2000, of "personal information" from children under 13. The new rules spell out what a Web site or online service "operator" must include in a "privacy statement," when and how to seek "verifiable consent" from a parent, and what responsibilities an operator has to protect children's privacy and safety online.

WHY DO WE NEED SOMETHING LIKE COPPA?

Aside from the obvious safety issues and the anecdotal evidence regarding stalkers and pedophiles on the Net, there is an economic reason for this type of regulation. Infor-mation has become a valuable commodity. And, as with every other valuable commodity, those who have it are trying to protect it, and those who want it are trying to come up with new ways to acquire it. One of the reasons for something like COPPA is to prevent companies from exploiting children's ignorance of the value of personal information from both a safety and an economic perspective.

For example, according to a recent survey by the Annenburg Public Policy Center,[4] two-thirds of children aged 10 to 17 were willing to provide online the names of their family's favorite stores in exchange for a free gift. In addition,

- 54% were willing to provide the names of their parents' favorite stores;
- 44% were willing to disclose the type of car the family uses;
- 39% were willing to discuss the amount of their allowance and whether their parents talk "a lot" about politics;
- 26% were willing to disclose details about their parents' activities on the weekend; and
- 16% of 10—12-year-olds admitted to having given information about themselves to a Web site.

On top of these statistics, the survey also revealed that 46% of parents were not aware that Web sites could gather information about a user without the user's knowledge.

Given the willingness of children to disclose personal information about themselves and their families, as well as parents' ignorance of the methods used by Web sites to collect information, a regulation like COPPA is necessary to protect children and parents from their own ignorance.

WHY SHOULD YOU CARE ABOUT COPPA?

If COPPA applies to your Web site or online service and you do not comply with the regulations, the "operator" (see below) of the Web site or online service can be fined $10,000 per violation.[5]

WHO HAS TO COMPLY WITH COPPA?

The COPPA regulations apply to the following:

1. The "operator" of a commercial Web site or online service "directed to children under 13" that collects "personal information" from children or
2. The "operator" of a general-audience Web site who has "actual knowledge" that it is collecting "personal information" from children under 13.[6]

As with any law or regulation, the definitions are very important.

Who is an "operator"? To determine whether an entity is an "operator" with respect to information collected at a particular Web site, the FTC will consider such factors as:

- Who owns and controls the information once it is collected?
- Who pays for the collection and maintenance of the information?
- What preexisting contractual relationships exist in connection with the information?
- What role does the Web site play in collecting or maintaining the information?[7]

How do you determine whether a Web site is "directed to children"? A site will be evaluated based on:

- the subject matter;
- specific audio/video content;
- the age (or apparent age) of the models or other people pictured on the site;
- level of language used on the site;
- whether advertising on the site is targeted to children;
- the age of the actual or target audience; and
- whether the site uses "child-oriented features" such as animated characters.[8]

What is "actual knowledge"? If a site has a registration form that asks for a user's age and the user enters an age under 13, then the operator of the site must comply with the COPPA regulations with regard to information provided by that user.

What is "personal information"? According to the FTC, personal information is information about a child that is collected online, such as the child's name, home or other address including street name and name of a city or town, email address, telephone number, Social Security number, or any other information that would allow someone to contact the child — including information such as hobbies or interests or information collected passively via cookies or other tracking technology if such information is linked to other "individually identifiable" information about that child.[9]

Complying with COPPA

If the COPPA regulations apply to your Web site or online service, there are a number of things that you have to do to comply with them. The first requirement is that you must develop a "privacy statement" that describes the information practices of your site or service. Once you have developed your privacy statement, if you are the operator of a site directed to children, you have to provide a link to the privacy statement on the home page of your site and at each area where your site collects personal information from children. If you are an operator of a general-audience site with a separate area directed to children, you have to provide a link to your privacy statement on the home page of the children's area and, although the regulations do not appear to require it, it would be a probably be a good idea to provide the link at each area where your site collects personal information from children. All of the links to the privacy statement are required to be "clear and prominent" (i.e., in a larger font or a different color). The FTC guidelines specifically state, "A link in small print at the bottom of the page — or a link that is indistinguishable from other links on your site — is not considered clear and prominent."[10]

What is a "privacy statement"? According to the FTC, a privacy statement should be clearly written and understandable and should not include any unrelated or confusing materials. A privacy statement must contain the following information:

- the name and contact information (address, telephone number, and email address) of all operators collecting or maintaining children's personal information through the Web site or online service.
- the kinds of personal information collected from children (e.g., name, address, email address, hobbies, etc.) and how the information is collected (i.e., directly from the child or through cookies or other means).
- how the operator uses the personal information collected by the site. For example, is it used for marketing back to the child? Notifying contest winners? Allowing the child to make the information publicly available through a chat room? (Note that if your site provides any kind of chat or posting functionality, you are providing a way for the child to make information publicly available).
- whether the operator discloses to third parties information collected from children. If so, the privacy statement must also list the kinds of businesses in which the third parties are engaged; the general purposes for which the third parties will use the information; and whether the third parties have agreed to maintain the confidentiality and security of the information.
- that the parent has the option to agree to the collection and use by the operator of the child's information without also consenting to the disclosure of the information to third parties. (Note that the effect of this option is that you must develop a system for tracking whether or not information collected about each child can be disclosed to third parties.)
- that the operator may not require, as a condition of participation, a child to disclose more information than is reasonably necessary to participate in an activity.
- that the parent can review the child's personal information, ask to have it deleted, and refuse to allow any further collection or use of the child's information. The notice also must state the procedures for the parent to follow.[11]

Parental Consent

Until 2002 (when the system will be reviewed), the COPPA regulations provide for two levels of parental consent, depending on what the operator intends to do with the information it gathers. If the operator of a Web site does not share with other companies or organizations any of the information it collects from children, then it can ask parents to grant consent to collect the information via an email, provided that the operator uses some type of follow-up (i.e., delayed email, fax, letter, or phone call) to increase the likelihood that the parent has actually consented. If the operator of the Web site intends to share the information it collects, it must set up a more verifiable system to obtain parental consent. This means that an operator must make reasonable efforts (taking into consideration available technology) to ensure that before personal information is collected from a child, a parent of the child receives notice of the operator's information practices and consents to those practices. Possible methods for obtaining parental consent include requiring the input and verification of a credit card number, setting up toll-free numbers where operators are trained to recognize the difference between the voice and language of a 12-year-old and those of an adult, providing a form for parents to fill out and fax or mail back to the operator, or email from the parent signed with a digital signature.

The notice to parents must contain the same information included on the privacy statement on the Web site. In addition, an operator must notify a parent that it wishes to collect personal information from the child; that the parent's consent is required for the collection, use, and disclosure of the information; and how the parent can provide consent. The notice to parents must be clear and understandable, and must not contain unrelated or confusing information. An operator of a Web site or online service may provide the notice to parents by sending an email message, fax, or a notice by mail to the parent. Given the amount of information to be conveyed to a parent, a telephone call is probably not the best way to provide the notice.

Exceptions

The COPPA regulations require an operator to give a parent the option to agree to the collection and use of the child's personal information without agreeing to the disclosure of the information to third parties. However, if a parent agrees to the collection and use of their child's personal information, the operator may release that information to others who use it solely to provide support for the internal operations of the Web site or service, including technical support and order fulfillment.[12]

In the case of a monitored chat room, if all individually identifiable information is stripped from postings before they are made public and the information is deleted from the operator's records, then an operator is not required to get prior parental consent.

The COPPA regulations also include several exceptions that allow operators to collect a child's email address without getting the parent's consent in advance. Prior parental consent is not required for an operator to collect:

- a child's or parent's email address to provide notice and seek parental consent;
- an email address to respond one time to a single request from a child, provided that the operator then deletes the child's email address;
- an email address to respond more than once to a specific request (e.g., for a subscription to a newsletter), provided that the operator notifies the parent that it is communicating regularly with the child and provides the parent with the opportunity to stop the communication before sending or delivering a second communication to a child. (Note that to take advantage of this exception an operator must develop a system that notifies the parent before sending a second communication to the child and cancels the second communication if the parent does not consent.)
- a child's name or online contact information to protect the safety of a child who is participating on the site, provided that the operator notifies the parent and gives the parent the opportunity to prevent further use of the information;
- a child's name or online contact information to protect the security or liability of the site or to respond to law enforcement, if necessary, provided that the operator does not use it for any other purpose.[13]

New Notice and Consent Required for Changed Practices

An operator is required to send a *new notice and request for consent* to parents if the operator materially changes the collection, use, or disclosure practices to which the parent had previously agreed. This means that each operator must develop a system for tracking *by child* the date that a parent provided consent, the uses of information to which

the parent consented, and a means of contacting the parent in the event that the use by the operator changes. For example, parental consent for a child to participate in contests that require the child to submit specific, limited personal information would not cover the addition by the Web site or online service of access for the child to chat rooms. Alternatively, parental consent for the operator to disclose collected information to marketers that provide one type of product might not extend to providing the information to a marketer of a different type of product. Unfortunately, in this case there is no clear line to identify when new consent is required.

Disclosure of Collected Information to Parents

Upon request, an operator must disclose the general kinds of personal information it collects online from children (e.g., name, address, telephone number, email address, interests, hobbies, etc.). In addition, upon verified request from a child's parent, an operator must disclose to the parent the specific information collected by the operator from the parent's children who have visited the operator's site(s) or used the operator's service(s). The FTC requires that an operator must use reasonable procedures to verify that the person requesting the child's information is, in fact, the child's parent before providing the information. According to the FTC, acceptable methods for verifying the parent's identity include:

- obtaining a signed form from the parent via mail or fax;
- accepting and verifying a credit-card number;
- taking calls from parents on a toll-free telephone number staffed by trained personnel (although the FTC does not describe the nature of the training to be provided to the personnel);
- email from the parent signed with a digital signature;
- email from the parent including a PIN or password obtained through one of the other verification methods listed above.[14]

The FTC provides protection from liability for inadvertent disclosures of a child's personal information to someone who claims to be a parent so long as the operator has used one of the above procedures and was acting in good faith to respond to a request for parental access to the information collected by the operator.

Parental Revocation of Consent and Requests for Deletion

Under the COPPA regulations, a parent may at any time revoke his or her consent, refuse to allow an operator to further use or collect his or her child's personal information, and direct the operator to delete the information.[15] If a parent does this, the operator is allowed to terminate any service provided to the child, but only if the information at issue is reasonably necessary for the child's participation in that activity. For example, an operator may require a child to provide his or her email address to participate in a chat room. If a parent later requests that the operator delete the child's information, the operator may refuse to allow the child to participate in the chat room. However, if other activities or services on the site do not require an email address, the operator must continue to allow the child access to those activities or services.

"Safe Harbors"

The COPPA regulations specify that industry groups or others can propose self-regulatory programs to govern participants' compliance with the COPPA regulations. Such self-regulatory programs must provide for independent monitoring and disciplinary procedures and must be approved by the FTC. If an operator complies with an approved self-regulatory program, that will generally protect the operator from fines for violations of the COPPA regulations.[16]

So far, the FTC has received two applications to create self-regulatory programs. The first was submitted by PrivacyBot.com[17] and the second by the Children's Advertising Review Unit (CARU) of the Council of Better Business Bureaus.[18] The FTC has received public comments for both proposals,[19] and both are currently under consideration.

What Are People Doing Already?

Rather than set up procedures to comply with the COPPA regulations, E-Crush.com[20] and Email.com (a service of NBC Internet)[21] have banned those under 13 from their sites. From now on, Disney will require a credit card to set up a "family account" for use by those under 13, although the cards will not be charged.[22] As of April 24, 2000,

SurfMonkey.com estimated that it has spent between $50,000 and $100,000 to comply with the COPPA regulations.[23]FreeZone, a portal for kids from 8 to 14 that already requires parental consent, estimates that it will spend about $100,000 to comply with the COPPA regulations.[24] Alloy.com, a site targeted to teens, reported that it will spend approximately $200,000 to comply.[25]

Conclusion

Although necessary, the COPPA regulations place a substantial burden on Web sites that intend to collect and use or distribute personal information about children under 13. Sites will either have to develop the required systems and procedures or they will have to stop providing service to those identified as being under 13. It remains to be seen whether any of the regulatory "safe harbors" will be approved (the FTC may have acted on them by the time this article goes to print; if so, I'll provide an update in the next issue) and whether they will provide any relief from the burdens imposed by the COPPA regulations.

In the next issue, I'll discuss UCITA and some of its implications.

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. 15 U.S.C. §§6501-6506.

3. 16 C.F.R. §312.

4. "Weak Link in Net Privacy: Kids," <*http://www.zdnet.com*>.

5. Brown, "COPPA: Locked, Loaded, Patrolling the Net," <*http://www.zdnet.com*>, 3/13/2000.

6. "How to Comply with the Children's Online Privacy Protection Rule,"<*http://www.ftc.gov/bcp/conline/pubs/buspubs/coppa.htm*>, Nov. 1999.

7. Ibid.

8. Ibid.

9. 16 CFR Section 312.2.

10. "How to Comply with the Children's Online Privacy Protection Rule," <*http://www.ftc.gov/bcp/conline/pubs/buspubs/coppa.htm*>, Nov. 1999.

11. Ibid.

12. Ibid.

13. Ibid.

14. Ibid.

15. Ibid.

16. 16 CFR Section 312.10.

17. <*http://www.ftc.gov/os/2000/02/privacybot.pdf*>.

18. <*http://www.ftc.gov/privacy/safeharbor/caruappmaterials.pdf*>.

19. <*http://www.ftc.gov/privacy.safeharbor/privacybotcomments/index.html*> and <*http://www.ftc.gov/privacy/safeharbor/65FR/24960*>.

20. Bowman, "Sites Brace for COPPA Fallout," <*http://www.zdnet.com*>, 4/20/2000.

21. Brown, "COPPA: Locked, Loaded, Patrolling the Net," <*http://www.zdnet.com*>, 3/13/2000.

22. Bowman, "Sites Brace for COPPA Fallout," <*http://www.zdnet.com*>, 4/20/2000.

23. Angwin, Wingfield, and Tran, "COPPA Cost Too High for Some Sites," <*http://www.zdnet.com*>, 4/24/2000.

24. Bowman, "Sites Brace for COPPA Fallout," <*http://www.zdnet.com*>, 4/20/2000.

25. Ibid.

to the limit

by Steve Johnson
<scj@transmeta.com>

Steve Johnson has been a technical manager on and off for nearly two dec-ades. At AT&T, he's best known for writing Yacc, Lint, and the Portable Compiler.

and Dusty White
<dustywhite@earthlink.net>

Dusty White works as a management consultant in Silicon Valley, where she acts as a trainer, coach, and trouble-shooter for technical companies.

A lot of our behavior stems from who we think we are. If we think we are athletic, we will quickly take up a new game or jump at a chance to ski a new ski area. If we think we are clumsy, we won't.

We actually tell ourselves a lot of stories — I'm not very good at learning languages, I'm shy, I can't speak in public, I can't carry a tune, I'm a (good/very good/lousy) C++ programmer. (Men/women) (like/don't like) me. I (am/am not) management material.

Every time we tell ourselves a story like this that defines us, it also limits us. By saying who we are, we are also saying who we believe we are not. In the face of rapid change in our industry, can we afford to unconsciously shut ourselves off from anything?

So what can we do to identify and relax these barriers? The first, and often most difficult, task is "simply" to become aware of our internal talk, the stories we tell ourselves. Whenever you say "no," ask yourself how you came to that conclusion. And remember that even the "yes" answer may have a hidden "no" in it as well, since most of us need to give up something in order to do something else. By observing ourselves and listening to the excuses that automatically pop out of our mouths, we get insight into the stories we have told about ourselves.

Often these stories were true at one time in our life. Equally often, the conditions that caused them to be true at that time are no longer in force. If you were shy when living at home with parents who pushed you to succeed, then do you still need to be shy as a parent yourself in your 40s? Or is it a habit you can break? But you can't break it without knowing it is there.

These "limiting decisions" become part of the problem, part of what keeps you acting shy, stops you from learning to sing, or interferes with whatever else you would like to do. The unconscious power of these decisions can be enormous. People who tell themselves that they are clumsy may need to trip over things to reinforce their self-image. People who feel unattractive may need to sabotage relationships. People who feel they are dumb may need to prove it from time to time on the job.

In many cases, becoming aware of the limiting decisions, becoming conscious of how you came to make this decision, and understanding that these earlier conditions no longer apply can be the first steps in changing yourself to become more effective on and off the job.

So suppose you identify yourself with a trait — suppose you tell yourself you are shy. Here are some things you can do to get more insight into this story you tell yourself.

First of all, turn the adjective into a process. You are not shy, you do the process of "shying." When do you know to start "shying"? Are there ever times you don't shy? How are those times different from the times you are shying? Look carefully at what triggers your shying — if it happened differently, would it still work to trigger you to shy? Do you do your shying in response to men, or to women, or to children? Do you do it at home but not at work? Do you shy when you are alone, or with just one other person? Can you remember times in your life when you didn't shy?

The next step is to ask yourself what is the intention of your shying. (Chunk it up, to use the terminology of an earlier article of ours.) What would happen if you didn't
shy in a situation where you usually do? What doesn't happen because you are shying away from it? What wouldn't happen if you didn't shy away from it? If you imagine at some time in the future being able to function without shying, how would things be different?

Perhaps you are shy because you want to avoid failing in public. Is shyness the best way to avoid failing in public? Aren't there other ways you might achieve this end that are more effective than shying? If you could be better at some of these other strategies, could you stop shying?

Finally, it is hard for most of us to see ourselves as clearly as a good friend or a trained professional person can. So working on limiting decisions is often best done with a friend, a coach, or a counselor. It is rewarding work both for you and for them.

keeping employees by keeping them happy

by Christopher M. Russo
<*chris@thlogic.com*>

Chris Russo manages engineering at Genuity. His focus continues to be satisfying the customer — whether that be a Web developer, a system administrator, or an end user.

Employee retention is a difficult challenge that faces most managers today. This is the first in a series of articles designed to help managers better understand the unique needs of employees, the measures necessary for keeping them happy, and the justification and reasoning for doing so.

Most organizations in our economy — especially technical ones — have an unbelievably high turnover rate. They simply cannot seem to hang on to good people for any length of time. For example, prior to my taking over as manager of my present group, turnover was somewhere on the order of 100% annually. While certainly that number is a bit extreme, attrition rates of upwards of 30% and even 40% are not at all uncommon, and this is an enormous and expensive problem.

Since taking over, I have brought the team's attrition rate down to about 6%. I'd say that's pretty good, especially since I work in the technology industry, and I have some extremely talented people on my staff — the kinds of folks who could pretty much quit one day and have a new job the next.

Because of our general success, I have had a number of people ask me just how it is that we have managed to maintain, and in fact increase, our staffing. The answer, at least on the surface, is a fairly simple one: I keep my employees happy.

"Wow, that's easy — thanks so much!" is usually the sarcastic reply. But it is the truth. Simply stated, employees leave only when they are unhappy, whether it's because they aren't being treated well, don't like the work, haven't got the appropriate resources and training to do their jobs, or simply don't like you, the manager. This doesn't necessarily have to be a big emotional thing — unhappiness can take the form of a simple lack of desire to work within whatever situation or environment that they are in. For example, perhaps they are looking for a career change or are concerned that the work location is too far from home and family in case of an emergency.

So the obvious question, then, is: how do I keep my employees so happy? There are a great many different ways one can do this, and it is my hope to illustrate those that I feel are most critical as well as frequently universal in any organization. (Needless to say, if you run an auto-repair shop, getting everyone something nice for his or her nonexistent cubicle might be a little pointless.)

Before we get into how to keep our employees happy, however, it is very important that we spend a few moments to talk about how expensive it is to find good people. I will speak of specific methods for keeping employees happy in my following two articles in this series.

The Expense of Hiring

The reason we need to illustrate the expense of hiring is that many managers have a very distorted view of how to keep their staff, and it usually does not involve spending money or doing things that are (in their minds) unorthodox or extraordinary.

If you have a manager like this, then you will probably need to read, understand, and possibly use this information to justify some of the methods I personally use to keep my staff.

Simply stated, hiring people is very expensive. As an example, let's look at my situation. When I joined my present company I came in with three permanent staff members and one contractor. I was told that I needed to staff up to about 20 people, and that my overall staff should have a greater concentration of permanent people than contractors.

In my particular department, our requirements are pretty tough, so I usually have to review about 30 resumes before getting one that warrants a phone screen. Once I have such a resume, I contact the recruiter and spend a little time establishing an appropriate point in my day when I can conduct the phone screen. Once that's established, the person calls me at my office, and I usually spend 30 to 40 minutes conducting the actual phone screen. After the phone screen is complete, I take about 10 to 20 minutes to compile my notes on the candidate and communicate them back to the recruiter. Usually I have to go through about four phone screens before I find a candidate who seems worth bringing in for a full interview.

When the candidate comes in, we conduct a series of interviews to ensure that the person is a good fit for us, and vice versa. This usually takes about six people at roughly 30 minutes each. Then, if the candidate manages to make it through all of the interviews unscathed, I will usually spend 30 to 60 minutes wrapping up the interview process at the end.

Once the candidate is gone, the entire team will meet to discuss the results of the interview. Depending on how controversial we determine it to be, it may be the entire team, but it's usually just those who conducted the actual interview. This usually takes 15 to 30 minutes. It usually takes two or three interviews before we get a candidate who successfully passed through the full interview process and was recommended by the team as a good potential hire.

If it is decided that an offer should be extended, I will usually spend about 30 minutes working with human resources determining what the offer should be, and they usually spend a total of an hour putting together the paperwork, getting things signed, and actually extending the offer to the candidate.

Now this entire process disregards the possibility that the team discussion about the candidate becomes deadlocked and a decision takes longer to make. It also glosses over the possibility of the candidate counteroffering or being somewhat unsure for a few days, completely rejecting the offer, or any other complication that is actually fairly common in this process. With that in mind, let's see how many hours we have sunk into getting the first accepting candidate in the door:

1. Review 30 resumes to get to one viable for a phone screen (300 minutes).
2. Conduct phone screen and document results (60 minutes).
3. Repeat steps 1 and 2 four times to get candidate viable for interview ((300 + 60) * 4 = 1440 minutes).
4. Conduct interview process on-site (240 minutes).
5. Discuss candidate viability with team (210 minutes).
6. Repeat steps 1, 2, 3, and 4 three times to find candidate worth extending an offer to ((1440 + 240 + 210) * 3 = 5760).
7. Human resources wrap-up and offer extension (90 minutes).

TOTAL: 5850 minutes (97.5 hours)

Wow, that's quite a lot of time, but it doesn't seem an insurmountable number, does it? Sure, it's a lot — about one and a half weeks' worth of time, but is that really all that much, when you think about it? Well, if I were an infomercial I would say, "Wait before you decide — that's not all you get! There's more!"

A new employee is just that: new. He or she is spending a lot of time getting acclimated, attending employee orientation and training courses, reading documentation, getting used to standards, asking a lot of questions, making small mistakes, etc. In my environment it usually takes about eight weeks before a person really starts to settle in and begin getting some serious work done. During this time, I usually figure that about 40% of their hours are spent dealing with these kinds of issues, during which they will

also absorb about 10 to 20 hours of my existing staff's time being guided and mentored. Therefore, we additionally have:

8. Ramp-up time (8 weeks * 40 hours * 40 % = 128 hours).
9. Time lost by staff mentoring (20 hours).

This would bring us to a grand total of somewhere around 245 hours, which is over 6 work weeks.

Okay, that is beginning to be a more sizable number. Now, if you want to see something really scary, let's show what that number actually means. Many companies, mine included, tend to assume that the average employee costs them somewhere in the range of $100 an hour. This includes health benefits, costs for equipment and maintenance, infrastructure servers, IT staff, cleaning crew, building rent, facilities, etc. In other words, the cost to employ a person is far more than that person's salary — it's everything that is needed to support them as well.

If you now multiply our hourly figure of roughly 245 hours by $100 per hour, you come out with a grand total hiring cost of roughly $24,500.00 to hire one person.

Now think about what your attrition or turnover rate is at your company. As we said, many companies lose upwards of 30% of their people each year. Let's consider my organization — I have roughly 20 people on staff. If I lose and have to replace 30% of my staff each year, that means that I will have to hire six people yearly. If you figure it will cost me $24,500 for each hire, then I basically need to account somehow for nearly $150,000 each year in lost productivity and other expenses. I don't know about you, but there is a really slick Porsche 911 Twin-Turbo that I've been wanting that costs less than that. I figure my company could just buy me one and call it even; per my calculations, they owe me one a year, and I'm not figuring in the fact that the attrition rate was closer to 100% before I got there.

What's worse is that this figure doesn't get into the harder-to-nail-down aspects of opportunity costs (guns vs. butter — basic economics), expensive loss of knowledge and familiarity with experienced members, revenue losses in useful and possibly proprietary information traveling with ex-employees to competitors, etc.

So we've managed to put a reasonably disturbing dollar figure on the cost of losing employees. Unfortunately, I'm afraid there is even more to consider.

What about the fact that losing employees is really harrowing for the people in your environment? Many people who leave a company also leave behind a lot of friends who appreciated them not only on a professional, but on a personal level as well. These people left behind will feel the absence, and morale can suffer because of it.

Additionally, people leaving tend to create quite a wake, dragging with them at least one or two other employees. Sometimes it is to go to the same company; other times, it was just seeing someone get free of the unpleasantness of the work environment that inspires others to do the same thing. It brings to reality the prospect of a better world someplace else, and this pushes a lot of people over the edge.

The loss of a key person or two in an organization also creates feelings of distress for those left behind to "take up the slack." They can be riddled with feelings like, "I'm alone and have no one to ask questions or learn from, I need to leave," or "I cannot possibly fill this role, I need to leave," or "I really want to step up to the plate and fill this role, but the boss feels I am unable, so I will leave."

On the more sinister side, while many people are professional, there are a great many more people who, on the way out, are likely to share their feelings about the workplace a little more openly with their staff members. This can be bravado, it can be a last dig at the management, or it can just be venting. Whatever the case, this behavior is common and quite destructive, as it quickly and easily causes huge morale problems among the remaining staff.

While there are certainly many more examples, I believe I have made my point sufficiently: losing good employees is always a very bad thing. It's expensive to your organization in many ways, and more often than not it's also fairly unpleasant for all involved.

Please look for part two of this series, where we will go into the specifics of how to keep your employees happy within your organization.

A Study on Incident Costs and Frequencies

by Virginia Rezmierski, Adriana Carroll, and Jamie Hine
<*ver@umich.edu*>
<*adriana_carroll@hotmail.com*>

The Final Report for I-CAMP I, and the Final Report for this study, I-CAMP II, contain detailed descriptions of the cost-analyzed incidents, many additional details about cost and occurrence factors, and statistics regarding frequencies. The appendices to these reports also contain significant additional information. Both reports may be obtained by sending email to dlwhite@cic.uiuc.edu. The cost for each of the reports is $20.00 plus $3.00 shipping and handling (U.S. currency).

In 1999, the USENIX Association funded a project at the University of Michigan entitled "The Incident Cost Analysis and Modeling Project II"
(I-CAMP II). This article provides a brief overview of the project efforts
and findings.

The Problem

The implementation and rapid evolution of information technology (IT) resources at colleges and universities have increased the number of security and risk management issues. Physical and electronic security processes, common to the mainframe environment, are often not suitable in the more distributed computing environment that exists today on campuses. Several features of the new environment contribute to the increased security and risk management issues:

- Personnel skills and knowledge — Individuals who handle these distributed services as system administrators have differing levels of sophistication regarding the technology, laws, and ethics governing data security.
- Unfavorable trends — While threats to and attacks on distributed systems are increasing, administrator awareness of security issues on these systems is just beginning to develop. Sufficient resources are not yet being devoted to systems and network security.
- Time and skill requirements — College and university system administrators do not have sufficient time, and in some cases skills, to keep systems and networks operating, to address known vulnerabilities in operating systems and various applications, and to detect those vulnerabilities that are not readily obvious and investigate penetrations to systems.
- Management implications — Risk managers, accustomed to thinking in terms of risks against which the organization can insure, find themselves behind innovation in the area of information technology. Like system administrators, they find it difficult to convince senior managers of the need for more attention to management of IT risks and of systems security.

Given these trends, information is needed about the IT-related incidents occurring on campuses and about their actual and potential costs to the organizations.

The First I-CAMP Study

In 1997, the first "Incident Cost Analysis and Modeling Project" (I-CAMP), was funded by the Chief Information Officers of the CIC (Committee for Institutional Coopera-tion/Big 10) Universities. The object of the study was to design a cost-analysis model for IT-related incidents and to gather and analyze a sample of such incidents.

No particular incident type was sought for that study. For purposes of the first study, and extended to the present (I-CAMP II) study, "incident" was defined as:

> Any event that takes place through, on, or constituting information technology resources requiring a staff member or administrator to investigate and/or take action to reestablish, maintain, or protect the resources, services, or data of the community or of its individual members.

The first I-CAMP study examined 30 IT-related incidents, and researchers found that:

- 210 employees were involved in incident investigation/resolution;
- 9,078 employee hours were devoted to incident investigation/resolution;
- 270,805 computer/network users were affected by the incidents;
- calculated costs for the 30 incidents exceeded $1,000,000.

The I-CAMP II Study

The study was designed to refine the cost-analysis model, analyze additional incidents to ensure the usefulness of the model, and begin to collect data regarding incident frequencies to allow managers to evaluate organizational risks and costs. In Part I of this I-CAMP II study, the researchers provide a template for identifying true costs of incidents and providing consistency in calculations. Participating schools for I-CAMP II included:

- Cornell University
- Indiana University
- Michigan State University
- Northwestern University
- The Ohio State University
- The Pennsylvania State University
- Purdue University
- The University of California, Berkeley
- The University of Chicago
- University of Illinois at Chicago
- University of Illinois at Urbana-Champaign
- The University of Iowa
- The University of Maryland
- The University of Michigan—Ann Arbor
- University of Minnesota
- Stanford University
- The University of Texas at Austin
- The University of Wisconsin—Madison

ICAMP II PROJECT OVERVIEW

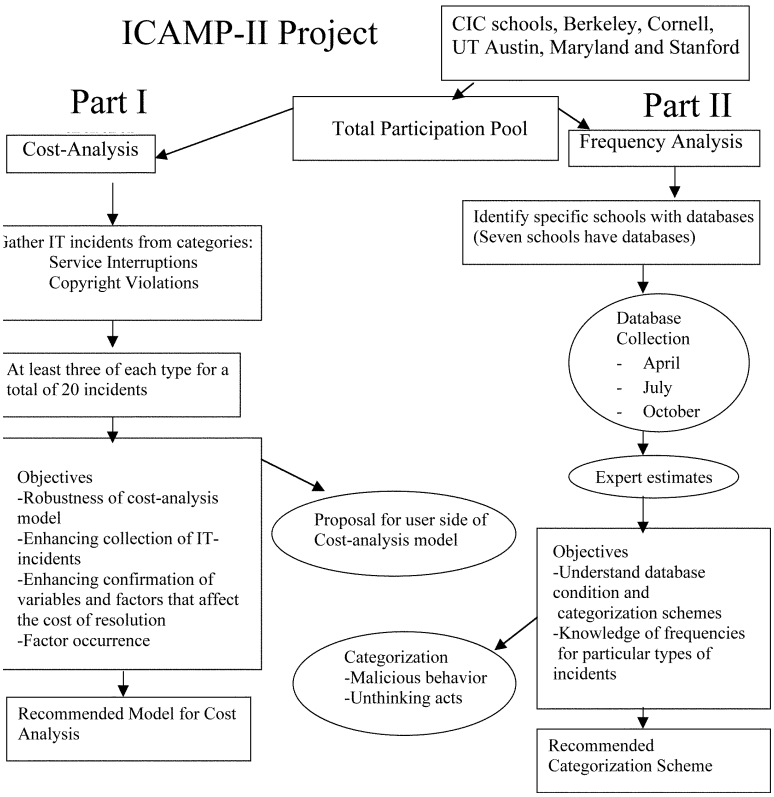


Figure 1

Figure 1 represents two major portions of the I-CAMP II study.

Incident Cost Analysis

We gathered and cost-analyzed data regarding purposeful/malicious behaviors of two types: (1) service interruptions — specifically, compromised access, insertion of harmful code, and denial of service, and (2) copyright violations — specifically, distribution of MP3 and Warez. Our goal was to augment the first sample of incidents (N=30) from the I-CAMP I study with the analysis of a small sample of these specific incident types (N=15). System security personnel from the participating schools indicated that they needed more data regarding the costs of service interruptions and copyright violations. They believed that while these incidents may be small in cost, they are occurring with high, and growing, frequency on campuses. The aggregate costs of these types of incidents may be significant.

One of the most controversial and difficult calculations to make when cost-analyzing IT-related incidents is the true cost for users when an incident occurs. If the user is a student, some individuals say that there are no costs to the institution, because the student is not paid a salary and can just do something else if the networks are down. Others say that any cost to the productivity of students, especially if downtime occurs during critical peak times such as examinations, are real costs — costs to morale, to reputation, to student time, and to productivity in general. They are costs that, while they are not included in the university budget, must be calculated in order to understand the risks to the institutional community from these types of incidents, because they indirectly affect the daily performance of IT personnel.

There are several methods that can be used for calculating the cost of student time. I-CAMP I used an average wage cost calculated from the average hourly rate paid to undergraduate and graduate part-time employees. For example,

if an incident resulted in an undergraduate student being unable to use the system for 5 hours, the calculated cost would be five hours times the average hourly wage for an undergraduate student.

In I-CAMP II we refined the user-side calculation to make it more consistent with economic theory. Calculations are based on the marginal costs to access the network and on the student's willingness to pay for one hour of study at the university level. Students choose on a rational basis where to study, depending on the tuition and fees that the university charges, which includes the availability of networks and computing systems.

When these systems are disrupted and the students are unable to work in their desired mode, it is a disruption to their time. Therefore a student has two possibilities--pay for a connection to another service provider, or wait until the university reestablishes the network service. We call the first option "the marginal cost to access the network," which is calculated as the cost of one hour of connection to a different service times the number of hours connected. The second option, "the willingness to pay for one hour of study," is a weighted average of in-state and out-of-state tuition and fees for any particular school divided by the number of hours of expected study for a full-time student. This constitutes the cost of one hour of loss of a student's time.

We tested this new model and calculated it in several of the incidents we cost-analyzed for this study. We concluded that this model is a more robust and sound model for calculating student costs in incident analysis. We recommend its use. For faculty or staff members, the cost should be calculated at his/her hourly wage.

Examples of the selected incidents were collected, described, and cost-analyzed according to the new user-side model and the study's costing template. In these 15 incidents, we found the following:

- 90 employees were involved in incident investigation and resolution.
- 506 employee hours were devoted to incident investigation and/or resolution.
- The estimated numbers of computer and network users who were affected by these types of incidents could not be calculated.
- Calculated costs for the 15 incidents totaled $59,250.
- The average calculated costs were as follows:
- For the (2) compromises of access incidents — $1,800.
- For the (3) harmful code incidents — $980.
- For the (2) denial-of-service incidents — $22,350.
- For the (3) hacker attacks incidents — $2,100.
- For the (5) copyright violations incidents — $340.

Gathering Frequency Data from Incident Databases

Part II of the I-CAMP II study goal was to understand the database condition and the categorization schemes of the participating schools, in order to begin to calculate the frequency of occurrence for particular types of incidents. We began by interviewing the contact persons at each of the 18 participating schools to identify those individuals who maintained incident databases. To our surprise, only 38% (7 of 18) maintained any form of incident database.

Of the seven schools with functioning incident databases, collection of data was still problematic. Four basic conditions made it difficult for the schools to provide data to the study:

- Too few and changing personnel — For 43% (3 of 7), new personnel or changes in personnel resulted in confusions or discontinuities in work processes. 57% (4 of 7) reported that they did not have enough staff members to maintain their logs or input data to the databases in a timely fashion.
- Confusion concerning the request — 14% (1 of 7) found that our request for data was confusing because it was more comprehensive than the data they had readily available in their database.
- Problems inputting data into the databases — 43% (3 of 7) reported that they manually entered data in the database or had to manually classify the data from email messages or flat files. Therefore the nature of their data made it difficult to fulfill our request for frequency data in the three designated time periods, April, July, and October. For 71% (5 of 7) the greatest difficulty was that because of limited resources, the databases were not kept up to date and therefore data had to be entered prior to their being able to respond to our data collection requests in each of the three time periods.

- Limited functionality of the log — 100% (7 of 7) had some difficulty responding to our request because their databases were not up to date, could not sort by incident type, or did not have information on the other variables for which we asked. For many of the respondents, their database tools were designed to be useful as recording tools, not as reporting tools. 100% (7 of 7) said that they wanted an interactive logging and sorting tool that generated periodic reports regarding frequency and types of incidents, incident trends, and other useful information.

Reoccurring Cost Factors

It is important to note that "lack of continuity" was one of the factors identified in the first I-CAMP study as contributing to the cost of incidents when they occurred. Here again, in I-CAMP II, it was seen as causing confusion and inefficiencies. Two other factors identified in the first I-CAMP study — factors that seem to contribute to the cost of incidents — also appeared again in this study. "Lack of knowledge," knowledge that would be provided by a sophisticated and fully functioning incident database, and "lack of resources," human resources to manage the data and investigate incidents, appear again, both contributing to inefficiencies and lack of desired functioning within the participating schools.

Data were collected from each of the participating schools and appear in the final project report. Since our ability to analyze the data from the database schools fell far short of our expectations due to the varied nature of the database classification schemes and the small number of schools with operational incident databases, we decided to turn again to the representatives from each of the participating schools to gather further information. Our intent was to gather estimates of the frequency of occurrences of selected types of IT incidents from experts on each of the campuses.

Expert Estimates of Frequencies

We asked campus experts to provide their estimates of the frequency of occurrence of three types of incidents: mail bombs, system probes, and Warez sites. Each representative was asked to estimate the number of incidents of each of the above types handled each year, the number identified at various points on the campus but not necessarily handled each year, and their estimate of the total number that occur on the campus each year. The I-CAMP II report provides statistics on each of these questions. Reported below are the statistics summaries.

Expert estimates regarding the occurrence, identification and handling of Mail bombs

	sum	*mean*	*median*	*range*
occurrence*	943	55.5	34	3—200
identified*	452	26.6	20	3—120
handled/logged	275	15.3	10.5	3—70

*Data was provided by only 17 of the 18 participants.

Expert estimates regarding the occurrence, identification and handling of Probes

	sum	*mean*	*median*	*range*
occurrence*	78290	4605.3	2000	515—40000
identified*	17519	1030.5	500	120—5500
handled/logged	10174	565.2	86	10—2920

*Data was provided by only 17 of the 18 participants.

Expert estimates regarding the occurrence, identification and handling of Warez

	sum	*mean*	*median*	*range*
occurrence*	932	54.8	27	3—400
identified*	301	17.7	12	0—56
handled/logged	270	15	11	0—56

*Data was provided by only 17 of the 18 participants.

In summary, it was striking that so many of the experts were so similar in their estimates of occurrences, identified incidents on campus, and handled incidents. Our data suggest that in estimating incidents, school size was not necessarily reflected in the size of the estimates. For mail bombs, approximately 30% of the incidents that were perceived to be occurring on campus were thought to be logged and handled, regardless of the size of the school. For system probes, the range of estimates was very large. This may indicate that the experts were truly guessing without any basis for their perceptions, or that they perceive very large numbers and know that they are unable to detect and handle even a small portion of those incidents.

It is interesting to note that for Warez sites, an incident type about which schools have been aware and educated, the percentage of those incidents perceived to be logged and handled relative to those occurring on the campus is much higher than for the other two types of incidents measured, especially probes. The low perceived occurrence of Warez sites may be the result of campus actions to combat copyright violations of software, resulting in decreases. Or it may be the result of the diverted attention of the experts to new types of copyright violations on campus; MP3 sites have overshadowed the older type of Warez incidents.

Toward a Comprehensive Categorization Scheme

Our study participants told us that they wanted an interactive database tool that would help them record and categorize incidents, that would assist them in investigating and categorizing data on incidents, and that would provide reporting functionality. But when data were collected from each of the schools having an incident database, we found that the categorization schemes being used by the different schools varied greatly. Without coherence among these schemes, no comparative or trend data can be analyzed across institutions. Therefore, we asked if a comprehensive category system for incident types existed. Our review of the literature indicated that the answer was no.

Several authors have focused attention on incidents that result from system-related vulnerabilities. Others have categorized a wider range of human-system interactions that result in both intentional and accidental IT-related incidents. Our review of the incident databases, the literature, and our research from I-CAMP I and I-CAMP II indicate that incidents fall into, and can best be understood by examining, three TARGET groups:
(1) incidents that target operating system (OS)/applications and exploit vulnerabilities; (2) incidents that target information/data and result in stolen or modified information, and (3) incidents that target humans or interpersonal interactions, and using technology, pr*Figure 2*imarily affect individual vulnerabilities and sensitivities.

Colleges and universities are not solely interested in the vulnerabilities that exist in operating systems and networks, except in areas of technical development and research. Neither are they solely concerned about vulnerabilities in data except insofar as they are accountable for data accuracy. Finally, they are not solely interested in human vulnerabilities except insofar as they affect the development of members of their community. Especially in colleges and universities, it is the interaction of humans, purposeful or accidental, with the vulnerabilities in the other areas that bring the focus upon the incidents we are studying.

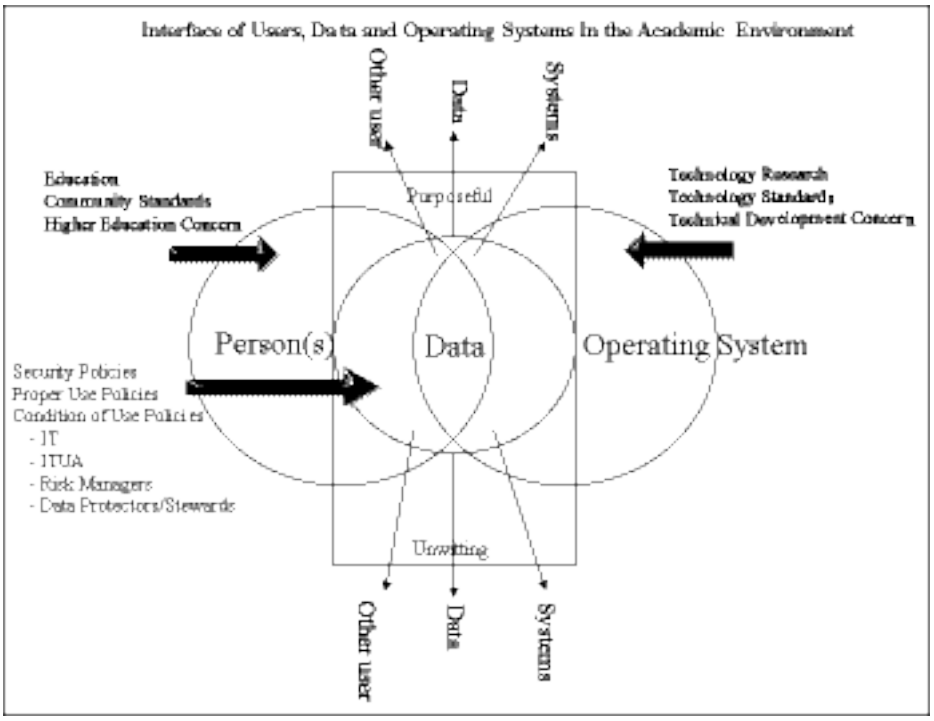


Figure 2 illustrates this interface of users, data, and operating systems that is important to academic environments. By viewing these incidents in this manner, insights into appropriate interventions or incident handling seem to arise.

I-CAMP II Categorization Model

A final step in the I-CAMP II project was to build upon the data collected in I-CAMP I and II, our review of the literature, and the notion of incident targets, and offer an incident-analysis model. Our beginning model focuses on the target of the incident (systems, data, or people), as well as a determination about whether the incident was an intentional or unintentional act, to help classify the incident. We have provided examples of incidents that we believe fall into some of the resulting categories. However, these examples are not meant to be all-inclusive. Using such a model, we believe that, over time, a comprehensive categorization scheme can be developed which will facilitate inter- and intra-institutional sharing of incident information, will improve internal reliability of incident classification, and will potentially improve consistency and justice in incident handling.

Summary and Conclusions

The I-CAMP II study confirmed the usefulness of a common template for gathering data on IT-related incidents. The study expanded, beyond the first study, the number and geographical representation of participating schools in the study. The study gathered and analyzed fifteen new incidents that were underrepresented in the first cost-analysis study. The I-CAMP II study refined the cost-analysis model by improving the calculation used for the user-side costs. The assumption that the costs for resolving the 15 selected incident types would be low was generally confirmed. The average cost for access compromise incidents was $1,800, for harmful code incidents $980, for denial of service incidents $22,350, for hacker attacks $2,100, and for copyright violation incidents $340.

The I-CAMP II study found that out of the 18 participating schools, only 7 had incident data collections in a working database. Through in-depth interviews with representatives from each of the participating schools, we found that nearly all of the schools had difficulty aggregating incident data from across the campuses. We found that the participating schools had too few and too frequently changing personnel to maintain the incident data repository/database in the manner desired. We found that all of the participating schools wanted to have a functional and robust database tool that would help them with managing incident data and with periodic reporting functions.

A clear conclusion from this study is that colleges and universities are not currently equipped to understand the types of IT-related incidents that are occurring on their campuses. They are not currently able to identify the number or type of incidents that are occurring. They are not able to assess the level of organizational impact these incidents

are having, either in terms of direct costs such as staff time, hardware and software costs, and costs to users, or in terms of indirect costs that may result from loss of reputation or trust due to a major IT incident.

After studying expert estimates of the frequency of specific incident-type occurrences, we concluded that the expert estimates of incidents logged and handled annually were very similar to the actual frequency counts for those same incident types when compared to the data from participating school databases. The team concluded that school size did not appear to affect the level of estimates given by experts of those schools for any of the three types of incidents — mail bombs, probes, or Warez. We concluded that in general, experts believe that they are identifying and handling only about 28% of the mail bombs that are occurring campuswide, approximately 12% of the system probes, and approximately 28% of the Warez sites.

Given the diverse categorization schemes used at the 7 participating schools with databases, and the absence of systematic data collection processes at the remaining 11 schools, the I-CAMP II team concluded that a common and more comprehensive categorization scheme would be beneficial to colleges and universities. We concluded that insufficient attention is being paid to the target of IT incidents — people, data, or systems. We recommended that a comprehensive system should encompass the taxonomies of operating system vulnerabilities that appear in the literature and are being used by newly emerging vulnerability scanning tools, as well as the types of interpersonal and policy violations that are seen. We began the development of such a model.

Final Recommendations

The I-CAMP II team provided the following specific recommendations for future research and best practice:

- develop a comprehensive language and categorization scheme;
- gain widespread approval for the use of this common scheme;
- encourage college and university system administrators to routinely use the cost analysis template in documenting incidents;
- encourage the creation of a central incident-reporting and cost-analysis center at colleges and universities;
- encourage and gain wide acceptance for systematic reporting of incident information, type, frequency, management processes, and trends to senior management for risk management;
- create an interactive, comprehensive database tool which provides the desired functionality for incident handlers;
- study the reliability of inter- and intra-institutional incident categorizations;
- study the consistency of inter- and intra-institutional incident management;
- encourage widespread commitment for regular inter-institutional data-sharing regarding incident trends, costs, types, and frequencies.

REFERENCES

References highlighted in the report include:

T. Aslam, I. Krsul, and E. Spafford, 1996, "Use of A Taxonomy of Security Faults," Technical Report TR-96-051, COAST Laboratory, Department of Computer Sciences, Purdue University.

P. Neumann, 1995, *Computer Related Risks*, ACM Press, Addison-Wesley Publishing, CA.

the bookworm

by Peter H. Salus
<*peter@pedant.com*>

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.

BOOKS REVIEWED IN THIS COLUMN

TRUST AND RISK IN INTERNET COMMERCE
L. Jean Camp
Cambridge, MA: MIT Press, 2000. Pp. 279. ISBN 0-262-03271-6.

EFFECTIVE TCP/IP PROGRAMMING
Jon C. Snader
Boston, MA: Addison-Wesley, 2000. Pp. 299. ISBN 0-201-61589-4.

ESSENTIALS OF THE JAVA PROGRAMMING LANGUAGE
Monica Pawlan
Boston, MA: Addison-Wesley, 2000. Pp. 301. ISBN 0-201-70720-9.

JAVA SERVER & SERVLETS
Peter Rossbach & Hendrik Schreiber
Reading, MA: Addison-Wesley, 2000. Pp. 429. ISBN 0-201-67491-2.

JAVASERVER PAGES
Larne Pekowsky
Reading, MA: Addison-Wesley, 2000. Pp. 282 + CD-ROM. ISBN 0-201-70421-8.

THE HUMANE INTERFACE: NEW DIRECTIONS FOR DESIGNING INTERACTIVE SYSTEMS
Jeff Raskin
Reading, MA: Addison-Wesley, 2000. Pp. 256. ISBN 0-201-137937-6.

Every time I open a newspaper or click on the TV, some pundit is telling me about eBusiness or eCommerce and its importance. Most of what I read and hear is absolutely worthless. (This may indeed be true of most of what the media purvey, but I just don't know enough about most things to be able to tell. Unfortunately, my guess is that the pundits know far less than a random cabbie in a European or North American city.)

To be frank, I've only got a half-dozen keepers on electronic commerce over the past few years.

So it's really nice to get another.

L. Jean Camp's view of commerce in general is that every transaction involves assumptions of trust and risk. These, in turn, relate to security, privacy, and reliability. Years ago, I would tear up the carbons of my charge slips, so that phony cards couldn't be created from them and the imprint of my signature. Card companies' elimination of those carbons made me feel better about my security and my privacy.

In any Internet transaction, the questions of who trusts whom and what the risks are arise. Who pays when trust is misplaced (both in terms of hard cash — virtual cash? — and in terms of data)? When there are failures, who is at risk? When a third party is involved, who is liable?

Lots of questions. Jean Camp may not have all the answers, but what she's done is extremely valuable in terms of putting the questions and in delivering a jargon-free presentation of the issues.

I use eCommerce a lot: I buy things, I sell my writings, and I deliver "product" via the Internet. As time goes on, more and more of us will transact more and more business via the Internet. This book does a fine job in presenting the risks and also the security features that go to ensure our trust.

TCP/IP

By and large, I admit to relying on the (many) tomes of Doug Comer and of the late Rich Stevens when it comes to understanding TCP/IP. So I'm generally wary when yet another book on TCP/IP thuds onto my desk.

But Snader's little book is good and useful.

He used Stevens's `groff` macros, so the look-and-feel is a familiar one. There are a lot of good tips and some neat code, but the last four "tips" may easily be the most important ones:

- 41. Read Stevens
- 42. Read Code
- 43. Visit the RFC Editor's Page
- 44. Frequent the News Groups

Yep.

So, if you do networking, buy Snader, too.

Where IPv6 is concerned, there are now seven volumes of RFCs edited by Pete Loshin. In June, I mentioned this "Big Book . . ." series. Once again, my compliments to him and to Morgan Kaufmann for these.

Too Much Java Keeps You Awake

It's nearly four years since I first complained about the number of Java books I had received. Well, I need to admit that there are a few new ones that I found worthwhile.

Monica Pawlan's volume is perfect for someone who's had a first course in programming — a real beginner may have some difficulties. But Pawlan has done a fine job with her explanations, and many chapters end with sections on "further information." There is a brief but good bibliography.

Rossbach and Schreiber came out last year in German; no translator is noted, but whoever did it deserves a pat on the back. If you want to build portable Web apps, you'll want this. The bibliography is thorough, but Rich Stevens is under "R," not "S."

JavaServer Pages is a new technology that is intended to facilitate page development. JSP is included in Sun's Java 2 Enterprise Edition. The book seems good, though the compulsory history sections (pp. 2—6) are fairly vapid. The brief chapter summaries are quite handy.

Some Apologies

A while back I complained that there was no book on `grep`. Several readers pointed out that Friedl & Oram, *Mastering Regular Expressions* (O'Reilly, 1997) has a lot on both `grep` and `egrep`. That's true, and Friedl and Oram have a fine book — but it goes into Perl, Python, Emacs, etc., as well as `grep`.

Second, I wrote that Ted Dolotta created the `-mm` macros. Ted wrote me:

> What you say there is not quite correct: it is true that I decided to have the `-mm` package written (there was some feeling at the time that Mike Lesk's `-me` macros could be improved/enlarged upon). But most of the design and implementation was done by John Mashey and Dale Smith. I provided the overall management, critiqued the design and the documentation (surprise!), and served as alpha tester. I also designed and implemented the whole footnote mechanism — a very complicated piece of `troff` code which was subsequently broken and never repaired when someone (who did not understand it) tried to add a feature to automatically print legal notices at the bottom of every page below the footnotes; turnover footnotes have never worked since.

I do try to get things right . . .

Jef Raskin
The Humane Interface

Reviewed by Steve Johnson

Jef Raskin is best known as the designer of the Macintosh interface. In *The Humane Interface*, he has written a very entertaining and, in the best sense, radical book about how we deal with computers and applications.

Probably half the book is spent putting together a model that describes how people interact with programs. Fairly high-level concepts ("you can only focus on one thing at a time") were mixed artfully with much more mundane ideas (such as a model that estimates how long the average user will take to, for example, click in a dialog box and type 10 characters of text). I found myself saying "of course" over and over again, as he pointed out obvious things that I'd just never brought into consciousness.

As one example, Fitt's law predicts how long it will take to move the cursor and click a button, based on the size of the button and how far you need to move the cursor. Halving the dimensions of a button can add nearly 150 msec. to the time it takes to press it (and this doesn't count the increased error rate). I recently "upgraded" a program I use frequently, and found the upgrade significantly harder to use. After reading this book, I now see that the buttons' being smaller is a lot of the reason.

He also has a discussion of information theory, and the uses and abuses of it in interface design. His goal for an interface is that it become monotonous — like a perfect waiter, it serves you without calling attention to itself. He loathes error boxes where the only thing you can do is say "OK." His critiques of the Windows interface are bullseye accurate — it becomes hard to use any Windows application after reading this book, because you keep being aware of how much harder you are working than you need to.

Finally, he discusses a number of radical ideas for interfaces. One is based on a two-dimensional infinite field where data can be piled, and you can zoom in and out to find the information you need. Another is a truly radical notion of eliminating file names from the user interface. As a long-time UNIX user, this idea almost sent me into shock, but it's good to have your world view upset every couple of decades!

Finally, the book is very engaging to read. It is well laid out, written in a light, somewhat ironic style, and peppered with amusing quotes. It would be an excellent read for anyone involved in generating user interfaces (and who isn't?).

Sun Comes Through for USENIX

by Jane-Ellen Long, Managing Editor, *;login:*

Thanks to the help of Hal Stern, Distinguished Engineer at SUN Microsystems, USENIX has received a donation of two Enterprise 250s, with dual 400MHz UltraSPARC II processors with 2MB cache, 8 256MB memory expansions for each, 2 internal 18.2GB 10,000 RPM drives, 19" color monitors, and Solaris 7 OS.

This equipment will enable USENIX to replace its aging servers and generally improve performance both internally and externally.

Thank you, Sun Microsystems!