Advanced search

Linux Journal Issue #97/May 2002

Groking INTRUSION DETECTION Systems
LINUX JOURNAL
SHAKE IT UP!
Cool Cats
KERNEL INTERNALS
Is Linux still CUTTING EDGE?
We talk to kernel developer TED TS'O
BUILDING ADVANCED iptables
BEOWULF basics

Lowering Latency in Linux: Introducing a Preemptible Kernel

Robert Love

Issue #97, May 2002

Whether you seek higher scores in Quake, are an audio enthusiast or want a smoother desktop, lowering latency in the kernel is an important goal.

Performance measurements come in two flavors, throughput and latency. The former is like the width of an expressway: the wider the expressway, the more cars that can travel on it. The latter is like the speed limit: the faster it is, the sooner cars get from point A to point B. Obviously, both quantities are important to any task. Many jobs, however, require more of one quality than of the other. Sticking to our roadway analogy, long-haul trucking may be more sensitive to throughput, while a courier service may be more demanding on latency. Lowering latency and increasing system response, through good old-fashioned kernel work, is the topic of this article.

Audio/video processing and playback are two common beneficiaries of lowered latency. Increasingly important to Linux, however, is its benefit to interactive performance. With high latency, user actions, such as mouse clicks, go unnoticed for too long—not the snappy responsive desktop users expect. The system cannot get to the important process fast enough.

The problem, at least as far as the kernel is concerned, is the nonpreemptibility of the kernel itself. Normally, if something sufficiently important happens, such as an interactive event, the receiving application will get a priority boost and subsequently find itself running. This is how a preemptively multitasked OS works. Applications run until they use up some default amount of time (called a timeslice) or until an important event occurs. The alternative is cooperative multitasking, where applications must explicitly say, “I'm done!”, before a new process can run. The problem, when running in the kernel, is that scheduling is effectively cooperative.

Applications operate in one of two modes: either in user space, executing their own code, or within the kernel, executing a system call or otherwise having the kernel work on their behalf. When operating in the kernel, the process continues running until it decides to stop, ignoring timeslices and important events. If a more important process becomes runnable, it cannot be run until the current process, if it is in the kernel, gets out. This process can take hundreds of milliseconds.

Latency Solutions

The first and simplest solution to latency problems is to rewrite kernel algorithms so that they take a minimal, bounded amount of time. The problem is that this is already the goal; system calls are written to return quickly to user space, yet we still have latency problems. Some algorithms simply do not scale nicely.

The second solution is to insert explicit scheduling points throughout the kernel. This approach, taken by the low-latency patches, finds problem areas in the kernel and inserts code to the effect of “Anyone need to run? If so, run!” The problem with this solution is that we cannot possibly hope to find and fix all problem areas. Nonetheless, testing shows that these patches do a good job. What we need, however, is not a quick fix but a solution to the problem itself.

The Preemptible Kernel

A proper solution is removing the problem altogether by making the kernel preemptible. Thus, if something more important needs to run, it will run, regardless of what the current process is doing. The obstacle here, and the reason Linux did not do this from the start, is that the kernel would need to be re-entrant. Thankfully, the issues of preemption are solved by existing SMP (symmetric multiprocessing) support. By taking advantage of the SMP code, in conjunction with some other simple modifications, the kernel can be made preemptible.

The programmers at MontaVista provided the initial implementation of kernel preemption. First, the definition of a spin lock was modified to include marking a “nonpreemptible” region. Therefore, we do not preempt while holding a spin lock, just as we do not enter a locked region concurrently under SMP. Of course, on uniprocessor systems we do not actually make the spin locks anything other than the preemption markers. Second, code was modified to ensure that we do not preempt inside a bottom half or inside the scheduler itself. Finally, the return from interrupt code path was modified to reschedule the current process if needed.

On UP, spin_lock is defined as preempt_disable, and spin_unlock is defined as preempt_enable. On SMP, they also perform the normal locking. So what do these new routines do?

The nestable preemption markers preempt_disable and preempt_enable operate on preempt_count, a new integer stored in each task_struct. **preempt_disable** effectively is:

```
++current->preempt_count;
barrier();
```

and preempt_enable is:

```
--current->preempt_count;
barrier();
if (unlikey(!current->preempt_count
    && current->need_resched))
    preempt_schedule();
```

The result is we do not preempt when the count is greater than zero. Because spin locks are already in place to protect critical regions for SMP machines, the preemptible kernel now has its protection too.

preempt_schedule implements the entry to **schedule** itself. It sets a flag in the current process to signify it was preempted, calls schedule and, upon return, unsets the flag:

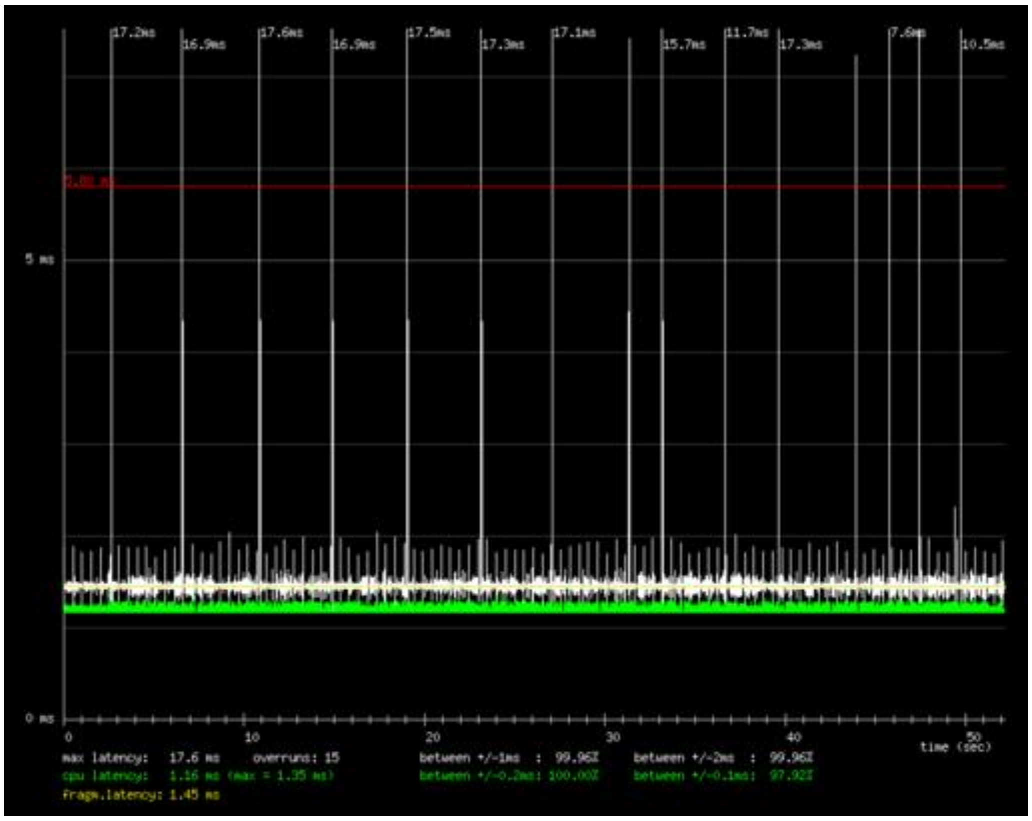
```
asmlinkage void preempt_schedule(void)
{
    do {
        current->preempt_count += PREEMPT_ACTIVE;
        schedule();
        current->preempt_count -= PREEMPT_ACTIVE;
    } while (current->need_resched);
}
```

The other entry to preempt_schedule is via the interrupt return path. When an interrupt handler returns, it checks the preempt_count and need_resched variables, just as preempt_enable does (although the interrupt return code in entry.S is in assembly). The ideal scenario is to cause a preemption here because it is an interrupt that typically sets need_resched due to a hardware event. It is not always possible, however, to preempt immediately off the interrupt, as a lock may be held. That is why we also check for preemption off preempt_enable.

The Results

Thus, with the preemptive kernel patch, we can reschedule tasks as soon as they need to be run, not only when they are in user space. What are the results of this?

Process-level response is improved twentyfold in some cases. (See Figure 1, a standard kernel, vs. Figure 2, a preemptible kernel.) These graphs are the output of Benno Senoner's useful latencytest tool, which simulates the buffering of an audio sample under load. The red line in the graphs represents the amount of latency beyond which audio dropouts are perceptible to humans. Notice the multiple spikes in the graph in Figure 1 compared to the smooth low graph in Figure 2.



```
max latency:   17.6 ms     overruns: 15         between +/-1ms   :  99.96%     between +/-2ms   :  99.96%
cpu latency:   1.16 ms (max = 1.35 ms)          between +/-0.2ms: 100.00%      between +/-0.1ms:  97.92%
fragm.latency: 1.45 ms
```

Figure 1. Result of a Latency Test Benchmark on a Standard Kernel

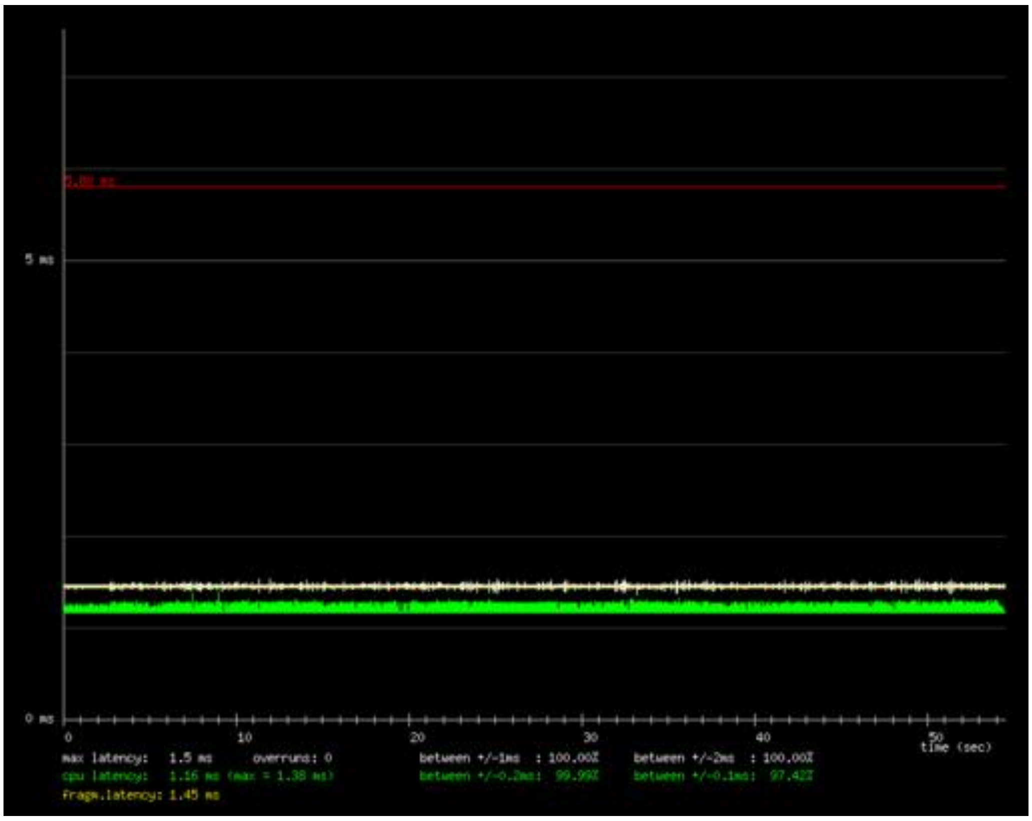


Figure 2. Result of a Latency Test Benchmark on a Preemptible Kernel

The improvement in latencytest corresponds to a reduction in both worst-case and average latency. Further tests show that the average system latency over a range of workloads is now in the 1-2ms range.

A common complaint against the preemptible kernel centers on the added complexity. Complexity, opponents argue, decreases throughput. Fortunately, the preemptive kernel patch improves throughput in many cases (see Table 1). The theory is that when I/O data becomes available, a preemptive kernel can wake an I/O-bound process more quickly. The result is higher throughput, a nice bonus. The net result is a smoother desktop, less audio dropout under load, better application response and improved fairness to high-priority tasks.

Table 1. Throughput Test: dbech Runs

Changes to Programming Semantics

Kernel hackers are probably thinking, “How does this affect my code?” As discussed above, the preemptible kernel leverages existing SMP support. This makes the preemptible kernel patch relatively simple and the impact to coding practices relatively minor. One change, however, is required. Currently, per-CPU data (data structures unique to each CPU) do not require locking. Because they are unique to each CPU, a task on another CPU cannot mangle the first CPU's data. With preemption, however, a process on the same CPU can find itself preempted, and a second process can then trample on the data of the first. While this normally is protected by the existing SMP locks, per-CPU data does not require locks. Data that does not have a lock, because it is protected by its nature, is considered to be “implicitly locked”. Implicitly locked data and preemption do not get along. The solution, thankfully, is simple: disable preemption around access to the data. For example:

```
int catface[NR_CPUS];
preempt_disable();
catface[smp_processor_id()] = 1;  /* index catface
                                     by CPU number */
/* operate on catface */
preempt_enable();
```

The current preemption patch provides protection for the existing implicitly locked data in the kernel. Thankfully, it is relatively infrequent. New kernel code, however, will require protection if used in a preemptible kernel.

Work for the Future

We still have work to do. Once the kernel is preemptible, work can begin on reducing the duration of long-held locks. Because the kernel is nonpreemptible when a lock is held, the duration locks are held corresponding to the system's worst-case latency. The same work that benefits SMP scalability (finer-grained locking) will lower latency. We can rewrite algorithms and locking rules to minimize lock held time. Eradicating the BKL will help too.

Identifying the long-held locks can be as difficult as rewriting them. Fortunately, there is the preempt-stats patch that measures nonpreemptibility times and reports their cause. This tool is useful for pinpointing the cause of latency for a specific workload (e.g., a game of *Quake*).

What is needed is a goal. Kernel developers need to consider any lock duration that extends over a certain threshold, a bug for example, 5ms on a reasonably modern system. With that goal in mind, we can pinpoint and ultimately eliminate the areas of high latency and lock contention.

Conclusion

The Linux community is large and diverse, and Linux is used in embedded systems all the way through large servers. Preemptive kernel technology provides benefits beyond real-time applications. Desktop users, gamers and multimedia developers alike stand to benefit from reduced latency. A solution is needed for both the 2.4 and 2.5 kernel trees; perhaps the same solution for each is not best. With 2.5 under development, however, now is the time to implement a feature that provides an immediate gain, as well as the framework for further improvement. The result will be a better kernel.

Resources

Robert Love (rml@tech9.net) is a Mathematics and Computer Science student at the University of Florida. When not hacking Linux, Robert enjoys auto racing, Thai food and punk rock.

Archive Index Issue Table of Contents

Advanced search

How the PCI Hot Plug Driver Filesystem Works

Greg Kroah-Hartman

Issue #97, May 2002

Greg describes how the PCI Hot Plug core implements a RAM-based filesystem and how you can do the same for your drivers.

On May 14, 2001, H. Peter Anvin announced to the linux-kernel mailing list: “Linus Torvalds has requested a moratorium on new device number assignments. His hope is that a new and better method for device space handling will emerge as a result.”

Peter is the “Linux Assigned Names and Numbers Authority”, meaning that all kernel driver authors had to go through him to get a major and minor number pair for their drivers. The freeze on assigning new numbers naturally caused a lot of discussion about what this “better method” for device space handling would be. One idea that emerged was making a driver that could implement a filesystem to control the user-space interaction with the driver.

During this time, I was cleaning up the PCI Hot Plug driver written by Compaq for their servers. A PCI Hot Plug driver allows you to shut down a PCI card while the machine is running, pull out the card, replace it with another one and then power that card back on, if you have the proper hardware on your motherboard. This is very useful for servers that cannot be shut down but need to have new network cards added, faulty devices removed and other service-type operations.

The PCI Hot Plug driver was originally written to interact with user space as a character device; ioctl calls were made to the device node to shut down PCI slots, power up PCI slots, turn PCI slot indicator lights on and off and run different manufacturing tests on the device. To get information about the number of different PCI slots in the system and the state of the slots (power and indicator status), a /proc directory tree was used. This directory tree was read-only.

As I worked to split the PCI Hot Plug core functionality out of the Compaq driver, so that other PCI Hot Plug drivers would have a common interface for the user, I realized that a single filesystem would be a better fit both to show PCI-slot information and to allow user control. All information and control over the driver would be handled from one place, instead of having two different types of interfaces.

The PCI Hot Plug driver core has been merged into the main kernel tree as of 2.4.15, and it exports a filesystem called pcihpfs that is used to control the driver. When you mount the filesystem, you get a tree with directories called 3, 4, 5 and so on, which are the physical numbers of the PCI slots. Every file in a slot directory can be read to find the value for that bit of information about the slot. The files power and attention can be written to in order to set the power (0 or 1) or attention (0 or 1) values. The test file is used to send hardware test commands to the hardware. The adapter file detects if an adapter is present in that slot or not, and the latch file describes the position of the physical latch (if any) for that slot.

So, you can enable the power in slot 5 to be turned with:

```
echo 1 > 5/power
```

from the pcihpfs root. If a PCI card is present in that slot, the whole PCI-initialization sequence will execute for that card, including a call out to /sbin/hotplug with the PCI information, so that the module for that device can be loaded automatically by the system [see Greg's Kernel Korner in the June 2002 issue of *LJ*].

Because of this filesystem, a user-space program does not have to make special ioctl() calls to a character device, allowing users to access a wider range of options for how they want to control their devices.

The rest of this article describes how the PCI Hot Plug core implements a RAM-based filesystem and how you can do the same thing for your drivers.

First, you need to declare the filesystem in your driver. To do this, use the DECLARE_FSTYPE macro, which is defined in the include/linux/file.h file. The pci_hotplug driver uses the DECLARE_FSTYPE macro in the following way:

```
static DECLARE_FSTYPE(pcihpfs_fs_type, "pcihpfs",
          pcihpfs_read_super, FS_SINGLE | FS_LITTER);
```

This creates a static variable of the type struct file_system_type called pcihpfs_fs_type and initializes some of the structure's fields. The name field is set to pcihpfs, which will be used by users in mounting our filesystem, so choose a name that makes sense and is not currently in use by any other filesystem in the kernel. We set the flags field to both FS_SINGLE and FS_LITTER.

FS_SINGLE means that, for this filesystem, we will have only one instance of the superblock. Therefore, wherever the filesystem is mounted in the system, all mountpoints will point to the same location in the filesystem (remember that you can mount the same filesystem at different points in a directory tree). The FS_LITTER option means that we want this filesystem to keep the tree in the dcache. This is set because our filesystem will live entirely in RAM and will not have a backing store of the data on any physical device, like a disk.

The read_super field of the pcihpfs_fs_type points to the function that will be called when the kernel wants to read the superblock of our filesystem. A superblock is the structure in a filesystem that is used to describe the entire filesystem. The kernel will call this function when the filesystem is asked to be mounted. When this function is called, we need to tell the kernel exactly what our filesystem looks like.

But before our filesystem can be mounted, we need to tell the kernel that our filesystem is present. This is done with a simple call to register_filesystem() with our file_system_type as the only parameter. This is done in the pci_hotplug module's initialization function with the following bit of code:

```
dbg("registering filesystem.\n");
result = register_filesystem(&pcihpfs_fs_type);
if (result) {
  err("register_filesystem failed with %d\n", result);
  goto exit;
}
```

Likewise, when the pci_hotplug module is being shut down, we unregister our filesystem type with the following single line of code:

```
unregister_filesystem(&pcihpfs_fs_type);
```

Right after we register our filesystem, we want to create some virtual files that will allow a user to read and write values that our driver wants to export and change. If a user mounts the filesystem before he or she wants to create a file, the kernel already will have created the filesystem at some virtual location. Odds are that the filesystem has not been mounted, however, right after it is created, we need to get the kernel to mount the filesystem before we can add a file (otherwise our file creation fails, which prevents anyone from using that file).

There are two different ways of solving this problem. The first way is to wait until our filesystem is really mounted (we know this when our read_super function is called) and then create all of our files. This method requires us to do a bunch of work at mount time and to be constantly aware of whether our filesystem is currently mounted; remember, we need to add or remove files at different points in time. The usbdevfs filesystem (no relation to devfs, just an unfortunate name similarity) is an example of a filesystem that implements this solution to the problem.

However, we don't want to keep track constantly of when our filesystem is mounted, and we would like to be able to create or remove a file whenever we want. To do this second method, we need to tell the kernel to mount our filesystem internally. This solves the problem of keeping track of the current mount state. Listing 1 shows how we accomplish this.

Listing 1. Telling the Kernel to Mount the Filesystem Internally

Let's walk through Listing 1 to try to understand what it is doing and how it is doing it. This is also a good example of how to do proper locking techniques for when the kernel is running on a multiple-processor machine.

First we grab a spin lock, called mount_lock, with the line

```
spin_lock(&mount_lock);
```

This lock is used to protect our internal count if our filesystem is an example of what is needed to do this properly. internally mounted. Okay, previously I stated that we didn't want to keep track of whether we were mounted. Trust me, this simple function, combined with a simple function to unmount the filesystem (described later), is much easier to understand and work with than is the option of trying to determine if we have been mounted by a user. For an example of what is needed to do this properly, see the code in drivers/usb/inode.c in the 2.4.18 and earlier kernels.

After we grab our spin lock, check to see if our internal mount variable has been set:

```
if (pcihpfs_mount) {
        mntget(pcihpfs_mount);
        ++pcihpfs_mount_count;
        spin_unlock (&mount_lock);
        goto go_ahead;
}
```

If it has been set, we call mntget() to increment our internal mount count; mntget() is a simple inline function in the include/linux/mount.h file. We then increment our internal count variable, unlock our spin lock and jump to the end of the function, as we are finished (yes, it's okay to use **goto** in the kernel, sparingly).

Otherwise, we have not mounted this filesystem yet. So we unlock our spin lock:

```
spin_unlock (&mount_lock);
```

and call kern_mount to mount our filesystem internally:

```
mnt = kern_mount (&pcihpfs_fs_type);
if (IS_ERR(mnt)) {
    err ("could not mount the fs...erroring out!\n");
    return -ENODEV;
}
```

We unlock our spin lock, as the kern_mount() function can take a long time and may even cause the kernel to sleep and schedule another process. Remember that you cannot hold a spin lock if schedule() can be called while the lock is held —very bad things can happen if you do this.

Now that we have mounted our filesystem, we grab our spin lock again:

```
spin_lock (&mount_lock);
```

and check to see if our internal mount variable is still zero:

```
if (!pcihpfs_mount) {
        pcihpfs_mount = mnt;
        ++pcihpfs_mount_count;
        spin_unlock (&mount_lock);
        goto go_ahead;
}
```

“Wait!”, you are saying. “Why are we looking at pcihpfs_mount? We already know that it is set to zero; we checked it just a few lines of code ago. Why check again?” Well, remember the call to kern_mount() that we mentioned could sleep? If our call to kern_mount() sleeps, and another process comes through this same piece of code (remember we are running on more than one processor, and multiple user threads could be happening at the same time), then it could have already successfully mounted our filesystem and incremented the pcihpfs_mount variable. Because of this, we need to check it again.

So if another process has not come through and mounted our filesystem, we save off the pointer to our now mounted filesystem for other functions to use later, increment our internal count, unlock our lock and exit.

But if another process already has mounted our filesystem, we then do:

```
mntget(pcihpfs_mount);
++pcihpfs_mount_count;
spin_unlock (&mount_lock);
mntput(mnt);
```

This matches what we originally did in this same situation, back at the beginning of the function.

The code to unmount our filesystem is much simpler:

```
static void remove_mount (void)
{
       struct vfsmount *mnt;
       spin_lock (&mount_lock);
       mnt = pcihpfs_mount;
       --pcihpfs_mount_count;
       if (!pcihpfs_mount_count)
              pcihpfs_mount = NULL;
       spin_unlock (&mount_lock);
       mntput(mnt);
       dbg("pcihpfs_mount_count = %d\n",
           pcihpfs_mount_count);
}
```

In this function, we simply grab our lock (the same lock we used when mounting the filesystem), decrease our count of the number of times the filesystem was mounted (we need to unmount for every time we mounted it) and unlock our lock. Then we tell the kernel that we want to unmount the filesystem with a call to mntput().

When the kernel wants to mount our filesystem—virtually because we called kern_mount() or because a user mounted it first—our pcihpfs_read_super() function is called. In it, we need to set up a few kernel structures that describe what our filesystem looks like and list where to find the functions that the kernel will call during the lifetime of the filesystem. This is done with the following lines of code:

```
sb->s_blocksize = PAGE_CACHE_SIZE;
sb->s_blocksize_bits = PAGE_CACHE_SHIFT;
sb->s_magic = PCIHPFS_MAGIC;
sb->s_op = &pcihpfs_ops;
```

With this, we state that our filesystem's block size is equal to the page cache size; we set up our filesystem's magic number (must be unique across all filesystems in the system) and point to our list of super_operations structure functions.

Then we initialize the superblock's root inode by doing:

```
inode = pcihpfs_get_inode(sb, S_IFDIR | 0755, 0);
if (!inode) {
      dbg("%s: could not get inode!\n",__FUNCTION__);
      return NULL;
}
```

We will describe what pcihpfs_get_inode() does in a bit, but if that succeeds, we then allocate the root dentry for the inode we just created and save that dentry in the superblock structure:

```
  root = d_alloc_root(inode);
  if (!root) {
       dbg("%s: could not get root dentry!\n",
           __FUNCTION__);
       iput(inode);
       return NULL;
  }
  sb->s_root = root;
  return sb;
```

That is all we need to do to initialize our superblock, and now the kernel has mounted our filesystem.

pcihpfs_get_inode() is another function that we need to create for our filesystem. It is called whenever we need to create a new inode for our filesystem. Listing 2 shows what the pci_hotplug driver uses to do this.

Listing 2. Creating a New Inode

First we call the kernel new_inode() function in order to create and initialize a new inode structure. If this succeeds, we then proceed to fill up a number of the fields with the necessary information. The i_uid and i_gid members are set to the current process' uid and gid values, insuring that whoever has the permission to create the inode can access it later. The i_atime, i_mtime and i_ctime members refer to the inode's access time, last modified time and time of last change. We set all of these variables to the current time. If this inode is a "regular" file type, then we point to our set of default_file_operations as the set of functions that should be called whenever the inode is acted upon (open, write, read, etc.). If this inode is a directory inode, we point to our default set of directory inode functions. And if the inode is neither a regular inode nor a directory inode, we then let the kernel initialize it with a call to init_special_inode().

So, now that the filesystem is internally mounted, how do we create a file that a user can read and write to? To do this, we call our fs_create_file() function, passing in the name of the file we want to create, the mode of the file, a pointer to the parent directory of the file (if this is NULL, we default to the root directory of the filesystem), a pointer to a blob of data that we want assigned to this file and a pointer to a set of file operations that will be called when the file is accessed (see Listing 3).

Listing 3. Creating a File that a User Can Read and Write to

Here we call pcihpfs_create_by_name to get a new dentry with all of the specified information. After our new dentry is created, we save off our data pointer and point the dentry file_operations to the one we really want to have called when this dentry's inode is accessed.

The struct file_operations that we assign to an inode differs depending on the kind of file we created. For the “power” file, which reports if the specific PCI slot is on or off and also controls turning the slot on or off, we use the following structure:

```
static struct file_operations power_file_operations = {
        read:         power_read_file,
        write:        power_write_file,
        open:         default_open,
};
```

The interesting functions here are power_read_file and power_write_file. This is what is called whenever the file is read from or written to. The other functions are called when the different operations are made on the file. When open() is called, the kernel calls default_open; when llseek is called, the kernel calls default_file_lseek() and so on.

power_read_file() is a fairly simple function. All we want to do is return the current power status of the specific PCI slot. The code to do this is:

```
page = (unsigned char *)__get_free_page(GFP_KERNEL);
if (!page)
        return -ENOMEM;
retval = get_power_status (slot, &value);
if (retval)
        goto exit;
len = sprintf (page, "%d\n", value);
```

This code allocates a chunk of memory (one page), gets the power status of a specific PCI slot (through the call to get_power_status()) and then writes a string representation of this status to the chunk of memory. The chunk of memory is then copied into user space. Remember, the original memory is located in kernel space; if you want the user to be able to see the memory, you need to call

```
if (copy_to_user (buf, page, len)) {
       retval = -EFAULT; goto exit;
}
```

where buf is a pointer to the user-space buffer that was originally passed to the read() call. So when a user issues the command:

```
cat /tmp/pcihpfs/slot2/power
```

the result is:

```
1
```

The power_write_file() function is equally as simple. We want the user to be able to control the power of a PCI slot with a simple echo command, something like

```
echo 1 > /tmp/pcihpfs/slot3/power
```

to turn on the power to the third PCI slot in the system. To do this, we need to convert the string representation of the value that is passed to us into a binary number and determine what slot-specific function to call (see Listing 4).

Listing 4. Controlling the Power of a PCI Slot

First we create a buffer that is one byte bigger than the user string and fill it with zeros. Then we copy the buffer from user space into our kernel buffer, convert it into a binary number with the simple_strtoul() function, and then act on the value of the binary number by either calling disable_slot() or enable_slot() on the specified PCI slot.

With those two simple functions mentioned above, we have now created a driver interface that can be accessed by any user, without needing to make special ioctl-type calls.

When the driver shuts down, it needs to remove all of the files that it had originally created in the filesystem, in order to be allowed to unmount the filesystem and free up all of the allocated memory. To do this, it calls the fs_remove_file() function (see Listing 5).

Listing 5. Calling the fs_remove_file() Function

This function needs a pointer to the dentry that the call to fs_create_file returned. It determines if the dentry has a valid parent, as you need the parent of the dentry in order to be able to remove it. Then it calls into the kernel VFS layer to remove the dentry (different calls are made depending on whether the dentry refers to a directory or to a file).

We have described the basic filesystem functions that are needed to implement a filesystem in a driver. For a better description of how all of the different pieces work together, look at the code in the drivers/hotplug/pci_hotplug_core.c file in the Linux kernel tree.

This article has been based on what is necessary for the 2.4 kernel. The 2.5 kernel should make things even easier, due to the exporting of most of the ramfs functions. This will enable more code sharing among the RAM-based filesystems, decreasing the amount of work a driver author has to do and preventing the author from doing things incorrectly.

Acknowledgements

I would like to thank Pat Mochel for writing the ddfs/driverfs code upon which a lot of the pcihpfs code was originally based. **driverfs** is a new filesystem in the 2.5 kernel that will also help driver authors in exporting driver-specific information into user space, as well as provide a tree of all devices, making power management tools much easier.

I would also like to thank Al Viro for answering a lot of VFS-related questions and for enabling a filesystem to be written with such a small amount of code.

Resources

Greg Kroah-Hartman is currently the Linux USB and PCI Hot Plug kernel maintainer. He works for IBM, doing various Linux kernel-related things and can be reached at greg@kroah.com.

Netfilter 2: in the POM of Your Hands

David A. Bandel

Issue #97, May 2002

David gives detailed information on working with Netfilter. See next month's Kernel Kornter for even more on this topic. those targets.

Following the publication of “Taming the Wild Netfilter” in the September 2001 issue of *LJ* [/article/4815], I received a number of e-mails, most asking for more detailed information on working with Netfilter. To satisfy those requests, this time I will delve a little deeper. For those of you who haven't read and tried out a basic setup, I suggest you do so. This article is slightly more advanced and intended for those who have a grasp, tenuous as it may be, on the basics as described in the aforementioned article.

Preparing Your System for an iptables Upgrade

In order to get the most out of Netfilter and the user-land component iptables, you'll need to upgrade both your kernel and iptables. While there's nothing wrong with the kernel and iptables that came with your distribution, the Netfilter code is under constant development. You also certainly might have no idea what patches your distribution saw fit to include in the iptables area (probably none). And, not all patches show up as Netfilter modules or iptables match extensions. I do, however, recommend you don't try what's in this article for the first time on your currently running firewall. Make sure you know what to expect by experimenting on a test system.

The final recommendation in the paragraph above brings up a very important point. This article is based on iptables-1.2.4 and the Linux kernel 2.4.17. Your results will almost certainly vary if you use different versions. The principles will be the same. Don't panic; just try to make some intelligent decisions about what you want. Also understand that just as oil and water don't normally mix, some of the choices you make regarding the modules you want will affect other modules in the same way—that is, some modules don't mix well with others. Looking at the 2.4.18-prepatches, some of the iptables patches applied for this article will be incorporated in 2.4.18. I suggest a look at the 2.4.18 Changelog when this kernel version is finally released to see what you now won't need to try to patch (the patch would fail anyway, detected or not—see below).

In this article, we'll be using Rusty Russell's patch-o-matic that will patch both the iptables and kernel sources. This patch-o-matic (POM) isn't completely automatic and will not attempt to patch anything without your approval. It also will test the patch to be applied first to see if it applies correctly. If it doesn't, you will be told and given several options. If a patch doesn't succeed, your best and safest bet is to continue without applying it. But we'll see that as we go along.

First, download the latest kernel version you want to use (available from www.kernel.org). It can be 2.4.16 or higher. I always recommend waiting at least a week after the latest stable version is out before trying it. That way, if some small bug has made its way into the latest kernel (2.4.15's shutdown filesystem corruption bug comes to mind), you'll probably know about it and avoid a potentially nasty situation.

Using whatever method appeals to you, open and configure your new kernel. This article won't cover kernel building, but a number of articles and sites can bring you up to speed on this (the definitive guide is found in your kernel source tree under Documentation/Changes). I suggest you configure as modules all the Netfilter code. For now, you'll need to choose at least:

1. Code maturity level options-->Prompt for development and/or incomplete code/drivers

and

2. Networking options-->Network packet filtering (replaces ipchains)

and from here also go ahead and enter

-->TCP/IP Networking-->IP: Netfilter Configuration (click to go to subpage)

3. On the IP: Netfilter Configuration subpage configure all modules.

If you want, select the IPv6 protocol, and you can then also configure the IPv6 Netfilter modules. You'll need to proceed at least as far as the “make dep” step with this kernel to get everything prepared.

By the way, if you read near the bottom of the Help that comes with the Network packet-filtering choice, you'll find you should choose Y if your system will act as a router; otherwise select N, and if unsure, select no. I don't know how sage this advice is. Even simple hosts often need the extra protection that can be afforded by Netfilter. You'll have to decide that question for yourself based on your best risk assessment for your network and how the host will be used. We'll see how Netfilter is, in fact, used on machines other than routers, below.

Digging into iptables

Once you have your kernel ready, download and open the latest iptables (available from netfilter.samba.org). Change into the iptables directory, and you're almost ready to start. If your kernel is not located in /usr/src/linux, then you'll need to tell iptables where to find it. Additionally, if you don't want to install iptables in /usr/local/, you'll need to specify where you want to install it. There's a reason for each of these parameters (listed below) to be included. Since iptables will be patching the kernel, it must know where to find the kernel, and the iptables binary must know where to find the extensions. The location of the extensions is hard-coded into the binary, so you can't arbitrarily move things around later—you'll have to rebuild and re-install.

The following arguments are available for iptables builds:

- KERNEL_DIR=/path/to/kernel/source (default: /usr/src/linux)
- BINDIR=/path/to/install/binaries (default: /usr/local/bin)
- LIBDIR=/path/to/install/lib-extensions (default: /usr/local/lib)
- MANDIR=/path/to/install/manpages (default: /usr/local/man)

At this point, I must note that I often work in a chroot environment, particularly when beating on the kernel sources, etc., so I don't inadvertently damage a working system. However, I've found that the patch-o-matic doesn't work properly chroot-ed. Normally, patch-o-matic will create a temporary directory just above the kernel source tree where it patches and tests, then replaces the kernel sources from there. In a chroot environment (at least on my systems), this directory is never created, and the directory above the kernel source tree becomes a mess as it fills with the kernel source. I've been remiss and not taken the time to look at the problem sufficiently to identify the root cause. But it's not important if you just back up your kernel source before continuing.

The first command you'll want to use (assuming the kernel source directory is located in your $HOME directory) is:

```
make pending-patches KERNEL_DIR=$HOME/linux
```

You should have no problems with this target. It will tell you what patches it wants to install. You should say “yes” to all these. If, for some reason, any patch doesn't apply (the program may tell you the patch failed to apply), don't worry.

The patch already may be incorporated in the kernel source, but the patch logic was unable to detect it. Just tell the script “no” the second time around. Do *not* force the patch on. Although this is an option, it normally will result in the script aborting. Once pending-patches completes, it will tell you the kernel is ready for compilation. But we're not quite ready yet.

Patches applied or attempted on my system were ipt_LOG.patch (successful) and tos-fix.patch (failed). The tos-fix.patch failed because a fix was applied, but it did not correspond exactly to the patch in the patch-o-matic.

Once you have applied all pending patches to the kernel source, you're ready to take a look at new patches that have not been incorporated into the kernel.

Some time back, Rusty Russell, lead Netfilter developer, introduced the “make patch-o-matic target” to help folks incorporate new things in the kernel without having to know how to use **patch**. This target works fairly well, but don't expect it to work perfectly. Sometimes the patch logic is sufficiently old, and the kernel source sufficiently changed, that a particular patch won't work. In recent months, the patch-o-matic has grown quite a bit and some patches break others. So Rusty incorporated yet another target, “most-of-pom”, to allow new iptables builders to get access to as many of the patches as possible but reduce the possibility of failure.

My recommendation to you is to run **make most-of-pom**, first saying “no” to everything but noting those patches you're interested in. Then run **make patch-o-matic**, noting any new patches not in most-of-pom that you might be interested in. If no new patches interest you in patch-o-matic, stick to most-of-pom. If any new patches do interest you that are only available in patch-o-matic, take careful note of any other patches those new, interesting patches might break. The worst offender as of this writing seems to be the drop-table patch. We'll not look at that patch for this very reason. But if you need it, just read and heed the warnings with that patch and others that tell you they will be broken by it.

In some cases, such as with the H323-conntrack-nat patch, you will not be able to apply a patch to the kernels we use in this article. If you can't do without this particular patch, you probably won't be able to use the experimental make targets for patching (patch-o-matic or most-of-pom). If this is you (I had this need for one customer's system), you need to go to roeder.goe.net/~koepi. The patch there includes newnat5, h323, talk, ftp and irc nat helpers. This is a standard patch applied using the usual patch utility.

While running **make KERNEL_DIR=$HOME/linux patch-o-matic**, I selected the following patches:

```
   NETLINK.patch (successful)
   NETMAP.patch (successful)
   iplimit.patch (successful)
   mport.patch (successful)
   pkttype.patch (successful)
   psd.patch (successful)
   realm.patch (successful)
   snmp-nat.patch (failed: already in kernel)
   string.patch (successful)
   tos-fix.patch (failed: already in kernel)
   ulog.patch (failed: incompatible with kernel
               or previously applied patch)
   LOG.patch.ipv6 (failed: already in kernel)
   REJECT.patch.ipv6 (successful)
```

While going through pending-patches and the patch-o-matic, you'll want to note a few things. The screen is divided by a line. Above the line is a welcome note and a warning. Below the line is the information you should look at.

First, you will have one or more lines that will tell you which patches already have been applied. You'll note as you go through patch-o-matic that it lists the patches applied in pending-patches. In fact, because we're running patch-o-matic, we don't need to run pending-patches; those patches also would have been applied here. You only need to run pending-patches if you don't also run most-of-pom or patch-o-matic, such as if you decided to use the newnat5/h323 patch mentioned above.

Below the already-applied lines comes a line:

```
   Testing... Patchname.patch STATUS (comment)
```

The patchname.patch is the patch being tested. The STATUS will normally be NOT APPLIED. The comment will be one of (x missing files) or (x reject of y hunks). Missing files means the patch hasn't been applied (or the particular corresponding files wouldn't be missing), or the patch doesn't match what's in the kernel sources. In general, the reject means a patch has been applied, it just doesn't match the patch in the patch-o-matic for whatever reason. The most common reason is a small fix was made between the patch in patch-o-matic and the patch in the kernel. When you see reject, it is a foregone conclusion that patch won't apply. Not all patches will work. Note that the ulog.patch failed. This failed either because it was incompatible with a previous patch or with the (changed) kernel sources since the patch was originally created.

Third comes information about the patch, author, status of patch, what the patch is, what it does, often an example to clarify how to use it and perhaps a comment.

Finally, the question, do you want to apply this patch? The choices are No (default), yes, test, force, quit and help as indicated by [N/y/t/f/q/?].

Once we've added the patches we want, we're done. Now the kernel is ready for compilation. Or is it? Well, yes. However, we've added targets to the kernel. I suggest you return to the kernel tree, run **make oldconfig** and select the new Netfilter matches and targets we've incorporated (or what was the sense?). Now you can continue to compile the kernel. After you install the kernel and reboot into it, you're ready to put your new matches and targets to work.

Compiling and Installing iptables

While the kernel source is compiling, there's plenty of time to compile and install iptables. Remember to supply the KERNEL_DIR= (if it's not in /usr/src/linux) and the BINDIR=, LIBDIR= and MANDIR= arguments if you don't want your new binaries and extensions installed in /usr/local/.

One small fix before we start compiling. For whatever reason, the NETLINK extension does not compile. So if you chose the NETLINK.patch (as I did) you need to make a minor adjustment. Just **cd** into the extensions directory and open the Makefile using your favorite text editor. The first line is our shebang line. The second line is blank. The third line starts off PF_EXT_SLIB: and contains various extensions to be made and installed. Add **NETLINK** to the end of the line and save the file back.

Now **cd** back up to the root of the iptables source tree and run your **make** and **make install**, adding the arguments noted above if required.

Some Installation Closing Notes

Above, we used a modified patch process to patch the kernel. If you, like me, grab kernel patches whenever they come out, you'll find that some will no longer apply cleanly because the kernel sources have been modified. So when I do want to try out a new kernel, I save the old .config file, wipe out the old kernel sources and start fresh. You can do that or remember to save a tarball of your kernel source tree before modifying it.

If you built a modular kernel previously using patch-o-matic (or pending-patches or most-of-pom) and are only adding a few more modules, after you use **make oldconfig** to add the new modules, you can do a **make modules; make modules_install** and start using those modules.

If you want to see the information again that you saw while adding the patch-o-matic patches, it's available in the iptables-x.x.x/patch-o-matic/ directory. The files *.patch.help contain the information. In most cases, the examples in these files are duplicated in the kernel configuration help.

The Wall—One Brick at a Time

Now that we have the modules we want compiled and installed, we're ready to put them to work. But before we start, we need to decide exactly what we're going to do. In order to do that we need to lay some groundwork. This groundwork isn't so important when all we have is our home system and we want to let everyone inside out but keep everyone outside out as well. Our state table alone practically assures us that's what we'll have; add masquerading or SNAT and we're done. This is what we had using the basic scripts from the “Taming the Wild Netfilter” article (September 2001 *LJ*).

But firewalls in use at companies are rarely so simple. They demand that we first understand (and maybe even restructure) our network topology. We also need to understand exactly what it is we want our firewall to accomplish. Sometimes, this is not much more than for our home system, but often it is radically different. We can use the company's network security policy to assist us (we do *have* an NSP, don't we?), plus some knowledge of what we want from our network access. We won't discuss risk assessments here [see Mick Bauer's “Practical Threat Analysis and Risk Management” in the January 2002 issue of *LJ*], but their findings should be kept in mind to help guide us in the overall scheme.

Topology, Shmopology—Where Do I Plug in My Laptop?

Many years ago we talked about our internet-connected hosts. They were all directly connected to the Internet. No big deal, as all the system administrators knew each other and things were friendly. Then everyone else discovered the Internet, and we had to make some changes. As things mushroomed out of control, we forgot or never knew who our neighbor system administrators were. We found our systems under attack. So we left our public systems directly connected but started hiding our users' hosts behind packet filters to help protect them. The systems between our router and packet filter were said to be on our DMZ, or demilitarized zone. The rest were on our trusted network behind the packet filter.

Today, few companies would configure their systems this way. In our current situation, usually only honeypots are deliberately left defenseless. Today, the two most common configurations either have a firewall with two internal NICs (one for the trusted network and one for the internal, public access network) or two separate firewalls (the first allowing public traffic into a controlled, but not trusted network and a second permitting entry into our inner sanctum or trusted network).

While small companies may mix the trusted and controlled networks on one private internal network, it is best to keep these separated whenever possible.

You also should control who has access to which area. Firewalls do a lot to keep bad guys out, but do little to protect against bad guys already inside. In fact, you may find it's prudent to put a firewall up between accounting and marketing and engineering and production. None has much business in any of the other's files.

Because this article is principally about iptables, I'll not cover more on network topology. But we needed to understand the above to see how the configurations below work. They really aren't too much different from the point of view of the external firewall, only the internal one(s), if needed, will look a bit more like the basic firewall I presented in the first article. That is, the internal firewalls won't accept new traffic except from the trusted side. What goes out also can be moderated to an extent, and we'll look at that a little bit also.

Running iptables on Nonfirewall Systems

There are times we might want to run iptables on a nonfirewall system. Despite the advice you may have read (as noted in the last paragraph of the “Preparing Your System for an iptables Upgrade” section above), there are times you'll want to run iptables on simple hosts. The simplest, but most common example would be a student system on a university network. In this case, you really should trust no other system. So you'll probably want to accept only related, established traffic.

Another example might be if you have decided to use an XDM server where most users work, but your internet policy only permits certain employees rights to surf the Web. How to deal with this? Well, fortunately, we can deal with this fairly simply with rules like the following:

```
iptables -t filter -I OUTPUT -p tcp
--dport 80 -m owner --uid-owner 500 -j REJECT
iptables -t filter -I OUTPUT -p tcp
--dport 80 -j ACCEPT // only required if OUTPUT
                     // policy is DROP/REJECT
```

or:

```
iptables -t filter -I OUTPUT -p tcp
--dport 80 -m owner --uid-owner 500 -j ACCEPT
iptables -t filter -I OUTPUT -p tcp
--dport 80 -j REJECT // only required if OUTPUT
                     // policy is ACCEPT
```

Naturally, you'd need a list of either those permitted access or those denied. Also, you wouldn't want to write individual rules. I suggest handling the rules like this: for i in **cat surfweb.txt**, do

```
iptables -t filter -I OUTPUT -p tcp
--dport 80 -m owner --uid-owner $i -j REJECT
done
```

Just create a list of users to REJECT (or to ACCEPT and change the rule to match) as the file surfweb.txt. Add user IDs to this list as needed. You might find the above construct valuable for other repetitive rules as well. Note, however, this only prevents them from surfing from the XDM server, not from their local system.

So how might they be stopped from surfing from their local system? Well, the firewall simply could drop or reject packets coming from the disallowed IP. Easy, right? I mean, this is what packet filters are all about. But wait, we're using DHCP and don't necessarily know in advance what the IP will be. Looks like we've outsmarted ourselves—or have we? While we may not know the IP address, one thing we can know is the MAC address. So we get a list of MAC addresses from the systems (or via arp, or from the dhcpd.leases file). Then we use a rule like the following:

```
iptables -t filter -I FORWARD -i eth0 -m mac
--mac-source <MAC> -j ACCEPT
iptables -t filter -A FORWARD -i eth0 -p tcp
--dport 80 -m state --state NEW -j DROP
```

This is best done in a loop like we did earlier, with the MAC addresses in one file and then looping through them.

Note: to use the MAC address to permit or deny systems, remember that they must be on your local network—that is, directly connected, via a hub, to the firewall. If the systems in question are behind an internal firewall, and not connected on the same LAN segment as the external firewall, you must put this rule on the inner firewall.

My point here is crucial: you must know what the system with the rule on it can know about packets it is to control. Only a system where packets originate can know which user ID belongs to the process originating the packets. Only a system on the local LAN segment can know the MAC address of an originating system. After that, we have only the information available in the IP header.

Summary and a Look Ahead

This month we looked at Rusty's patch-o-matic, installing an updated kernel and the user-land iptables utility. Probably the most important part is making sure that if things go wrong you can recover. Meanwhile, Rusty has worked very hard on ensuring you don't need to recover. After that we looked at a couple of common network configurations. You'll need to remember these when you dive into next month's Kernel Korner, which will be a part two of this article. Finally, we took a quick look at how and where iptables might be used in nonfirewall situations to control network resources.

Next month we'll look at managing services behind our NAT-ed firewall, specifically how to make the most use of the IPs your ISP has assigned, and how, with this configuration, to handle services like e-mail properly. We'll also look at more matches, targets, tables and some common errors when building rules.

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

Archive Index Issue Table of Contents

Advanced search

Taking Advantage of Linux Capabilities

Michael Bacarella

Issue #97, May 2002

Concentrating on user privileges to appease the security paranoid.

A common topic of discussion nowadays is security, and for good reason. Security is becoming more important as the world becomes further networked. Like all good systems, Linux is evolving in order to address increasingly important security concerns.

One aspect of security is user privileges. UNIX-style user privileges come in two varieties, user and root. Regular users are absolutely powerless; they cannot modify any processes or files but their own. Access to hardware and most network specifications also are denied. Root, on the other hand, can do anything from modifying all processes and files to having unrestricted network and hardware access. In some cases root can even physically damage hardware.

Sometimes a middle ground is desired. A utility needs special privileges to perform its function, but unquestionable god-like root access is overkill. The ping utility is setuid root simply so it can send and receive ICMP messages. The danger lies in the fact that ping can be exploited before it has dropped its root privileges, giving the attacker root access to your server.

Fortunately, such a middle ground now exists, and it's called POSIX capabilities. Capabilities divide system access into logical groups that may be individually granted to, or removed from, different processes. Capabilities allow system administrators to fine-tune what a process is allowed to do, which may help them significantly reduce security risks to their system. The best part is that your system already supports it. If you're lucky, no patching should be necessary.

A list of all the capabilities that your system is, well, capable of, is available in /usr/include/linux/capability.h, starting with CAP_CHOWN. They're pretty self-explanatory and well commented. Capability checks are sprinkled throughout the kernel source, and grepping for them can make for some fun midnight reading.

Each capability is nothing more than a bit in a bitmap. With 32 bits in a capability set, and 28 sets currently defined, there are currently discussions as to how to expand this number. Some purists believe that additional capabilities would be too confusing, while others argue that there should be many more, even a capability for each system call. Time and Linus will ultimately decide how this exciting feature develops.

The Proc Interface

As of kernel 2.4.17, the file /proc/sys/kernel/cap-bound contains a single 32-bit integer that defines the current global capability set. The global capability set determines what every process on the system is allowed to do. If a capability is stripped from the system, it is impossible for any process, even root processes, to regain them.

For example, many crackers' rootkits (a set of tools that cover up their activities and install backdoors into the system) will load kernel modules that hide illicit processes and files from the system administrator. To counter this, the administrator could simply remove the CAP_SYS_MODULE capability from the system as the last step in the system startup process. This step would prevent any kernel modules from being loaded or unloaded. Once a capability has been removed, it cannot be re-added. The system must be restarted (which means you might have to use the power button if you've removed the CAP_SYS_BOOT capability) to regain the full-capability set.

Okay, I lied. There are two ways to add back a capability:

1. init can re-add capabilities, in theory; there's no actual implementation to my knowledge. This is to facilitate capability-aware systems in the event that init needs to change runlevels.
2. If a process is capable of CAP_SYS_RAWIO, it can modify kernel memory through /dev/mem. Among other things, it can modify kernel memory to grant itself whatever access it desires. Remove CAP_SYS_RAWIO, but be careful: by removing CAP_SYS_RAWIO, programs such as X most likely will fail to run.

Editing cap-bound by hand is kind of tedious. Fortunately for you, there's a utility called lcap that provides a friendlier interface to cap-bound. Here's how one would remove CAP_SYS_CHOWN:

```
lcap CAP_SYS_CHOWN
```

Once done, it becomes impossible to change a file's owner:

```
chown nobody test.txt
chown: changing ownership of `test.txt':
       Operation not permitted
```

Here's how you would remove all capabilities except CAP_SYS_BOOT, CAP_SYS_KILL and CAP_SYS_NICE:

```
lcap -z CAP_SYS_BOOT CAP_SYS_KILL CAP_SYS_NICE
```

One thing to note: modifying cap-bound restricts the capabilities of future processes only. Okay, not exactly future processes but any process that calls exec(2) (see the function compute_creds in the kernel source file fs/exec.c). Currently running processes keep the capabilities with which they started.

Modifying the capabilities of an existing process leads us into the next section, and here's the catch I spoke about above. Running lcap with no arguments lists what your system is capable of. If you see that CAP_SETPCAP is disabled, you need to make a change to your kernel. It's simple enough to describe here. In the kernel source tree, edit include/linux/capability.h. You're changing the lines:

```
#define CAP_INIT_EFF_SET
to_cap_t(~0 & ~CAP_TO_MASK(CAP_SETPCAP))
#define CAP_INIT_INH_SET  to_cap_t(0)
```

so that they read:

```
#define CAP_INIT_EFF_SET  to_cap_t(~0)
#define CAP_INIT_INH_SET  to_cap_t(~0)
```

and then recompile.

There's actually a reason that CAP_SETPCAP is disabled by default: it's deemed a security risk to leave it enabled on a production system (a patch exists for this condition but has yet to be applied as of this writing). To be on the safe side, make sure to remove this capability when you're done playing.

The System Call Interface

As of this writing, the syscalls capset and capget manipulate capabilities for a process. There are no guarantees that this interface won't change. Portable applications are encouraged to use libcap (www.kernel.org/pub/linux/libs/security/linux-privs/kernel-2.4) instead.

The prototype for capset is

```
int capset(cap_user_header_t header,
const cap_user_data_t data);
```

HEADER is a fancy way to say which pid you're operating on:

```
typedef struct __user_cap_header_struct {
        __u32 version;
        int pid;
} *cap_user_header_t;
```

If pid is -1, you will modify the capabilities of all currently running processes. Less than -1 and you modify the process group equal to pid times -1. The semantics are similar to those of kill(2).

The DATA argument allows you to choose which capability sets you plan to modify. There are three:

```
typedef struct __user_cap_data_struct {
        __u32 effective;
        __u32 permitted;
        __u32 inheritable;
} *cap_user_data_t;
```

The permitted set contains all of the capabilities that a process is ultimately capable of realizing.

The effective set is the capabilities a process has elected to utilize from its permitted set. It's as if you had a huge arsenal of poetry (permitted set) but chose only to arm yourself with Allen Ginsberg for the task at hand (effective set).

The inheritable set defines which capabilities may be passed on to any programs that replace the current process image via exec(2). Please note that fork(2) does nothing special with capabilities. The child simply receives an exact copy of all three capabilities sets.

Only capabilities in the permitted set can be added to the effective or inheritable set. Capabilities cannot be added to the permitted set of a process unless CAP_SETPCAP is set.

The Filesystem Interface

Sadly, capabilities still lack filesystem support, limiting their usefulness to a degree. Someday, the mainstream kernels will allow you to enable capabilities in a program's inode, obviating the setuid bit in many system utilities.

Once fully supported, permitting the ping utility to open raw sockets could be as simple as:

```
chattr +CAP_NET_RAW /bin/ping
```

Unfortunately, more pressing kernel issues have delayed work in this area.

If you're so inclined, you can use libcap to hack your favorite services so that they are capability-aware and drop the privileges they no longer need at startup. Several patches exist for xntpd that do just this; some even provide their modified version as an RPM. Try a Google search if you're interested in a capability-aware version of some root-level process you find yourself often shaking a fist at.

setpcap can be used to modify the capability set of an existing process. For example, if the PID of a regular user's shell is 4235, here's how you can give that user's shell the ability to send signals to any process:

```
setpcaps 'cap_kill=ep' 4235
```

An example use of this would be to allow a friend who is using your machine to debug a CGI script to kill any Apache processes that get stuck in infinite loops. You'd run it against their login shell once and forget about them.

Here's an example that utilizes execcap and sucap to run ping as the user “nobody”, with only the CAP_NET_RAW capability. Our target of choice for ping is www.yahoo.com:

```
execcap 'cap_net_raw=ep' /sbin/sucap nobody
nobody /bin/ping www.yahoo.com
```

This sample isn't terribly useful because you need to be root to execute it, but it does illustrate what is possible. Despite some of these shortcomings, system administrators still can take measures to increase the security of their system. A system without CAP_SYS_BOOT, CAP_SYS_RAWIO and CAP_SYS_MODULE is extremely difficult for an intruder to modify. They cannot hack kernel memory, install new modules or restart the system so that it runs a backdoored kernel.

If your system logs are append-only and your core system utilities immutable (see chattr(3) for details), removing the CAP_LINUX_IMMUTABLE capability will make it virtually impossible for intruders to erase their tracks or install compromised utilities. Traffic sniffers like tcpdump become unusable once CAP_NET_RAW is removed. Remove CAP_SYS_PTRACE and you've turned off program debugging. Such a hostile environment is a script kiddy's worst nightmare, and there is no choice but to disconnect and wait for the intrusion to be discovered.

Conclusion

Capabilities can provide sophisticated, fine-grained access control over all aspects of a Linux system. At last, security paranoids will have some tools they so desperately need in their endless fight against “them”.

Resources

Michael Bacarella (mike@bacarella.com) is president of Netgraft Corporation, a firm specializing in web system development and information security analysis. He shares an apartment in New York with his wonderful fiancée and a most fearsome green iguana (the iguana's name is Kang.

Debugging Kernel Modules with User Mode Linux

David Frascone

Issue #97, May 2002

Programming in kernel space has always been left to the gurus. Few people have the courage, knowledge and patience to work in the realm of interrupts, devices and the always painful kernel panic.

When you write programs in user space, the worst thing that can happen to your program is a core dump. Your program did something very wrong, so the operating system decided to give you all of its memory and state information back to you in the form of a core file. Core files can then be used to debug your program and fix the problem.

When you program in the kernel, there is no operating system to step in and safely stop your code from running and tell you that you have a problem. The Linux kernel is pretty nice to its own code. Sometimes it can survive a panic, if you are doing something wrong that is relatively benign (these panics are typically called oopses). But, there is nothing to stop your code from overwriting or accessing memory locations from anywhere in the kernel's address space. Also, if your module hangs, the kernel hangs (technically, your current kernel thread hangs, but the result is usually the same).

These problems may sound benign to the naïve, but they are serious issues. If the kernel panics, you rarely know exactly what caused the panic. The typical solution is to put printks everywhere and hope that you stumble across the problem before the messages are lost to the reboot. All of this is assuming that you do not corrupt your filesystem. I have lost an entire filesystem before due to a poorly timed panic (and due to the fact that a badly initialized pointer was overwriting some of ext2's internal structures).

The first thing you learn when kernel programming is to keep all your code on NFS. Files remain safe on another machine. But, that does not save you the time of having e2fsck run every time you panic. Plus, you still can lose your filesystem, even if your source code is safe on another machine.

So, with all of these issues, it is not surprising how few have entered the realm of kernel programming. Now, all that can change.

Virtual Machines and UML

Back in the mainframe days, when timesharing machines were the norm, the idea of a virtual machine was born. A virtual machine is an encapsulated computer completely at your disposal. A program on a virtual machine has no real access to the physical hardware. All hardware access is controlled by the machine or emulator.

VMware (www.vmware.com) has a very powerful virtual machine that allows you to run any x86-based operating system under Windows NT, 2000, XP or Linux. SoftPC (an 8086 emulator allowing you to run Windows and DOS programs) has been available on Motorola 68k-based computers (i.e., the Macintosh) since 1988.

True virtual machines are sometimes too expensive for the learner's budget. (VMware Workstation for Linux costs $299 US from their web site.) Thankfully, there is now a free alternative for those only wanting to run Linux: User-Mode Linux (UML).

User-Mode Linux (user-mode-linux.sourceforge.net) is not a complete virtual machine. It does not emulate different hardware or give you the ability to run other operating systems. But, it *does* allow you to run a kernel in user space. This gives you several benefits when it comes to development: the host filesystem is safe from corruption, the virtual filesystem is undoable (which makes it safe from corruption), you can run multiple machines on one machine (this is useful for testing intermachine communication, i.e., network messages, without having to use multiple machines) and it is very easy to run the kernel in a debugger.

Setting up UML

Running UML is easy. You can download one of the binary packages (kernel binaries, plus a couple of tools), or you can download the kernel patch. You also need to download a filesystem. I'd recommend playing with the binaries first, then building a custom kernel to suit your needs. The HOWTO covers all of these topics and more.

One useful benefit of UML is Copy-on-Write files. These files allow you to modify a virtual filesystem, without modifying the base filesystem. All writes or modifications to a filesystem are stored in these files, typically ending with the extension .cow.

So, when you are working, and you panic the filesystem, all you do is remove the .cow file (which will be recreated), and your corrupted filesystem is restored to its pristine version. (There are also tools to incorporate the changes in a .cow file back into the original filesystem, if you want to keep your changes.)

Debugging Modules

Once you have UML up and running, it's time to play. I've written a very simple kernel module for testing. It uses four devices, /dev/gentest[0-3]. The module treats each device a little differently. Device 1 is a sink (just like /dev/null). Device 2 stores a string for later retrieval. You can read the status of the module from device 3, and device 0 could be any of the other three devices, depending on how it is configured. (You can change the configuration with ioctl calls.) The kernel module is available from www.frascone.com/kHacking/gentest-0.1.tar.gz.

Debugging with printk

So, let's make a bug—a nasty one. Let's say when someone opens device 4 (**cat /dev/gentest4**), the module hangs in a nasty loop: **for(;;) i++;** (see Listing 1). Deadlocks or hangs are common errors when writing programs. They are sometimes hard to find. Typically programmers just use printks to locate the errors: **printk("Got here!\n");**. This type of debugging works, but you still hang the system several times before you find the problem. With constant fscks, it can get ugly. But, with UML, you just add in the printks and reboot to a fresh filesystem every time to test it.

Listing 1. Test Bug

UML will help us find that bug with printks, but it is nothing that would have caused us more than a few reboots. Now let's make our first really nasty bug. Let's say that when someone reads from device 5 (i.e., **cat /dev/gentest5**); the module starts to overwrite all memory: **memset(0, 0, 0xffffffff);** (see Listing 2). Overwriting memory is a common error in C programs. In the kernel it is especially nasty and can sometimes cause an instant reboot, keeping you from seeing any printks that are generated. These bugs can still be isolated with printk, but it is a very time-consuming process.

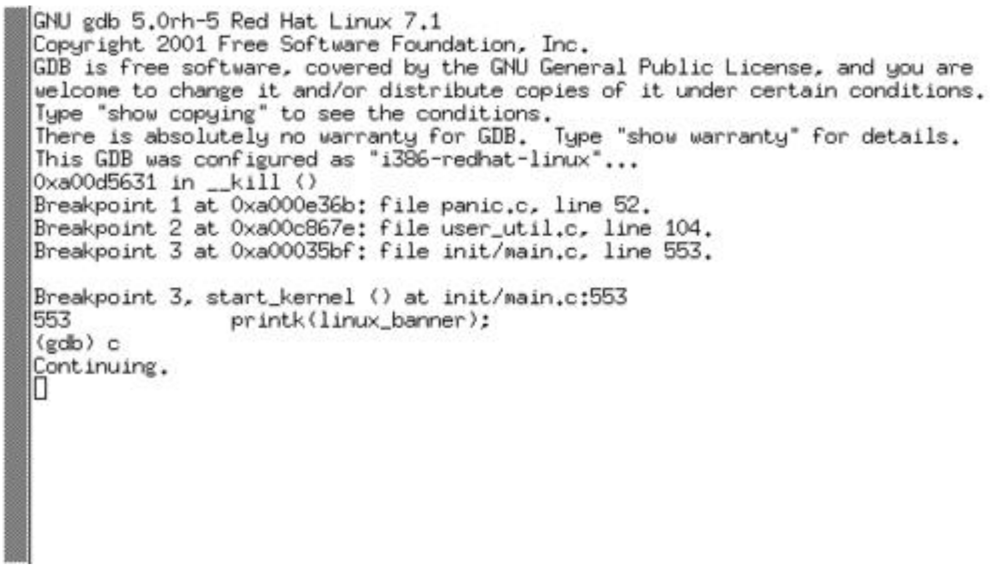
Listing 2. A Really Nasty Bug

Debugging with GDB

From what I've covered so far, UML is a great debugging tool. You can use it to keep your filesystem safe when debugging modules. But there's something more: GDB.

As most experienced kernel programmers know, there is already a way to debug a kernel using GDB and the serial line. But, in my experience, it really doesn't work very well. The GDB shim in the kernel sometimes hangs, and you need two machines to make it work. I have successfully debugged kernels running in VMware on one machine by redirecting the virtual machine's serial port to a file, but it was slow going, since the kernel portion of the GDB code could still sometimes hang.

UML makes all that a thing of the past. With UML, you can run the entire virtual machine under GDB, attach to a kernel while it's running, or even after a panic. The easiest way to run UML under GDB is to add the command-line flag **debug** to your runline. UML will then spawn GDB in an xterm for you and stop the kernel. For most purposes, just type **c** to allow the kernel to continue booting up (see Figure 1).



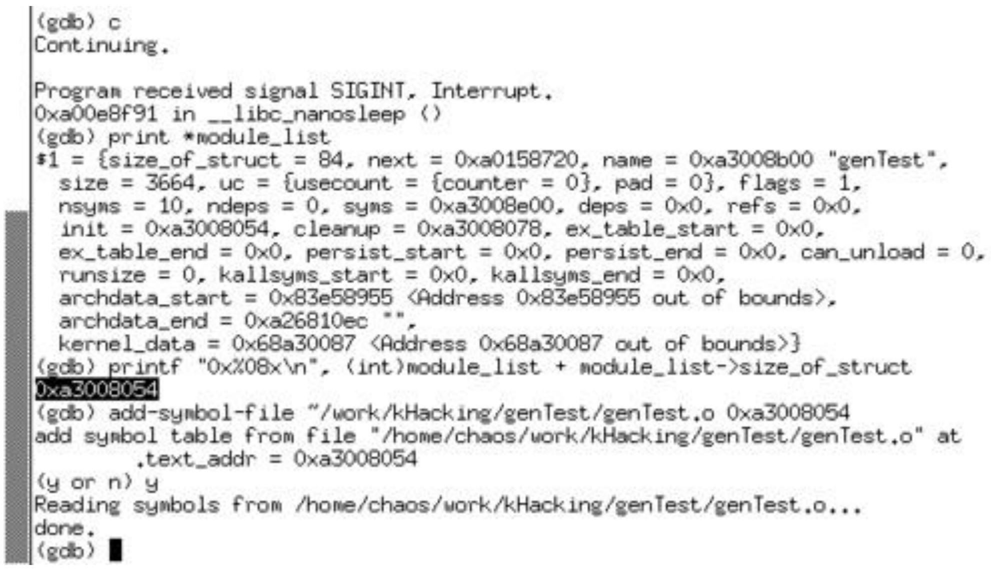
```
GNU gdb 5.0rh-5 Red Hat Linux 7.1
Copyright 2001 Free Software Foundation, Inc.
GDB is free software, covered by the GNU General Public License, and you are
welcome to change it and/or distribute copies of it under certain conditions.
Type "show copying" to see the conditions.
There is absolutely no warranty for GDB.  Type "show warranty" for details.
This GDB was configured as "i386-redhat-linux"...
0xa00d5631 in __kill ()
Breakpoint 1 at 0xa000e36b: file panic.c, line 52.
Breakpoint 2 at 0xa00c867e: file user_util.c, line 104.
Breakpoint 3 at 0xa00035bf: file init/main.c, line 553.

Breakpoint 3, start_kernel () at init/main.c:553
553             printk(linux_banner);
(gdb) c
Continuing.
```

Figure 1. Running UML under GDB

To debug the module, you first have to load the module, then tell GDB where the symbol file is, then set any breakpoints you need.

So, first things first, load the module. Included in the source code is a simple shell script called loadModule that loads the module and creates the devices if they do not already exist.

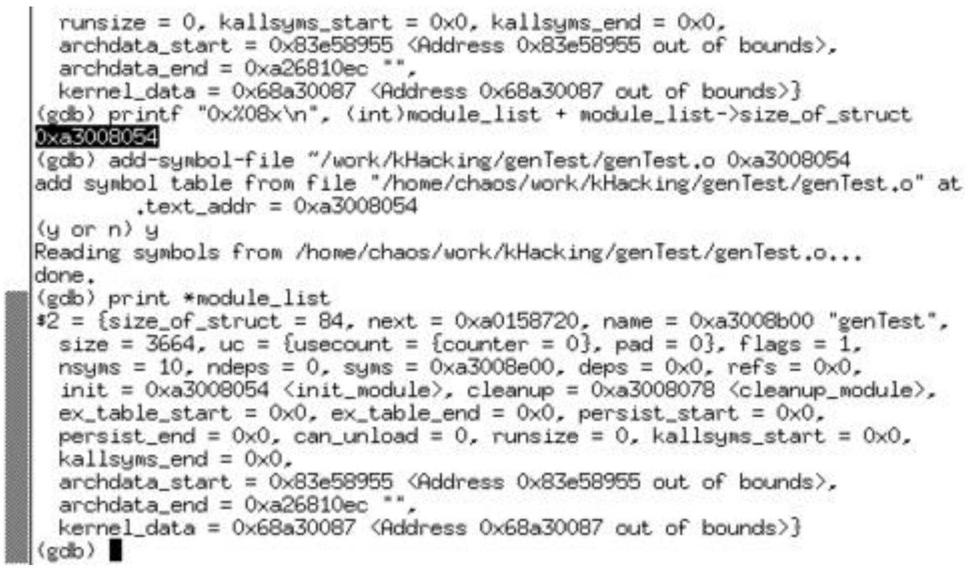
Once the module is loaded, press Ctrl-C inside the GDB window to pause the kernel, and look at the module_list pointer. The last module loaded should be at the head of the list. You can use a simple printf command to get the address of the module. You'll need it when loading the symbol file (see Figure 2).



```
(gdb) c
Continuing.

Program received signal SIGINT, Interrupt.
0xa00e8f91 in __libc_nanosleep ()
(gdb) print *module_list
$1 = {size_of_struct = 84, next = 0xa0158720, name = 0xa3008b00 "genTest",
  size = 3664, uc = {usecount = {counter = 0}, pad = 0}, flags = 1,
  nsyms = 10, ndeps = 0, syms = 0xa3008e00, deps = 0x0, refs = 0x0,
  init = 0xa3008054, cleanup = 0xa3008078, ex_table_start = 0x0,
  ex_table_end = 0x0, persist_start = 0x0, persist_end = 0x0, can_unload = 0,
  runsize = 0, kallsyms_start = 0x0, kallsyms_end = 0x0,
  archdata_start = 0x83e58955 <Address 0x83e58955 out of bounds>,
  archdata_end = 0xa26810ec "",
  kernel_data = 0x68a30087 <Address 0x68a30087 out of bounds>}
(gdb) printf "0x%08x\n", (int)module_list + module_list->size_of_struct
0xa3008054
(gdb) add-symbol-file ~/work/kHacking/genTest/genTest.o 0xa3008054
add symbol table from file "/home/chaos/work/kHacking/genTest/genTest.o" at
        .text_addr = 0xa3008054
(y or n) y
Reading symbols from /home/chaos/work/kHacking/genTest/genTest.o...
done.
(gdb) █
```

Figure 2. Module List

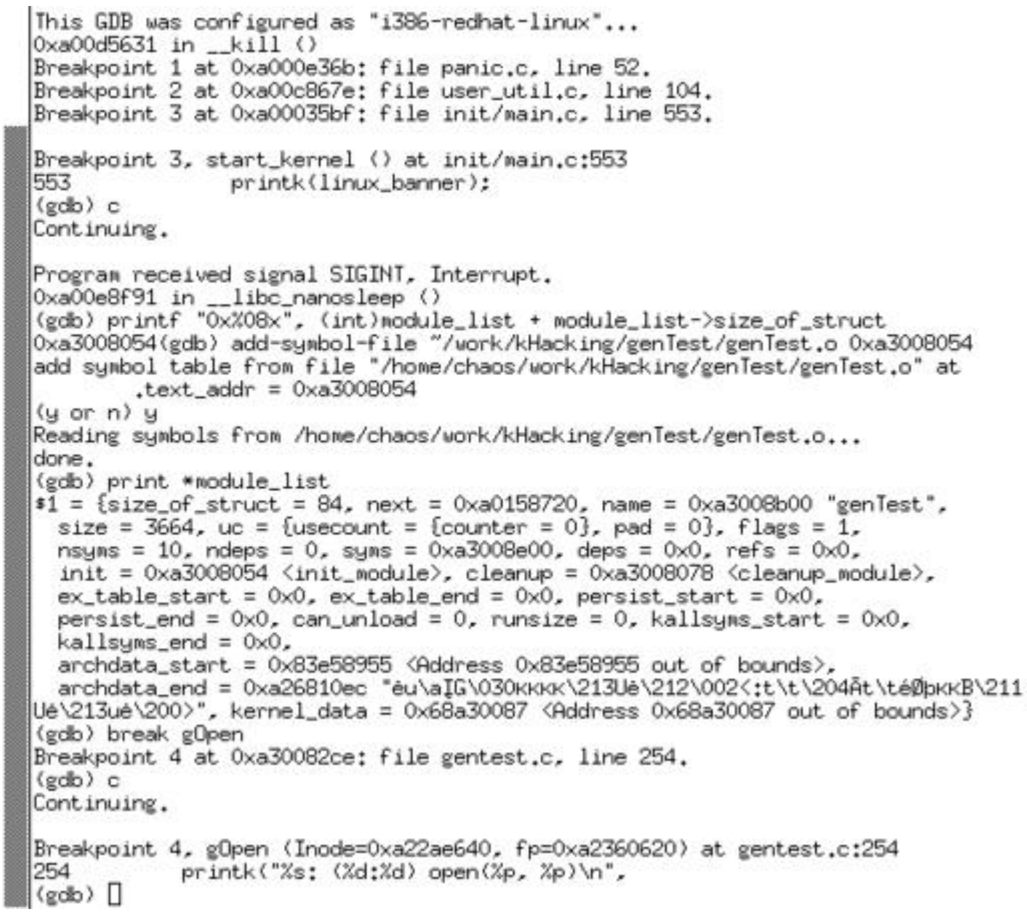
Now, load the symbols file with the command **add-symbol-file MODULE_PATH ADDRESS**. The filename used is the filename on the host system, *not* on the virtual machine. After answering “y” to an “Are you sure you know what you're doing?” question, the symbol file is loaded. You can check that it has been loaded correctly by re-examining the module_list pointer again. Notice that now the init and cleanup pointers have the appropriate function names associated with their addresses (see Figure 3).



```
  runsize = 0, kallsyms_start = 0x0, kallsyms_end = 0x0,
  archdata_start = 0x83e58955 <Address 0x83e58955 out of bounds>,
  archdata_end = 0xa26810ec "",
  kernel_data = 0x68a30087 <Address 0x68a30087 out of bounds>}
(gdb) printf "0x%08x\n", (int)module_list + module_list->size_of_struct
0xa3008054
(gdb) add-symbol-file ~/work/kHacking/genTest/genTest.o 0xa3008054
add symbol table from file "/home/chaos/work/kHacking/genTest/genTest.o" at
        .text_addr = 0xa3008054
(y or n) y
Reading symbols from /home/chaos/work/kHacking/genTest/genTest.o...
done.
(gdb) print *module_list
$2 = {size_of_struct = 84, next = 0xa0158720, name = 0xa3008b00 "genTest",
  size = 3664, uc = {usecount = {counter = 0}, pad = 0}, flags = 1,
  nsyms = 10, ndeps = 0, syms = 0xa3008e00, deps = 0x0, refs = 0x0,
  init = 0xa3008054 <init_module>, cleanup = 0xa3008078 <cleanup_module>,
  ex_table_start = 0x0, ex_table_end = 0x0, persist_start = 0x0,
  persist_end = 0x0, can_unload = 0, runsize = 0, kallsyms_start = 0x0,
  kallsyms_end = 0x0,
  archdata_start = 0x83e58955 <Address 0x83e58955 out of bounds>,
  archdata_end = 0xa26810ec "",
  kernel_data = 0x68a30087 <Address 0x68a30087 out of bounds>}
(gdb) █
```

Figure 3. Loading the Symbol File

Now that the module is loaded, you can set any breakpoints you want. I'll set a breakpoint at open and then try to cat one of the devices (see Figure 4).

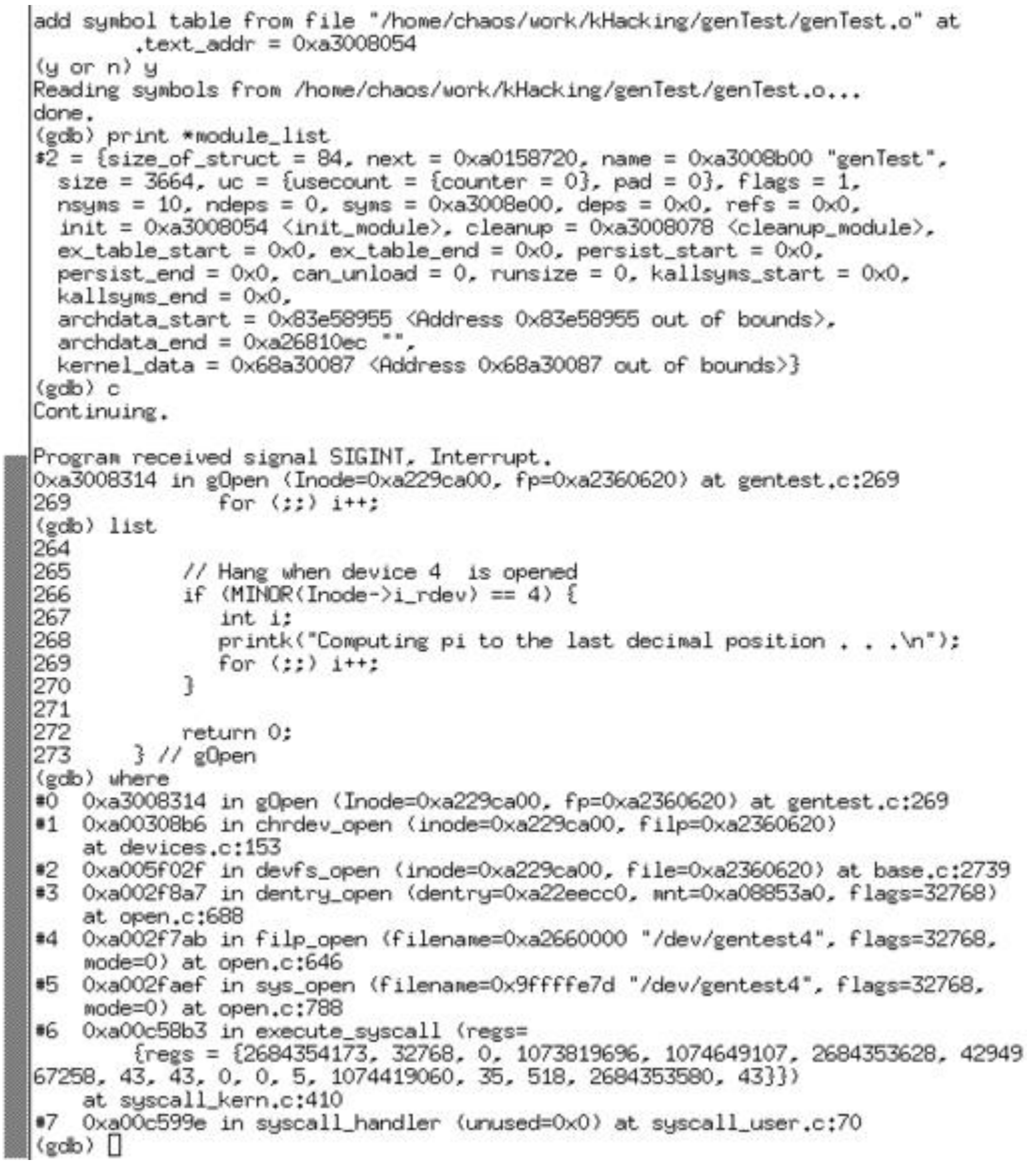


```
This GDB was configured as "i386-redhat-linux"...
0xa00d5631 in __kill ()
Breakpoint 1 at 0xa000e36b: file panic.c, line 52.
Breakpoint 2 at 0xa00c867e: file user_util.c, line 104.
Breakpoint 3 at 0xa00035bf: file init/main.c, line 553.

Breakpoint 3, start_kernel () at init/main.c:553
553             printk(linux_banner);
(gdb) c
Continuing.

Program received signal SIGINT, Interrupt.
0xa00e8f91 in __libc_nanosleep ()
(gdb) printf "0x%08x", (int)module_list + module_list->size_of_struct
0xa3008054(gdb) add-symbol-file ~/work/kHacking/genTest/genTest.o 0xa3008054
add symbol table from file "/home/chaos/work/kHacking/genTest/genTest.o" at
        .text_addr = 0xa3008054
(y or n) y
Reading symbols from /home/chaos/work/kHacking/genTest/genTest.o...
done.
(gdb) print *module_list
$1 = {size_of_struct = 84, next = 0xa0158720, name = 0xa3008b00 "genTest",
  size = 3664, uc = {usecount = {counter = 0}, pad = 0}, flags = 1,
  nsyms = 10, ndeps = 0, syms = 0xa3008e00, deps = 0x0, refs = 0x0,
  init = 0xa3008054 <init_module>, cleanup = 0xa3008078 <cleanup_module>,
  ex_table_start = 0x0, ex_table_end = 0x0, persist_start = 0x0,
  persist_end = 0x0, can_unload = 0, runsize = 0, kallsyms_start = 0x0,
  kallsyms_end = 0x0,
  archdata_start = 0x83e58955 <Address 0x83e58955 out of bounds>,
  archdata_end = 0xa26810ec "èu\aĮG\030κκκκ\213Uè\212\002<:t\t\204Āt\téØpκκB\211
Uè\213uè\200>", kernel_data = 0x68a30087 <Address 0x68a30087 out of bounds>}
(gdb) break gOpen
Breakpoint 4 at 0xa30082ce: file gentest.c, line 254.
(gdb) c
Continuing.

Breakpoint 4, gOpen (Inode=0xa22ae640, fp=0xa2360620) at gentest.c:254
254             printk("%s: (%d:%d) open(%p, %p)\n",
(gdb) ▯
```

Figure 4. Setting Breakpoints

Now, let's run our two tests and see how hard the bugs are to find when using GDB. On the first test, the system still hangs. But, now we can press Ctrl-C in the debugger and see where it is hung.

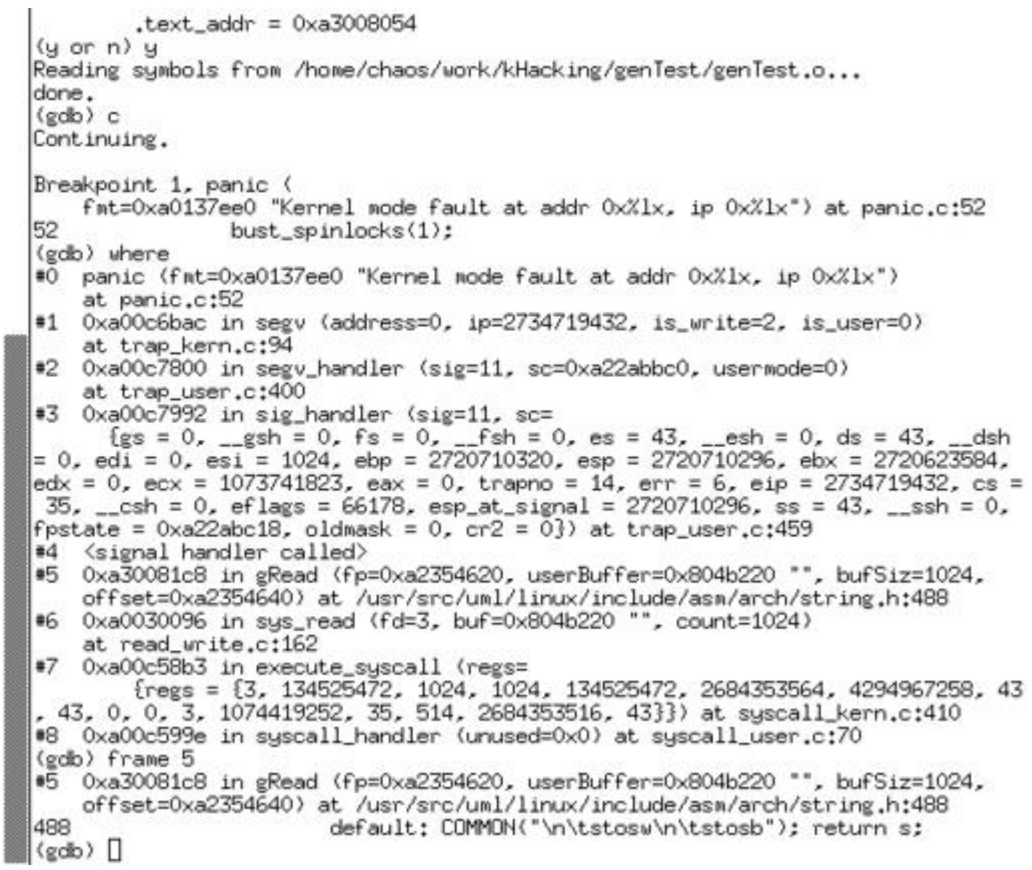
In the hang test (see Figure 5) it is obvious that the current stopping point is inside the for loop. If we really want to have fun, we can print out the value of i to see what it contains.



```
add symbol table from file "/home/chaos/work/kHacking/genTest/genTest.o" at
        .text_addr = 0xa3008054
(y or n) y
Reading symbols from /home/chaos/work/kHacking/genTest/genTest.o...
done.
(gdb) print *module_list
$2 = {size_of_struct = 84, next = 0xa0158720, name = 0xa3008b00 "genTest",
  size = 3664, uc = {usecount = {counter = 0}, pad = 0}, flags = 1,
  nsyms = 10, ndeps = 0, syms = 0xa3008e00, deps = 0x0, refs = 0x0,
  init = 0xa3008054 <init_module>, cleanup = 0xa3008078 <cleanup_module>,
  ex_table_start = 0x0, ex_table_end = 0x0, persist_start = 0x0,
  persist_end = 0x0, can_unload = 0, runsize = 0, kallsyms_start = 0x0,
  kallsyms_end = 0x0,
  archdata_start = 0x83e58955 <Address 0x83e58955 out of bounds>,
  archdata_end = 0xa26810ec "",
  kernel_data = 0x68a30087 <Address 0x68a30087 out of bounds>}
(gdb) c
Continuing.

Program received signal SIGINT, Interrupt.
0xa3008314 in gOpen (Inode=0xa229ca00, fp=0xa2360620) at gentest.c:269
269             for (;;) i++;
(gdb) list
264
265         // Hang when device 4  is opened
266         if (MINOR(Inode->i_rdev) == 4) {
267             int i;
268             printk("Computing pi to the last decimal position . . .\n");
269             for (;;) i++;
270         }
271
272         return 0;
273     } // gOpen
(gdb) where
#0  0xa3008314 in gOpen (Inode=0xa229ca00, fp=0xa2360620) at gentest.c:269
#1  0xa00308b6 in chrdev_open (inode=0xa229ca00, filp=0xa2360620)
    at devices.c:153
#2  0xa005f02f in devfs_open (inode=0xa229ca00, file=0xa2360620) at base.c:2739
#3  0xa002f8a7 in dentry_open (dentry=0xa22eecc0, mnt=0xa08853a0, flags=32768)
    at open.c:688
#4  0xa002f7ab in filp_open (filename=0xa2660000 "/dev/gentest4", flags=32768,
    mode=0) at open.c:646
#5  0xa002faef in sys_open (filename=0x9ffffe7d "/dev/gentest4", flags=32768,
    mode=0) at open.c:788
#6  0xa00c58b3 in execute_syscall (regs=
        {regs = {2684354173, 32768, 0, 1073819696, 1074649107, 2684353628, 42949
67258, 43, 43, 0, 0, 5, 1074419060, 35, 518, 2684353580, 43}})
    at syscall_kern.c:410
#7  0xa00c599e in syscall_handler (unused=0x0) at syscall_user.c:70
(gdb) 
```

Figure 5. Hang Test

Now, the memory overwrite is a bit more difficult. Not because it is a panic, but because I used memset. **memset**, in the GNU libc, ends up inserting inline assembly into your code, so it looks like your bug is in string.h, instead of your module. But, it still lets you know which function the error occurred in, and you still know it is inside of a memset (see Figure 6).



```
        .text_addr = 0xa3008054
(y or n) y
Reading symbols from /home/chaos/work/kHacking/genTest/genTest.o...
done.
(gdb) c
Continuing.

Breakpoint 1, panic (
    fmt=0xa0137ee0 "Kernel mode fault at addr 0x%lx, ip 0x%lx") at panic.c:52
52              bust_spinlocks(1);
(gdb) where
#0  panic (fmt=0xa0137ee0 "Kernel mode fault at addr 0x%lx, ip 0x%lx")
    at panic.c:52
#1  0xa00c6bac in segv (address=0, ip=2734719432, is_write=2, is_user=0)
    at trap_kern.c:94
#2  0xa00c7800 in segv_handler (sig=11, sc=0xa22abbc0, usermode=0)
    at trap_user.c:400
#3  0xa00c7992 in sig_handler (sig=11, sc=
      {gs = 0, __gsh = 0, fs = 0, __fsh = 0, es = 43, __esh = 0, ds = 43, __dsh
= 0, edi = 0, esi = 1024, ebp = 2720710320, esp = 2720710296, ebx = 2720623584,
edx = 0, ecx = 1073741823, eax = 0, trapno = 14, err = 6, eip = 2734719432, cs =
 35, __csh = 0, eflags = 66178, esp_at_signal = 2720710296, ss = 43, __ssh = 0,
fpstate = 0xa22abc18, oldmask = 0, cr2 = 0}) at trap_user.c:459
#4  <signal handler called>
#5  0xa30081c8 in gRead (fp=0xa2354620, userBuffer=0x804b220 "", bufSiz=1024,
    offset=0xa2354640) at /usr/src/uml/linux/include/asm/arch/string.h:488
#6  0xa0030096 in sys_read (fd=3, buf=0x804b220 "", count=1024)
    at read_write.c:162
#7  0xa00c58b3 in execute_syscall (regs=
      {regs = {3, 134525472, 1024, 1024, 134525472, 2684353564, 4294967258, 43
, 43, 0, 0, 3, 1074419252, 35, 514, 2684353516, 43}}) at syscall_kern.c:410
#8  0xa00c599e in syscall_handler (unused=0x0) at syscall_user.c:70
(gdb) frame 5
#5  0xa30081c8 in gRead (fp=0xa2354620, userBuffer=0x804b220 "", bufSiz=1024,
    offset=0xa2354640) at /usr/src/uml/linux/include/asm/arch/string.h:488
488                     default: COMMON("\n\tstosw\n\tstosb"); return s;
(gdb) []
```

Figure 6. Memory Overwrite

Also, you still can examine any local variables in the current function (gRead) or any global variables to help you find the problem.

Conclusion

While UML might not let you debug a device driver (since UML does not have access to the physical hardware on the machine), it is an invaluable aid in debugging kernel modules. It allows you to write and debug kernel modules as easily as other C programs, without fear of panics, deadlocks and data loss. It is a useful addition to any kernel hacker's toolbelt.

Resources

David Frascone (dave@frascone.com) currently works for the SunLabs division of Sun Microsystems, Inc. His current project is a Diameter (AAA) implementation. He is active in the IETF's AAA working group.

Crystal Space: an Open-Source 3-D Graphics Engine

Howard Wen

Issue #97, May 2002

Howard provides an introduction to Cystal Space, and open-source alternative to commercial 3-D graphics engines.

Want to make a 3-D graphics game or application? First, you will need a 3-D graphics software engine on which to build it. Traditionally, your choices have been limited to programming your own from scratch or paying a high licensing fee to use another company's, which can restrict what you do with your final product commercially.

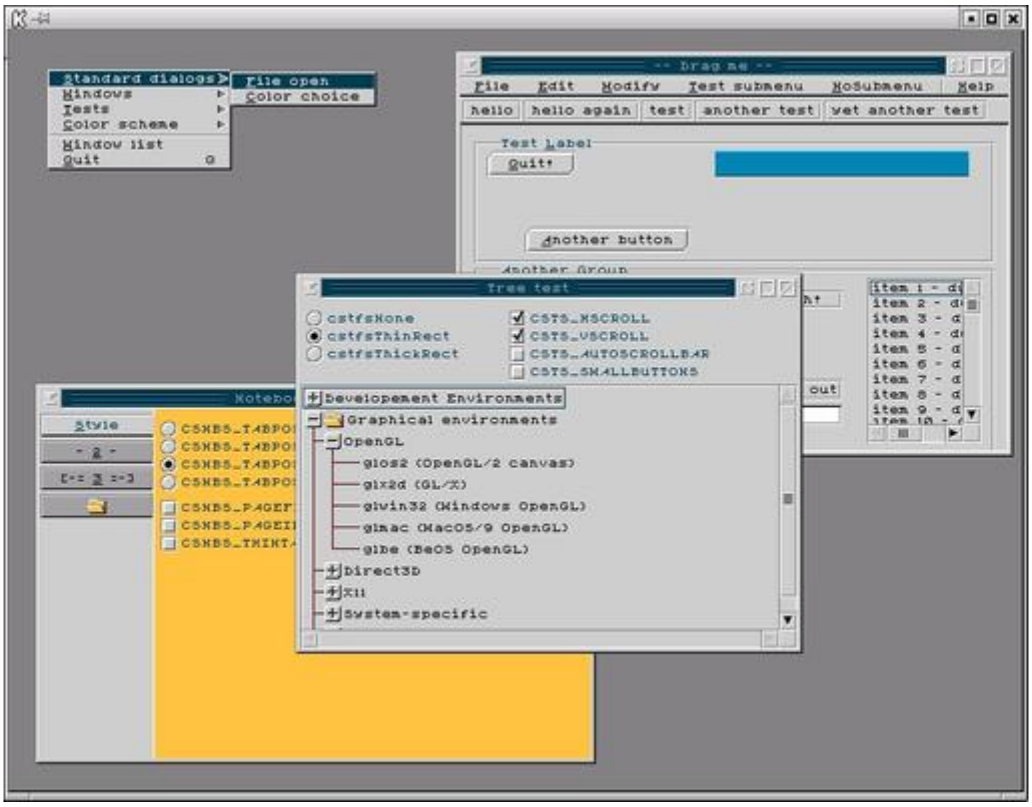
However, a third alternative exists: Crystal Space, an open-source 3-D graphics engine created by 31-year-old Belgian programmer, Jorrit Tyberghein.

Crystal Space™

An Indoor Environment Rendered with Crystal Space

Tyberghein created Crystal Space in 1997 after seeing games like *Doom* and *Quake* and wondering how they were made. With no prior experience in graphics coding, he researched the Internet on 3-D graphics programming and put together the first version of Crystal Space in two months. Tyberghein released the code as open source, and the Crystal Space development community soon was born. The graphics engine has since been ported from its original Linux code to UNIX, Windows 32-bit and NT and other OSes.

Most of the software currently being made with Crystal Space are role-playing games. The fact that the engine includes full control of one or more camera perspectives and has built-in support for networking makes it easier to develop multiplayer games, like role players. There are also flight simulators, real-time strategy games and first-person shooters, similar to *Doom* or *Quake*, under development using the engine.



A Test Application Showing Several Features of the Crystal Space

As for how Crystal Space compares to commercial 3-D graphics engines, specifically those that power games like *Quake III* and *Unreal Tournament*, presently it is comparable in terms of specific graphics features but not in regard to code readiness and performance speed. But in general, Crystal Space is much richer because it is not just a game-specific engine. It is also a general-purpose graphics API, and it is being put to use to drive applications like a multimedia internet browser, a sound-editing program and an image viewer.

Andrew Zabolotny, a 28-year-old programmer from St. Petersburg, Russia, who has been contributing to Crystal Space's high-level coding says: “We have lots of things *Quake* or *Unreal* will never need because a first-person shooter is a very specific type of game that needs just a limited set of features.”

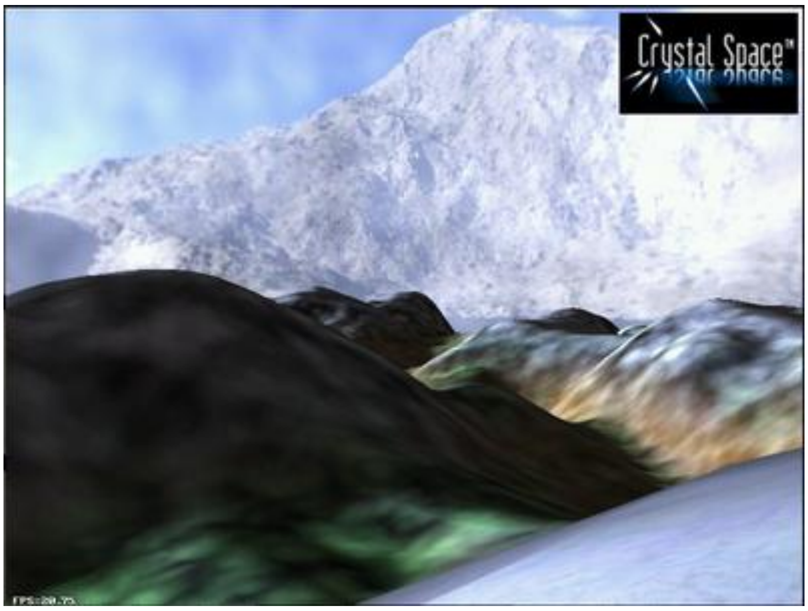
Crystal Space's Features: the Good and the Bad

Developers who have been working with Crystal Space cite the engine's stability (it rarely crashes) and the completeness of its 3-D-rendering features as reasons for using it and contributing to its development.

Yet how necessary is a 3-D graphics engine these days, considering the advanced and powerful 3-D graphics chipsets that have become a standard feature in most PCs? The simple answer is that even with the best 3-D technology, you still need a software engine to manage what the hardware is doing.

“There is always a limit to what hardware can do”, says Tyberghein, Crystal Space's creator. “Even the best hardware cannot handle millions or more polygons just by rendering them all. So an engine still has to do some optimization to limit what is sent to the hardware.”

For example, a software engine like Crystal Space is needed to determine quickly which parts of a game's virtual environment are visible to the user. This helps to maximize the performance of the 3-D chipset by not wasting its resources in making it render an object that will not actually appear on the monitor, even if it is supposed to exist (though off-camera, away from the user's point of view) within the virtual environment.



Example Environment Created with Crystal Space's Landscape

Crystal Space is free under LGPL. This 3-D game development kit written in C++ supports numerous graphics features and fancy effects: true six degrees of freedom; colored lighting; mipmapping; portals; mirrors; alpha transparency; reflective surfaces; frame-based and skeletal-animated 3-D sprites; procedural textures; radiosity; particle systems; halos; volumetric fog; scripting using

Python or other languages; 8-bit, 16-bit and 32-bit display support; Direct3D, OpenGL and software graphics rendering; font support; and hierarchical transformations, to list only a few.

Although it cannot match graphics engines like *Quake III*'s or *Unreal Tournament*'s in performance, Crystal Space has its own advantages over them: for one, it is cross-platform, so you can write code that will run equally well on Linux, UNIX, Windows 32-bit, Windows NT and seven other OSes to which the engine has been ported.

Crystal Space has a flexible plugin system so that a single executable works with various renderers like OpenGL, Direct3D and Glide. Since the first release of the engine, the OpenGL renderer in particular has been rewritten and runs a lot faster than in its previous incarnation.

A Starship Flying toward Earth in a Scene Rendered in Real Time

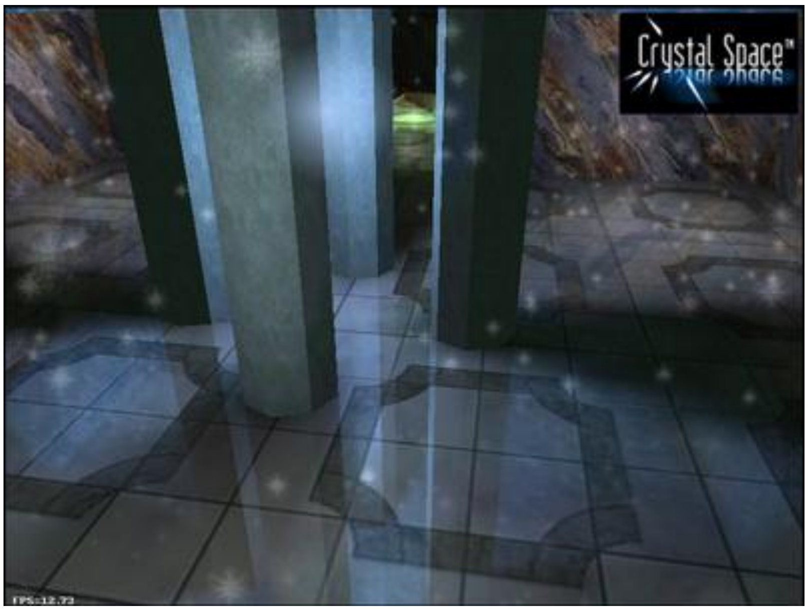
Also important is that Crystal Space can read many 3-D graphics file formats automatically. There are several importers supporting various 3-D formats (such as 3DS, OBJ, MDL, MD2, LWO and ASE). Inversely, the engine has a set of Python scripts so that environments and models can be exported to Crystal Space from within Blender.

Although the primary purpose of the engine is to produce 3-D graphics, it even has a 2-D API to go along with its 3-D API. “I wrote a full-blown windowing GUI based entirely on the Crystal Space low-level API”, says Zabolotny.

Pieces of separate code even can be used outside of Crystal Space in projects unrelated to game or multimedia development. These include a csIniFile class (for .ini file management), an SCF (shared class facility) subsystem, a csArchive class (which deals with .zip files) and a VFS (virtual filesystem) subsystem.

The most significant weakness of Crystal Space is its lack of good collision detection programming. Thomas Hieber, who has been developing Crystal Shooter, a first-person shooter, with Crystal Space, has been spending most of his time improving the engine's capabilities in this area. “There is some support in it, but it is not very useful for games”, says the 30-year-old software engineer from Germany. “There is only static testing of object-against-object that basically will return information about where the collision occurs, if any. But there is no good support for fast-moving objects.”

Another complicated issue is how Crystal Space handles lighting. It supports colored static and dynamic lighting with soft shadows, but getting these to work fast under all possible game processing circumstances is a challenge and one which Crystal Space currently does not totally meet.



Snow Falling over a Shiny Floor Demonstrating Crystal Space's

The Future of Crystal Space

The Crystal Space community definitely needs programmers to contribute. Tyberghein is looking for people who are skilled in programming graphic engine internals and adept in algorithmic thinking—essentially those who can help fine-tune the performance of the core engine itself. “I have lots of people helping on the other parts of Crystal Space (i.e., OpenGL and Direct3D programming, Windows and Linux porting) but very few people are capable of helping me with the engine”, he says.

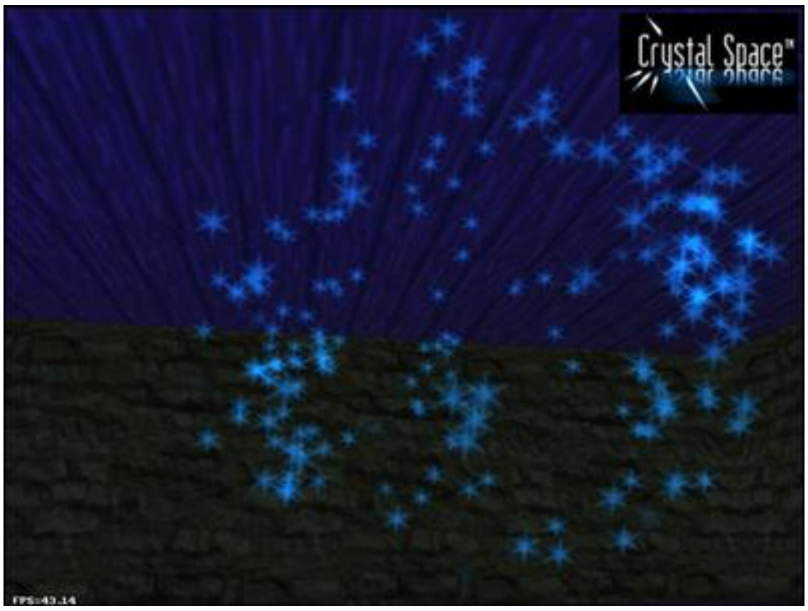
“If we had more good programmers, we could do much more”, Zabolotny says. “We primarily need people skilled in cross-platform C/C++ programming.”

As of this writing, the primary goal for the Crystal Space team is achieving API-stability. “Our development version is now rather stable, but there are still a few things to do”, says Tyberghein. The current release of Crystal Space, 0.90, serves as a predecessor to the long-awaited 1.0 release. The API between 0.90 and 1.0 should be nearly the same, but the release of 0.90 is meant to facilitate bug hunting and documentation writing.

One of the enhancements in 0.90 being tested is a revamped landscape-rendering engine that is more tightly and better integrated within Crystal Space's code than it was in previous versions. There are several new special effects that the graphics engine can draw, like hazes and lens flares, and there is the addition of a particle-rendering system. On a technical level, Crystal Space's tools have been made much more modular and simpler to access. More plugins and code, which were previously available in separate libraries, have been incorporated.

Ultimately, could Crystal Space ever evolve to the point where it has what it takes for commercial game development and become as widely used as proprietary 3-D graphics engines? Even Tyberghein expresses doubts:

> If you license the *Quake III* engine, then you're sure to get a quality product that will work. So if you want technical support, you should not use a free engine. However, if you feel like you can cope with the lack of support, or if funding is a problem, then an open-source engine is for you.



Stars Rendered with Crystal Space's Particle Animation System

Hieber concedes that “Crystal Space is miles away from *Quake III*”, but he does not believe this will hinder anyone from making great games with Crystal Space. It is, after all, well designed, though it doesn't necessarily have powerful technologies, which affect the quality of games. “Look at *Tomb Raider* or *Half-*

Life", he points out. "Neither has a really great 3-D engine, but they all have been successful because of the value of their game play."

Visit the Crystal Space site at crystal.sourceforge.net.

email: wen@airmail.net

Howard Wen has covered the video game industry for over ten years, writing for several publications and web sites including Wired, Salon.com, Playboy.com, GameSpot.com, O'Reilly Network and the *Dallas Observer*. He first started reporting on the video game industry as a staff writer for *VideoGames & Computer Entertainment*. He can be reached at his site, www.howardwen.com.

The Beowulf State of Mind

Glen Otero

Issue #97, May 2002

Where everyone meets OSCAR, becomes a Scyld administrator and builds a cluster that rocks.

So you want to build a Linux cluster? Yeah, you and everybody else it seems lately. However, the term Linux cluster is a popular one that doesn't mean the same thing to everybody. When asked, many people would associate Linux clusters with the high-availability, failover, redundancy and load-balancing infrastructure of some e-commerce web site or application server—your typical web farm. Others would associate Linux clusters with those parallel-processing, computational juggernauts, also known as Beowulf clusters. They would all be correct. If you think that's confusing, you should try to wrap your head around the term bioinformatics. Right now, Linux clusters and bioinformatics are two of the hottest technological trends, yet these terms couldn't be more vague. I should know, they're my job—well, more of an adventure really. Yep, that's me, Linux Prophet: clandestine cluster consultant, Beowulf Bad Boy, boon to bioinformatics and alliterative arse.

But enough about me—and more about Beowulf. The brainchild of Donald Becker and Thomas Sterling while working for a NASA contractor in 1994, Beowulf has grown into the poster child for open-source, clustered computing. The Beowulf concept is all about using standard vanilla boxes and open-source software to cluster a group of computers together into a virtual supercomputer. “What would you do with your own supercomputer?” you ask. Heck, what can't you do? For example, you can rip MP3s or assemble the human genome. These two applications take advantage of multiple processors in slightly different ways. When ripping MP3s on a Beowulf, you are essentially spreading all MP3 creation, typically an MP3 per CPU, over the cluster. In this way you are parallelizing several serial jobs by starting them all at once, each on a different CPU. Each MP3 job is an entity unto itself and doesn't need to exchange information with any other MP3 job while running. So, theoretically, one thousand MP3s (of equal size) can be completed by one thousand processors (of equal speed) in the same time that one MP3 can be completed by one processor. That's life in a perfect world. We won't really see linear speed increase when launching 1,000 MP3 jobs due to network latency and bandwidth overheads related to scaling up to 1,000 processors.

Most programs that run on Linux can be made to run on a Beowulf similar to the MP3 example with minimal effort to achieve vastly increased throughput. However, most experimental data sets, like those found in weather prediction or DNA sequence comparison, cannot be broken up similar to MP3s because the results from crunching one part of the data set affect the calculations on other parts. The computing that needs to be done in these cases is analogous to ripping one massive MP3 with multiple processors. You can imagine that if the work required to rip one MP3 were divided among multiple processors, the processors would need to communicate with each other to synchronize and coordinate its completion. Programs of this type utilize message-passing programming libraries so that results from one part of the program's computations can be communicated and synchronized with other parts running on different processors.

There are two common parallel-programming models: one utilizes a message-passing interface (MPI) library and the other a PVM, which stands for parallel virtual machine. Simple in theory but difficult to execute efficiently, parallel programming is a complex endeavor. The intricacies of high-performance computing are frustrating enough without having to work with the codes on expensive, temperamental, custom mainframes. The difficult-to-use, proprietary hardware typically associated with high-performance computing in the past is going the way of the dinosaur as more and more people use COTS (commodity-off-the-shelf) computer components and Linux to build Beowulf clusters that provide unbeatable price/performance ratios and fun on open, standardized platforms.

As cool as it sounds, practicing Beowulfery in the early days—like the rest of high-performance computing—was anything but straightforward and more akin to dabbling in witchcraft. Creating a Beowulf cluster required the downloading and installation of additional software tools, utilities and programming libraries to each Linux workstation in a typically heterogeneous network. Each Beowulf was a unique clustered hardware and software universe unto itself and was considered a work-in-progress. Cluster administration and maintenance required carnal knowledge of both resident hardware and software. The problems were many. But as with many open-source success stories, a community was formed that forged ahead.

Since 1994, a great deal of community and commercial development has resulted in significant advances in Beowulf computing and second-generation

Beowulf software distributions. That's right, distributions—on CDs. It's no longer necessary to cobble together disparate tools, software and drivers resident in far-flung corners of the Internet to build a Beowulf. Sound too good to be true? Well read on, because I'm going to discuss the different Beowulf software distributions, where to get them and how to get your own Beowulf cluster up and running sooner than you can say *schweeeeet*.

Physically, every Beowulf has a few things in common that make it a Beowulf and not just an ordinary network of workstations (NOW). Unlike a NOW, not every Beowulf node is created equal—it's a class struggle. Each Beowulf has a master, or head node, and multiple slave, or compute nodes.

Sporting a full-Linux installation, the master node is the command center of the Beowulf and runs the communicative dæmons necessary to transform LOBOS (lots of boxes on shelves) into a cohesive cluster. The master node is where the brains of the outfit (yes, you) wield supreme executive power over the compute nodes via installation and configuration of their software, the mounting of filesystems, job monitoring, resource allocation and access to the outside world.

The compute nodes, on the other hand, are there to do the master's computational bidding and are on a need-to-know basis as far as the master is concerned. Compute node intelligence can range from very dumb (containing very little code) to relatively smart (possessing a full-Linux installation). However, compute nodes having full-Linux installations still lack certain capabilities, provided by the master node, that keep them subservient to the master. For example, filesystems exported via NFS to the compute nodes (like users' home directories) typically reside on the master node. As a matter of fact, since all the Beowulf distributions we will cover embrace this approach by default, the cluster we build in this article will mimic this approach for simplicity's sake. But keep in mind that in reality, some I/O services can be distributed throughout the cluster, and files can be read and written, to and from, a variety of locations to accommodate data flow. But when starting out, it's often easiest to equip the master node with all of the cluster services and data needed by the compute nodes.

In a Beowulf, the master and slave nodes are networked together and communicate over a private LAN isolated from ordinary network traffic. The networking hardware is usually dictated by one's budget and ranges from 10Mbps Ethernet to very high-speed (greater than 1Gbps) proprietary hardware like Myrinet. The least expensive networks are achieved using Ethernet network cards, hubs and category 5 cable. And unless you want all users to be physically located at the master node when using the cluster, having your Beowulf command center connected to an outside network via a second network interface is highly recommended. With this setup, the master functions as a gateway straddling both the private Beowulf LAN and the public network of your organization. Users can log in remotely to the master over the public network in order to access the cluster's resources via the second network interface but cannot sidestep the master node and access compute nodes directly. Treating your Beowulf as a separate computational entity within your organization as I've outlined here provides many performance, administrative and security benefits. No wonder it's the design configuration supported by the current Beowulf software. Take a look at the physical layout of a typical Beowulf in Figure 1.

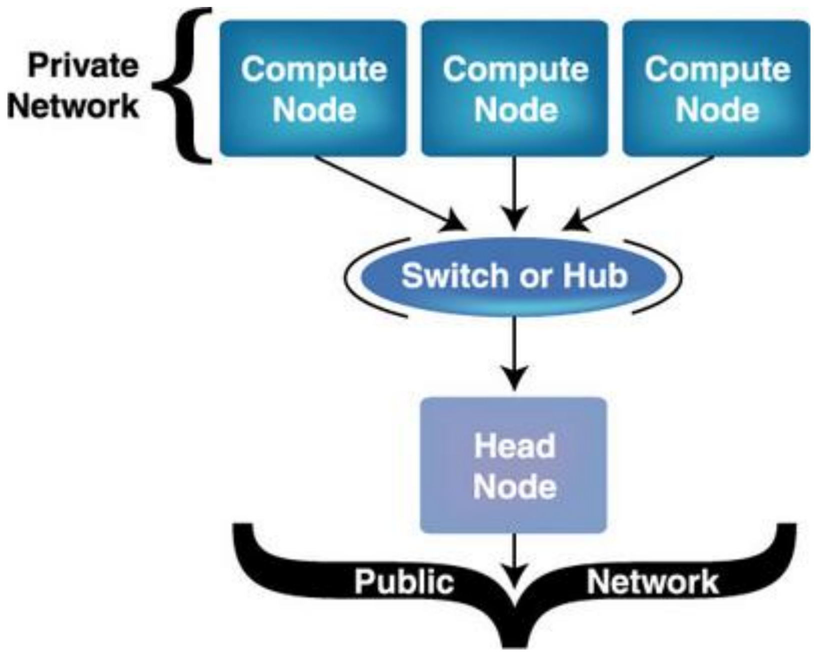


Figure 1. The Physical Layout of a Typical Beowulf

There are essentially two philosophies regarding the Beowulf operating system environment. Let me emphasize that both are good, just different. The two designs cater to different sorts of needs with regard to the cluster's purpose. That is, among other things, the cluster's functional role, the kinds of users, how many users, as well as the application(s) that will run on the cluster, strongly determine how cluster software and access should be configured and controlled. Not taking these things into consideration from the start will come back to haunt you later, so let's cover the two cluster design philosophies.

In the original Beowulf configuration, each node possessed a full-Linux installation, and user accounts were required on each node for an application to run. This configuration incurred a lot of overhead on each node to run any sort of application, and managing misbehaving processes was a rather draconian matter. Since then, rolling out this type of cluster has improved quite a bit. The use of DHCP, Red Hat's Kickstart, SSH, MPI, HTTP and MySQL have really improved cluster installation and administration. But once logged in to the master node, compute nodes still can be accessed by users and told to think for themselves. Compute node access and control is a feature that may be desirable for your particular cluster and its users, and thus represents an important administrative decision. Two cluster distributions built on this model are NPACI's Rocks and the Open Cluster Group's OSCAR (Open Source Cluster and Application Resource).

Driven by the creator of Beowulf, a second cluster paradigm has evolved that embraces a hive-mind approach to the master-compute nodes' relationship. The master node is the queen bee and possesses a full set of chromosomes, the ability to think for herself and control of the hive's actions. Having only a half a set of chromosomes, the compute nodes are the hive's drones and are as dumb as a bag of wet mice. The compute nodes cannot be logged in to remotely and therefore simply do the master's bidding. Despite my cool arthropod analogy, this configuration has been dubbed the single system image (SSI), and its flagship is the Scyld Beowulf distribution. The SSI represents the other extreme in compute node ideology and certainly has its advantages.

Which model is right for your purposes? Tough question, but it boils down to what you want users to be able to do to the system as a whole. Making an informed decision along these lines will require some tinkering around with the different available default cluster configurations. So, let's start to get a feel for the different clustering software by installing my homeboys' distribution, NPACI Rocks. This small group at the San Diego Supercomputer Center has built a rock-solid, easy-to-use Linux cluster distribution. All you need to build a cluster that Rocks is a network of x86 boxes (IA32 or IA64) similar to that in Figure 1, an internet connection, a CD burner, a CD and a floppy disk.

To begin, cruise over to the Rocks web site by pointing your browser at rocks.npaci.edu. Here you will find a brief introduction to cluster building in general and methods specific to the Rocks' Project madness. One thing that will become apparent when perusing the web site is that the Rocks cluster distribution was built with one goal in mind: to make building and administering Beowulf clusters easy. To achieve this lofty goal, the Rocks group 1) based their distribution on Red Hat Linux, 2) added to Red Hat Linux all the open-source software needed to use and administer a Beowulf cluster out of the box, 3) packaged all the cluster software in RPM format, 4) used Red Hat's Kickstart software to automate the master node and all compute node installs, 5) created a MySQL database on the master node to organize cluster information and 6) provided some software to tie it all together. All great ideas. As far as clustering software goes, the Rocks distribution includes PBS, Maui, SSH, MPICH, PVM, certificate authority software, Myricom's general messaging for Myrinet cards and much more. As if that weren't enough, they also designed some features that allow you to customize and build your own Beowulf distribution by adding your own software to Rocks. How cool is that?

Once you're at the Rocks web site, clicking on the Getting Started link on the left-hand side will link you to the stepwise instructions for building a Rocks cluster. Step 0 briefly describes the basics of physical cluster assembly. Hardware may vary, but the configuration should resemble that depicted in Figure 1.

Step 1 consists of downloading the bootable ISO images from the Rocks FTP site and burning your own CDs. NPACI Rocks is currently based on Red Hat Linux 7.1, so there are two installation CDs, but you only need the first one to build a Rocks cluster.

Step 2 consists of building the kickstart configuration file and installing the master (front end in Rocks parlance) node. Thankfully, the Rocks group has provided a CGI form on their web site that you can fill out to generate this file. Clicking on the link "Build a configuration file" will take you to the form. The form prompts you for the information it needs to generate a kickstart file that will configure the internal and external interfaces of the front end, as well as the administrative, timekeeping and naming services for your cluster. Plenty of advice and default values are provided on the web page to help you fill out the form.

Once you've filled out the form, click the Submit button. After clicking the button, your browser should ask you to save the file. Save the file as ks.cfg, and then copy the file to a DOS-formatted floppy. You now have the software necessary to install a Beowulf cluster—the Rocks CDs and your front-end-specific kickstart floppy. You're ready to Rock! Power on your front-end node-to-be, make sure it is set to boot from the CD-ROM drive, insert CD1 and reset the machine. At the "boot:" prompt, type **frontend**, insert the floppy and watch the front end install itself. When installation is complete, the machine will eject the CD and reboot itself. Remove the CD and floppy before the machine reboots to avoid re-installation.

Log in to the front end as root. You will be prompted to create your SSH keys. Once your keys have been generated, run **insert-ethers** from the command line. This program eases Beowulf installation and administration by entering compute node information it parses from compute node DHCP requests into the Rocks MySQL database. Choose Compute from the program's menu, insert CD1 into the first compute node, cycle the power on the node and the compute node will install itself. When installation is complete, the machine will eject the CD. Take the CD, place it into the next compute node and cycle the power. Repeat this process for all the compute nodes.

That's all there is to it. Granted, there's a lot going on behind the scenes that the Rocks guys have made transparent to you, but that was their intention. For a better understanding of the Rocks nuts and bolts, check out the documentation on their site. It's pretty good, and they're adding stuff all the time.

You're now teed up to begin running applications on your Beowulf. For a short tutorial on launching applications that use MPI and/or PBS, check out the information on the Rocks web site under Step 4: Start Computing. One bit of advice, the mpi-launch command is for starting MPI jobs that run over Myrinet, while mpi-run is for running MPI jobs over Ethernet.

That's all for now. Keep your eyes peeled—before you know it, I'll be back with another easy-to-install Beowulf distribution. And then another. And another. And then we'll build a computational grid with them. And then—world domination.

Resources

email: gotero@linuxprophet.com

Glen Otero has a PhD in Immunology and Microbiology and runs a consulting company called Linux Prophet in San Diego, California.

Interview with Ted Ts'o

Don Marti, Vernon

Issue #97, May 2002

Ted discusses his work on the Linux kernel, Linux International, Linux Standard Base and other areas of the Open Source community.

Don Marti and Richard Vernon recently had the rare opportunity of taking some time from Ted Ts'o's tight schedule to talk about his role with the Linux kernel, IBM and the Linux community. Ted seems to be everywhere in the Linux community—inside the kernel and out. He is currently a senior technical staff member of the Linux Base Technology Team for IBM's Linux Technology Center. He also chairs the Technical Board of Linux International, serves on the Board of Directors for the Free Standards Group, is a member of the Internet Engineering Task Force and serves on the Security Area Directorate of the IETF. Previously, he worked at MIT in Information Systems, where he was the development team leader for Kerberos. Through it all he's played a principal role in the development of the Linux kernel.

LJ Many Linux enthusiasts know you for your work on the Linux kernel, but are perhaps less familiar with your service to Linux International and the Free Standards Group. Could you talk a little about your respective capacities with those organizations?

Ted Well, I chair the Technical Board of Linux International. Linux International is a vendor group that got started back in the good old days of Linux startups—when Bob Young would personally show up at tradeshows and help hand out CDs containing the Slackware distribution. So from the beginning, Linux International had as a strong emphasis the concept that its members should band together to help "grow the pie".

The technical board was there to help make sure the organization stayed connected to its technical roots and later picked up the responsibility to examine applications to the Linux International Development Grant Fund, which is still operating today.

Very recently, Linux International has begun considering a new program that will focus on strengthening the various local Linux Users' Groups and working with them to support people who are interested in doing various types of "Linux Advocacy" (i.e., pushing Linux to be used in local public schools or in the corporate infrastructure). This is an idea that I've been discussing with Jon "maddog" Hall, and I think it's a great initiative. I hope it works out well.

As for the Free Standards Group, I currently serve on the Board of Directors for the FSG. The FSG provides a legal and financial home for the Linux Standards Base (LSB) and the Linux Internationalization (Li18nux) efforts. I was involved with the LSB from almost the very beginning because I believe in providing a stable environment so that members of the community can release binary distributions of programs that will run on any Linux system of the same architecture, regardless of the distribution that the user chose to use.

I was student systems programmer in MIT Project Athena during the height of the UNIX wars and saw firsthand how incompatibilities between the various UNIXes allowed Microsoft to dominate the desktop. So as a result, I've always thought that the LSB is incredibly important for the Linux community.

LJ How do you feel about the recent progress of LSB (the release of LSB 1.1), and what do you feel is the future for LSB—will it be something that evolves into an ubiqitous standard? What might be some of the advantages of LSB for developers who distribute software in source-code form?

Ted Progress on the LSB front has been slow but steady. LSB 1.1 isn't perfect, but it's at the stage where it should be possible for both distributions and independent software vendors to start implementing against it. We expect to start seeing LSB-compliant distributions and application programs within a year.

The LSB standard is working to make it possible for third-party application programs to be installed and run across multiple distributions. Initially, the majority of packages on a Linux system will still be provided by the distribution and will not be LSB-compliant packages.

Hopefully, as the distributions start seeing the advantages of the LSB, and as demand increases for more commonality between the various distributions, the LSB will help encourage distributions to start converging gradually, as new features are added. This will act to benefit all developers, even those who distribute code in source form already.

ABI-compatibility, while most important to people or companies that distribute sources in binary form, is also important to people who are using exclusively open-source software. For example, some library maintainers don't bother to change symbol names or even in some cases, library version numbers, when they make incompatible library changes. This can cause all sorts of headaches if two application programs installed on the same system need to reference different libraries. An extreme example of ABI-instability can be found in libgal (the GNOME Applications Library), which has had 19 different, incompatible ABI changes in about as many months. Even if source is available, this kind of ABI-instability is extremely inconvenient.

LJ What areas will FSG look at standardizing next?

Ted Well, there are two groups that have approached the FSG. One is interested in standardizing some kind of high-level printing libraries interface. Another group is interested in standardizing library interfaces for clusters. In general, the FSG doesn't try to find new technologies to standardize; instead it allows people who are interested in forming a workgroup to work on some standard to do so. The FSG Board simply insists that the process is open and, to the extent possible, that all interested parties are at the table while the standard is being developed.

LJ Could you briefly describe your work at IBM?

Ted Well, I'm continuing to work on the kernel, especially the ext2/ext3 filesystem work. I've also been consulting with some of the other teams at the Linux Technology Center, helping them with design issues and helping them make their contributions be more easily accepted into the mainline Linux kernel.

LJ What would you consider to be some of the most significant developments of the 2.5 series kernel?

Ted It's early in the 2.5 development series, so it's really hard to say right now. I'd say that scalability to larger machines and associated sub-goals, such as reducing or removing the need for the global "big kernel lock" is certainly going to be one of the more significant efforts in the 2.5 series. The introduction of the O(1) scheduler is also quite significant. Work to continue improving the virtual memory subsystem and the I/O subsystem also is ongoing and ultimately very important. With the exception of a few new features, such as better ACPI support and asynchronous I/O support, I suspect most of the improvements in the 2.5 kernel will be performance-related.

That being said, as Linus has said—and I very much agree—a lot of the exciting new work that is happening in the Linux community isn't necessarily happening in the kernel, but in user land. For example, who would have thought that five years ago, Linux would have not one, but *two* graphical desktop environment systems under development?

LJ You are the author of /dev/random. How will the Linux kernel hackers approach crypto-enabling technology in the kernel? Cautiously, or are the developers jumping in with both feet now that US export restrictions are looser?

Ted Well, Peter Anvin did some wonderful work laying the legal groundwork (thanks must also go to Transmeta for paying the legal bills) so that cryptographic software could be distributed from the kernel.org FTP distribution network. There are certainly some people who are still a bit cautious. That's understandable since many developers have lived behind the Crypto Iron Curtain for so long that they're still afraid the US government might change its mind and suddenly try to regulate cryptography again. At this point though, my belief is that the crypto genie is so far out of the bottle that this sort of nightmare scenario is very unlikely.

I think that it's only a matter of time before developers start adding more cryptography into the kernel. On the other hand, there's a lot of cryptographic solutions where the right place to put things really is outside of the kernel.

LJ With all the many demanding activities with which you are involved, how do you stay organized and find sufficient time to devote to each activity?

Ted It's hard. One of the disappointing things about being involved with doing more organizational tasks, such as serving on the board of the FSG and working on the LSB, is that it means I have less time to do real kernel-level programming. But, someone has to do it, and I happen to be somewhat good at it, so....

That being said, I am hoping that I'll be adjusting my workload so that I will have more of a chance to do some real programming than I have in the past year or two.

One of the other ways that I try to find time is to pass off projects to other people. For example, I was one of the original instigators of bringing Pluggable Authentication Modules architecture to Linux. At the time, I was working at MIT, and I visited Sun Microsystems to discuss some issues relating to Kerberos. Near the close of the meeting, the Sun engineers introduced me to this thing called PAM, and I immediately thought that this was a really great idea, and gee, wouldn't it be great if Linux could have it too. So I started suggesting that this would be a good thing to do, and next thing I knew, Andrew Morgan had stepped forward and ran with it. The funny thing about this whole story is that even though the engineers had been working on PAM for at least a year or two before they introduced it to me, the Linux-PAM Project had an initial implementation working, which was shipping in commercial distributions before Sun was able to ship a version of Solaris that had PAM support. That's what's so great about the open-source model.

LJ How is IBM contributing to the development of open-source software?

Ted Well, IBM started the Linux Technology Center with some 250 or so engineers, spread out across some 16 cities and six countries, all working on open-source software. And we have a mandate to try to get our changes accepted into the mainline versions of the kernel or whatever open-source project we might be working on. So we're trying very hard to work as members of the Linux and OSS community. Of course, the sort of OSS enhancements we choose to work on are also those that are important to IBM's customers, but that's true at all Linux companies. The wonderful thing is that in most cases, the interests of the global Linux community and the interests of IBM's and other Linux companies' customers are the same.

LJ What pieces of the kernel are you working on right now?

Ted Right now, I've been mainly focused on the ext2/ext3 filesystem. I'd like to work on reworking the tty layer, but there are only so many hours in a week. Maybe in a month or two, I'll have some time to actually try tackling that.

LJ How long have you been interested in amateur radio and what got you interested?

Ted I've had an amateur radio license since 1997. I got involved because I knew a lot of people at MIT who were using the MIT UHF Repeater to communicate and that sucked me in.

LJ What has been your role in developing Linux POSIX capabilities, and what is your position on the current number of 28? Do you think this should be maintained, or expanded?

Ted Like PAM, Linux POSIX capabilities is one of those things that I tried pushing, but with less success. I still think that something like POSIX capabilities is important, but I'm not so sure anymore that Posix capabilities is the right way to go about solving the problem. Most system administrators have trouble dealing with 12 bits of UNIX permission bits per file. Adding another 3 × 28 = 84 capability bits that must be configured correctly or the executable will either stop working or be insecure, is a nightmare.

A simpler system where programs are still setuid root, but then permanently drop all of the capabilities they won't need, is certainly a lot less flexible than the full POSIX-capabilities model, but I think it is so much easier to administer that this makes it far more important than other considerations.

LJ Have you tried SELinux? If so what do you think?

Ted No, I haven't had time to actively install and play with SELinux. I think it's great that the NSA has been working on it, though.

LJ Do you see a conflict yet between optimizing Linux for throughput on mid-range or large servers and going for small size and latency on embedded-class systems?

Ted Well, I think it's a challenge to come up with algorithms that work well on both mid-range and large servers, yet are also well adapted to typical desktop machines. But, I think it's doable. In some cases, perhaps the end result won't look like what has traditionally been done to support large-scale servers or small embedded-class sytems. But that's what makes working on the Linux kernel so neat! We're not always implementing things the traditional way, but trying to find new ways of skinning the proverbial cat.

So no, I don't think there will necessarily be a conflict between optimizing Linux both for large servers and small servers. I do believe that the primary tuning target will continue to be the typical desktop machine, since that's what most developers have and can afford. However, the typical desktop (as well as the typical embedded system) has been gradually becoming more and more powerful, and over time, the range of systems where Linux will have excellent performance will continue to grow with each new major version of Linux.

LJ Besides the Rubini and Corbet book, how would you recommend that people who want to contribute learn about the kernel, both for writing drivers for 2.4 and wild and woolly features for 2.5?

Ted The www.kernelnewbies.org site is definitely one of the best places to start. Beyond that, the best way to learn about the kernel is to jump in and start playing with it! Come on in! The water's fine!

LJ Anything you'd like to add?

Ted Only that I consider myself incredibly lucky. Ten years ago, Linux was just a hobby; something that I did for fun. Now it's become a major force in the computer industry, so I can work full-time on something that I once did just because I loved doing it. That's neat. That's really neat.

email: dmarti@zgp.org

Don Marti is technical editor of *Linux Journal*, and **Richard Vernon** is editor in chief of *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

Databases and Zope

Reuven M. Lerner

Issue #97, May 2002

Reuven shows you how easy it is to turn a simple Zope site into one that reads and writes data in a relational database.

Just about anyone who creates a serious web site will eventually want to connect it to a relational database. Relational database systems might be 30-year-old technology, but they're flexible, safe and fast. Using a database ensures that we can store and retrieve data needed by our web application without having to create our own persistent storage layer. This results in fewer bugs, greater speed and far greater safety.

Zope, the object-oriented web application server that we have discussed over the last few months, includes a built-in object database known as ZODB. ZODB is both powerful and easy to use; everything in Zope, including DTML documents and folders, is stored as an object in ZODB. The fact that ZODB supports such database concepts as transactions means that you can use it to store serious data, confident that no one else will be modifying information during the execution of a long, complex query.

But in many cases, ZODB isn't a good match for the data we want to store and retrieve. In many cases, this is because the data already exists, and we simply want to use Zope to access it. Perhaps we're creating a new persistent storage layer but want people to be able to access it from outside of Zope. Perhaps our data is more suited for the relational database model than an object database. And finally, perhaps our organization's IT department requires that all information be stored in a relational database.

For all of these reasons and situations, the standard Zope installation defines a ZSQL method object. This month, we'll take a look at ZSQL methods and at the general integration of Zope with relational databases. As you'll see, it's very easy to turn a simple Zope site into one that reads and writes data in a relational database.

Database Connections

Before we can work with a database, we must first connect to it. In Zope, we accomplish this by creating a database connection object. A Zope site can contain any number of such objects, each of which is then available for sending SQL queries to a database.

Zope comes with a single kind of database connection, which allows you to work with the simple Gadfly relational database. But while Gadfly is good for demonstrating Zope's database connectivity, it cannot match any other relational database in terms of speed or functionality. I suggest skipping Gadfly completely, installing a database adapter for the server to which you intend to connect.

I have a running PostgreSQL server on my office database server, so I decided to install the psycopg database adapter, one of several PostgreSQL adapters currently available on the Internet. (See Resources for more information on psycopg.) When installing these (and other) packages, remember that Zope typically comes with its own copy of Python, which is independent of any other copies that might be installed on your system. This means that you must install psycopg into the Python library defined by Zope (using $ZOPE/bin/python) rather than /usr/local/bin/python or /usr/bin/python.

Before we can install psycopg, we must first install the mxDateTime class written and distributed by eGenix. This package makes it possible to work with dates and times beyond the current UNIX limits (starting in 1970 and lasting until 2038) and provides a number of convenience routines to work with dates and times in various formats. Even if you don't use this module, you still will need to install it in order to get psycopg to install correctly. You can download mxDateTime from www.egenix.com/files/python/eGenix-mx-Extensions.html.

Note that you will want to download the "base" extensions package (which is free), rather than the commercial extensions package. Even if you are using an RPM-compatible distribution of Linux, you should not download the RPMs for mxDateTime. This is because we need to compile and install the libraries into our Zope Python tree, rather than the system Python tree.

After downloading and unpacking the mxBase package, you should be able to install it by switching into the mxBase directory and typing

```
$ZOPE/bin/python setup.py install
```

This will compile and install the mx module into your Python installation.

Installing psycopg

We're almost ready to install psycopg, a combination of Python and C that requires you to have the PostgreSQL development libraries installed. If you install PostgreSQL using RPMs, then you will need the postgresql-devel RPM for the appropriate version of PostgreSQL that you are running. This should install files in /usr/local/pgsql and /usr/include/pgsql, although some installations use postgresql instead of pgsql in both of these paths.

Now download the psycopg source code from initd.org/pub/software/psycopg. I retrieved version 1.0.4, but new versions seem to arrive every few weeks, so be sure to retrieve a recent version. In order to unpack and install psycopg, you will need to make the makesetup shell script (installed into $ZOPE/lib/python2.1/config in Zope 2.5b1, the latest version as of this writing) executable:

```
chmod 775 $ZOPE/lib/python2.1/config
```

To configure psycopg, change into its source directory and enter the following:

```
./configure
    --with-python=$ZOPE/bin/python
    --with-zope=$ZOPE
    --with-mxdatetime-includes=$ZOPE/lib/python2.1/
      site-packages/mx/DateTime/mxDateTime
    --with-postgres-includes=/usr/include/pgsql
```

You should obviously change the paths to reflect your installation, paying particular attention to the Python version number (2.1, in my case) and the PostgreSQL include directory.

While I remain convinced that there is a way to avoid doing so by passing configure another option, it seems that you must now edit the Makefile by hand to add a new header directory to the CFLAGS variable. Open the Makefile in your favorite editor and modify the CFLAGS definition (line 90 in my version) to include headers from $ZOPE/include/python2.1. Thus, if $ZOPE is /usr/local/zope, you would add the following to CFLAGS:

```
-I/usr/local/zope/include/python2.1
```

Save the Makefile, and then install psycopg with

```
make && make install && make install-zope
```

This will compile and install psycopg for Python and Zope within your $ZOPE directory.

Finally, move the psycopg shared library (psycopgmodule.so) from $ZOPE/lib/python2.1/site-packages to $ZOPE/lib/python2.1/lib-dynload/.

Configuring psycopg

You can test psycopg by restarting Zope and adding a new product in the root directory. (Unfortunately, restarting Zope is the only way to tell the system that a new product has been installed.) The product you want to install is called Z Psycopg Database Connection in the "add product" menu in the upper right-hand corner of the Zope management screen.

Each database connection object allows you to connect to a single database on a single host, with a single user name and password. This means that if you have divided your information across two different databases (or two different database servers), you will need two connection objects.

When you choose Z Psycopg Database Connection from the "add product" menu, you are then asked to provide some basic details about this database connection. You must enter an ID (which must be unique within a folder) and a title (which will appear in the management screen), as well as a database connection string. This connection string tells Zope how to find and connect to a PostgreSQL server. In my office, the atf database sits on the PostgreSQL server on "databases", and I can connect to it with the user "reuven" and no password. Thus, I enter the following connection string:

```
host=databases dbname=atf user=reuven
```

You can leave the rest of the items in their default state if you wish. Click on the Add button, and you will be returned to the folder in which you added the new connection object.

Clicking on the connection object displays several Zope tabs that you can use to administer the database connection. The four most interesting ones are:

- Status: this tab tells you whether the database connection is open (i.e., whether it is connected to your PostgreSQL server). It also allows you to close the connection.
- Properties: you can modify the properties that you set when you initially created the database connection object. This is particularly useful if the database is moved to a different server or if you change the password necessary to access it.
- Test: you can test the database connection by sending any SQL query to it. Of course, the query must be valid; if you send illegal SQL or try to address a table that does not exist, then you will get an appropriate error from the PostgreSQL server. For example, you can enter **SELECT * FROM pg_database;**. You can enter any SQL via this box, which can be convenient for testing your database when you have no direct Telnet or SSH connection. If you enter an INSERT or UPDATE query, Zope will

indicate that the query didn't return any results. As always, it's a good idea to avoid using **SELECT *** except in trivial examples to avoid being surprised by the order or name of columns in the result set.

- Browse: you can look through the tables in a PostgreSQL database with the browse tab, which displays a Zope tree-style list of tables and fields.

ZSQL Methods

Now that we have a database connection, we can create one or more ZSQL methods. Each ZSQL method is a single SQL query (with variable arguments, if you want) that works with a connection.

Let's create a ZSQL method that lets us add a new name into an address book. Of course, this means that we must first have an appropriate table defined in our database. We can create the table by sending the content of Listing 1 to PostgreSQL, either in the test tab for our database connection or by using the traditional psql command-line interface.

Listing 1. Creating a Table

If you try to add a ZSQL method when no database connection is available, Zope will display an error message, complaining that it could not find any suitable database connection.

Zope's built-in system of “acquisition” means that a ZSQL method can use any database connection above it in the Zope hierarchy. A user thus can choose a different database connection for each method, making it possible to integrate information from different databases in a single application—or to migrate your web site from one brand of database to another.

To create a ZSQL method, go to the folder in which you created your database connection and choose “ZSQL method” from the “add product” menu. You will be asked to enter several items: an ID (which must uniquely identify your object in this folder), a title (which will be visible in the management interface), arguments (which we will discuss in the next section) and finally the SQL itself. Your SQL query can be as simple or as complex as you like and can perform an INSERT, UPDATE or DELETE.

Once you have added your ZSQL method to the system, clicking on it brings up a number of Zope tabs. One of these tabs is labeled test, and (as you might expect) it allows you to test the query. If your query has arguments, then you are asked to enter them in an HTML form. If not, you are simply asked to click on the Submit Query button. This returns, as with the test tab from our database connection object, an HTML table describing the results of our query—or a message indicating that our query returned no results.

We can create a ZSQL method for each query we wish to perform. While this might seem a bit odd, it's actually a very flexible and elegant solution that I've grown to appreciate more and more. If I expect to perform 20 queries in a web application, I can put each of them in a separate ZSQL method and then invoke those methods from within DTML pages.

Within a DTML page, we can retrieve results from a ZSQL method by naming it in a <dtml-in> tag. For example, I create a ZSQL method that implements the following query:

```
       SELECT first_name, last_name, phone_number,
              fax_number, cell_number
         FROM AddressBook
     ORDER BY last_name, first_name
```

If I give this ZSQL the name “names-and-phone-numbers”, then I can invoke it from within a DTML document with the code in Listing 2. In just a few lines of DTML, we have successfully managed to produce a simple (but useful and flexible) ZSQL method. But how does it work?

Listing 2. Invoking ZSQL from within a DTML Document

When Zope receives a request for this DTML document, it parses the DTML and executes each of the tags contained within. The <dtml-in> loop construct expects a sequence as an argument; in this particular case, the sequence is the result from invoking the names-and-phone-numbers method. The <dtml-in> tag also assigns one variable for each column in the returned result set. This is how we can use the <dtml-var first_name> tag to print the user's first name; Zope automatically assigns the value of the first_name column to a variable called first_name.

In order to avoid printing unnecessary and blank lines, we use <dtml-if> to check that we did not receive a NULL or empty value back from PostgreSQL.

ZSQL Arguments

It's obvious how we can use ZSQL methods and DTML to perform the same query each time. But if we want to modify our basic query each time it is run, we will need to define one or more arguments.

For example, if we want to retrieve information about someone based on their last name (or a portion thereof, using SQL regular expressions), we will want to define the following sort of ZSQL method:

```
       SELECT first_name, last_name, phone_number,
              fax_number, cell_number
         FROM AddressBook
        WHERE last_name LIKE
     ORDER BY last_name, first_name
```

In DTML, we can replace the *XXXXXX* with the <dtml-sqlvar> tag, which automatically handles quoting for us. We must name the SQL variable that we are using, as well as indicate its type:

```
      SELECT first_name, last_name, phone_number,
             fax_number, cell_number
        FROM AddressBook
       WHERE last_name LIKE <dtml-sqlvar name_sqlregexp
                                type="string">
    ORDER BY last_name, first_name
```

In order for the above ZSQL method to work, we must name an argument (name_sqlregexp) in the appropriate text box when creating our method. Zope will take the value of that variable, place it inside of our query and retrieve the results.

We can put even more of the burden on Zope if we use a <dtml-sqltest> tag, which operates similarly to <dtml-sqlvar>:

```
      SELECT first_name, last_name, phone_number,
             fax_number, cell_number
        FROM AddressBook
       WHERE <dtml-sqltest name_sqlregexp op="like"
              type="string">
    ORDER BY last_name, first_name
```

If we have stored the above query in a ZSQL method named select_by_last_name, then Zope can automatically produce skeleton DTML documents that allow users to enter search terms and see results. To do this, simply choose the “Z Search Interface” product from the “add product” list. You will be able to choose from all of the searchable objects on the system, including the ZSQL method that we just created (select_by_last_name). Choose this, and give the report an ID (I used search_by_last_name). Also give a name to the “input ID”, which is a fancy term for the HTML form that will be used to send input to search_by_last_name. (I named it search_by_last_name_form.) In modern versions of Zope, you also must indicate whether you want the system to create DTML methods or page templates; we want the former.

Clicking on Add creates two new DTML methods in the current folder, corresponding to the names that you entered in the form. Clicking on the “input ID” URL will present a simple HTML form into which you can enter an SQL regular expression. Clicking on the submit button will send your query to the search_by_last_name DTML method, which will in turn invoke our ZSQL method (select_by_last_name), which will then pass along our query to PostgreSQL. PostgreSQL returns results to select_by_last_name, which returns a result set to search_by_last_name, which then displays them in our web browser.

You can, of course, modify the DTML methods that are created to match the style of your own site. You also can copy the DTML that Zope created automatically into your own DTML pages, using them as examples of how to create your own database queries.

Inserting

The only major task left is the implementation of an INPUT query, which adds items into the database. Luckily, this is rather easy: we create a ZSQL method that inserts a row into the database. Then we create a DTML document that submits its HTML form elements to another DTML document. This second document invokes <dtml-call> to our ZSQL method. *Voilà*--our record is inserted into the database.

Listing 3 shows the ZSQL method that we need, which I named insert_address_data. Now we'll create a simple DTML document, which will contain an HTML form (see Listing 4).

Listing 3. The ZSQL Method insert_address_data

Listing 4. DTML Document Containing HTML Form

Finally, we create the DTML document insert_address that receives input from insert_address_form and passes its arguments along to the ZSQL method insert_address_data:

```
<dtml-var standard_html_header>
<h2><dtml-var title_or_id></h2>
<dtml-try>
    <dtml-call insert_address_data>
<dtml-except>
    <p>Sorry, but the INSERT didn't work.</p>
<dtml-else>
    <p>Successfully inserted!
</dtml-try>
<dtml-var standard_html_footer>
```

Users can now insert information into our PostgreSQL table using an HTML form pointed at insert_address_form, and they can retrieve it using search_by_last_name_form. It's rather impressive that we can do so much in so few files—and even more so that we didn't have to touch a text editor once in order to get this to work, but that it could all be done using nothing more than our web browser.

Conclusion

While they are not perfect, I find ZSQL methods to be an elegant way to connect a page of HTML with an underlying database. ZSQL is yet another way in which Zope demonstrates its very flexible, elegant approach to web development—

albeit one that makes you scratch your head several times before it all becomes clear and obvious. Someone who already knows DTML and SQL can easily integrate a database into their Zope application—and with ZSQL methods, it's possible to divide work on a site between those who know SQL (and work on the ZSQL methods) and those who want to work on the DTML methods that invoke them.

Resources

email: reuven@lerner.co.il

Reuven M. Lerner is a consultant specializing in web/database applications and open-source software. His book, Core Perl, was published in January 2002 by Prentice Hall. Reuven lives in Modi'in, Israel with his wife and daughter.

Archive Index Issue Table of Contents

Advanced search

Getting to Know You…My Kernel

Marcel Gagné

Issue #97, May 2002

Marcel shows you different ways of looking at the Linux kernel from the outside.

François, *mon ami*, are you still listening to that Vorbis broadcast of the Linux kernel on Free Radio Linux? While I admire your determination to get to know the Linux kernel intimately for this month's issue, the entire transmission will take months. There are more interesting ways to get close to your kernel, François, and you will not look so lost, *mon ami*. Besides, our guests will be here soon and we must be ready.

Ah, *bonjour*, *mes amis*! It is so good to see you. Welcome to *Chez Marcel*, home of fine Linux fare, great wine, good food and excellent company—your company, *mes amis*. Please, sit and make yourselves comfortable. My faithful waiter, François, will run to the cellar and bring back the wine. We have some 1999 Cornas Champelrose that will be an excellent wine for today's menu.

And speaking of the menu, I would like you to think of tonight's menu as a kind of buffet or smorgasbord of little finger foods, small but very useful applications with which you can pepper your desktop. Ah, François, you are back. Wonderful. Please, pour for our guests.

Before you arrived, *mes amis*, I was telling François that getting to the heart of the Linux kernel is sometimes best done by looking at it from the outside. After all, your kernel is the heart and soul of your Linux system, not to mention the engine that drives all other applications. One way to appreciate what your system is doing is through the use of system monitors. Now, as you already know, there are literally hundreds of such programs, so where do we start? In my Linux kitchen, I run different desktops from time to time, most notably Window Maker and KDE. I am particularly fond of KDE's power and beauty, but Window Maker has a special place in my heart as well, specifically its elegant dock applications. These small programs in 64 × 64 pixel squares come in every variety, including a host of system and resource monitors.

For instance, check out Vito Caputo's WMSysMon (originally created by Dave Clark) at www.gnugeneration.com/software/wmsysmon. This little monitor provides visual feedback on memory use, swap use and percentage of I/O, as well as interrupts and pages being swapped in and out. It's a lot to pack into this small monitor, but seeing it in action (Figure 1) will give you an idea of what to expect.

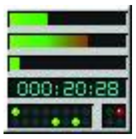


Figure 1. WMSysMon tells all!

Building this little application takes nothing more than extracting the source, moving into the application directory (wmsysmon-0.7.6/src) and executing a **make** followed by a **make install**.

To fire up this little monitor, simply run the command **wmsysmon**. You have the option of running the interrupts display as a meter or with nice LED-style blinking lights. *Mes amis*, I must tell you that what originally excited me about computers all those years ago was the important-looking display of blinking lights. Consequently, I run WMSysMon with the -l option.

You must admit, *mes amis*, that although WMSysMon looks very professional, when we cook with Linux, we occasionally lean toward lighter fare, *non*? If WMSysMon is too serious a monitor for you, consider BubbleMon-dockapp available at timecop's web site, www.ne.jp/asahi/linux/timecop.

Mes amis, I am going to pretend that you will find this program useful and immediately tell you how to build it:

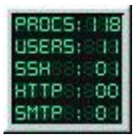
```
tar -xzvf bubblemon-dockapp-1.4.tar.gz
cd bubblemon-dockapp-1.4
make
make install
```

To run the program, type **bubblemon**. Several people have worked on various incarnations of a bubbling resource monitor applet, but the real genius here is the inclusion of a cute little yellow duck swimming across the top of a bubbling pond. The more your system is doing, the more bubbles rise to the surface. As you start to run out of memory, the water level will rise until your duck disappears from sight.

Figure 2. No, the duck tells all.

Move your mouse over the BubbleMon-dockapp, and your duck and pond will slowly fade to be replaced by a CPU usage graph complete with run average stats for the last 5, 10 and 15 minutes. Here's another tip. If you right click on the monitor while your duck is fading, you can capture an image in transition. In other words, you can see a ghostly image of your duck swimming across your CPU graph. It is possible to disable the swimming duck with the -d option, but why would you want to do such a thing? Ah, levity.

Of course, your kernel is busy doing other things as well, dealing with users, running processes, dealing with e-mail, web server requests and other internet connections. WMiNet is a Window Maker application created by Dave Clark, Antoine Nulle and Martijn Pieterse. Its purpose is to monitor (among other things) various network connections, processes and users. WMiNet is available at www.neotokyo.org/illusion.



```
PROCS: 118
USERS:  11
SSH  :  01
HTTP :  00
SMTP :  01
```

Figure 3. WMiNet, keeping an eye on things.

After you have extracted the source using **tar -xzvf wminet-2.0.3.tar.gz**, change to the wminet.app/wminet directory and do a **make** followed by a **make install**. The install will create a configuration file in /etc/wminetrc, where you can define what specific system processes the monitor watches. What I find particularly interesting with this one is that each of the five display lines can be configured for a single-click command. For instance, in position 1, I have a listing of active processes. If I click that line, it automatically launches **top**. Here's a sample from my own /etc/wminetrc file. I think you will find it is basically self-explanatory:

```
action1=rxvt -bg black -fg white -e top
action2=rxvt -bg black -fg white -e sh -c "w; read"
action3=rxvt -bg black -fg white -e sh
-c "netstat -etpn; read"
action4=rxvt -bg black -fg white -e tail
-f /usr/local/apache/var/logs/access_log
action5=rxvt -bg black -fg white -e sh -c "df -k;read"
```

When I have looked at these little Window Maker dock applications in the past, I have mentioned that you can run them on other desktops as well. When running them on something like KDE, however, there is the downside of all those processes open in the taskbar and the frameless nature of the applications themselves. They work just fine, but they lack the elegance of

Window Maker's dock that swallows the applications, thus making for a clean collection of windows into your system's soul.

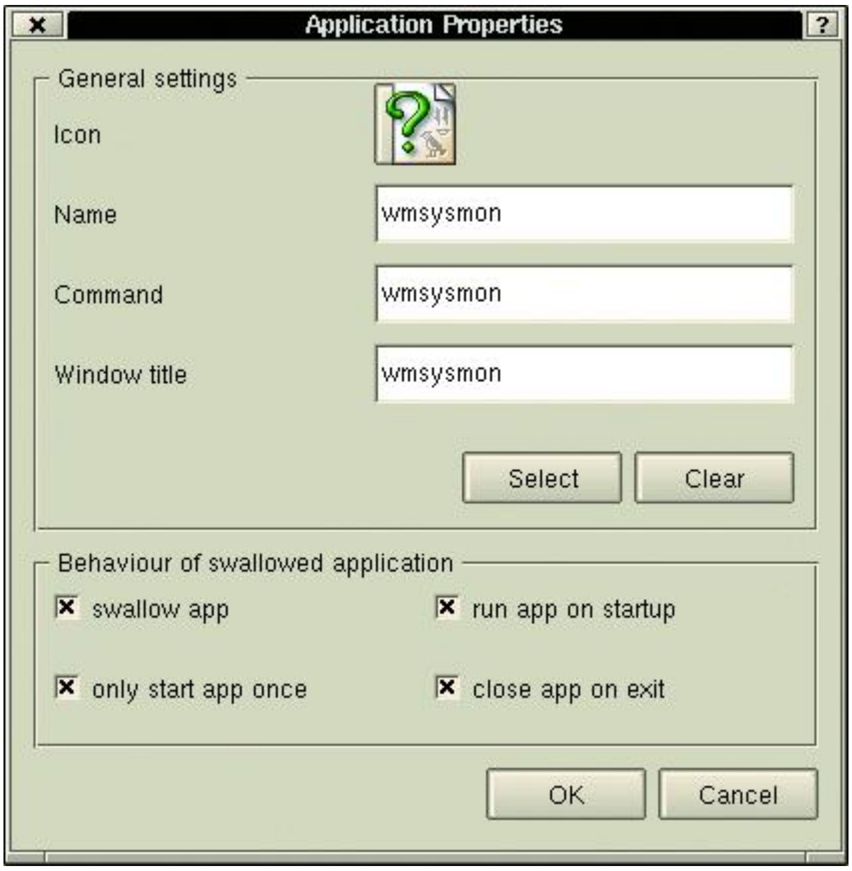
Thanks to Henning Burchart and the Kappdock, KDE users need no longer feel left out. This little tool sits quietly in the corner, waiting for you to attach and swallow all those Window Maker applications. Kappdock also places a little icon in KDE's icon tray. One click banishes all the running applications from your sight. One more click and they are back. I will show you how to work with it momentarily, but for now start by visiting Henning's web site at www.informatik.uni-oldenburg.de/~bigboss/kappdock.

Extract and build the source using the familiar extract, configure and make method:

```
tar -xzvf kappdock-0.44.tar.gz
cd kappdock-0.44
./configure
make
make install
```

Start Kappdock in the background by running it from the shell or launch it by pressing Alt-F2. You should see an innocuous little square on your screen with a small striped bar across the top and a black arrow to the right of that bar. (Notice as well the new icon in your KDE icon tray.) The bar actually is called the drag bar (because you can use it to drag your applications to any position on the desktop). I prefer to think of it as the dock itself. Clicking on the black arrow will cause kappdock to disappear from your desktop. Clicking on the tray icon will bring it back. So how do you attach applications? Please allow François to refill your glasses and I shall demonstrate.

Begin by right clicking on the single square you see attached to the dock. You can either select to edit the existing square or add a new one. If you right click the drag bar, you will get the option of either adding a new app or changing Kappdock's preferences. Some of these options include the position of the dock and the orientation of docked applications. To add an application, click New. You'll see a dialog box similar to the one in Figure 4.



Icon	
Name	wmsysmon
Command	wmsysmon
Window title	wmsysmon

[✓ swallow app] [✓ run app on startup]
[✓ only start app once] [✓ close app on exit]

Figure 4. Adding an Application to Kappdock

If you have an application already running (from the previous examples), you simply can click on the Select button. Your mouse pointer will change to a set of crosshairs. Move the crosshairs over to your running application, click and *voilà!*, your application is docked. Optionally, you simply can type in the information yourself. Have a look at Figure 5 for a well-stocked Kappdock.

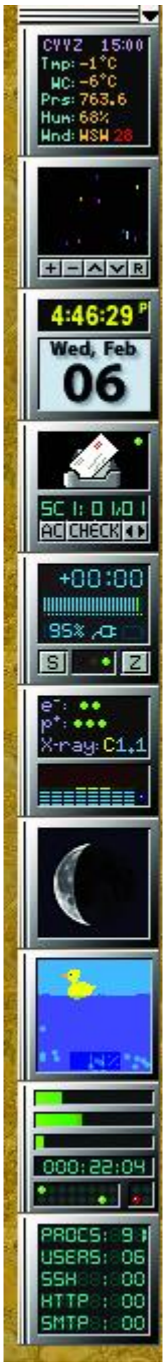


Figure 5. Busy Kappdock keeps KDE looking clean.

There you have it, *mes amis*. The clock (in this case, wmCalClock), she is telling us that closing time approaches once again. It is time to finish your wine and perhaps try out one last program. If you just can't seem to get your fill of dock applications, head on over to Ben Sinclair's Dock App Warehouse at www.bensinclair.com/dockapp.

This is a well-thought-out web site with tons of Window Maker dock applications. The graphical list of applications means that while you cannot judge an application by its graphic, you certainly can let your imagination go wild. There should be more than enough here to satisfy your appetite for quite some time.

On my own desktop, I run weather applications, clocks, graphical do-nothings, power monitors, moon, sun and space weather monitors—all these in addition to classic system resource monitors. The design of these little programs means they take up very little of your precious desktop space. Thank you once again, *mes amis*. Your visit to *Chez Marcel* is always a pleasure. Until next month. *A votre santé*! *Bon appétit*!

Resources

Marcel Gagné (mggagne@salmar.com) is president of Salmar Consulting, Inc., a systems integration and network consulting firm, and the author of Linux System Administration: A User's Guide, published by Addison-Wesley.

Understanding IDS for Linux

Pedro Paulo Bueno

Issue #97, May 2002

Pedro discusses the different types of intrusion detection systems and shows how to create signatures to identify attacks.

Do you feel your network is safe? Do you really know what is happening on your network right now? Once upon a time, there were network administrators who thought that the solution to their security was a simple firewall. In the past few years, we have verified that this is not true anymore. The need for some element that could alert and inform administrators about something strange in near real time resulted in intrusion detection systems (IDSes). In this article we discuss the types and models of IDSes: the host-based intrusion detection system (HBIDS), the network intrusion detection system (NIDS) and the new concept of hybrid-IDS. How to analyze the data generated and how to create signatures (the patterns that identify the attacks) also are discussed, as well as some examples of IDSes for Linux, like the open-source NIDS Snort.

What Is an IDS?

An IDS is a program that tries to detect strange packets and behaviors that may compromise a network. The first IDS was the host-based IDS, but the one that really got the market was the NIDS, the network-based. There is usually some software or appliance, called a sensor or agent, that has one or two network interfaces (as we will see later, it may work perfectly with one network interface), which works in promiscuous mode. In other words, it will catch all the packets that come to the interface and not just those with its particular destination IP address. In this way, the IDS can analyze all the packets that cross the network, check if they contain, for example, any suspicious strings and then decide how to perform a reaction, such as interacting with the firewall to create new rules to block the IP address, sending pager/e-mail alerts to the security administrator and so on. One important topic about the NIDS is where to deploy the sensor, inside or outside the firewall. I like to quote an interesting passage about this from SANS' GIAC Director Stephen Northcutt's book, *Network Intrusion Detection: An Analyst's Handbook*:

> An IDS before the firewall is an Attack detection and after the firewall is Intrusion detection....In a switched network, since we don't have broadcasting, we have two better options on deploying the NIDS, using a hub to force a broadcast or using a mirroring-port in the switch.

Where is the best place? We may have a long discussion about this since there are defenders of both points, but undoubtedly all agree that the best option is the use of sensors inside and outside the firewall.

IDS Models

To understand the IDS better, first we need to know how it works. Basically we have two models of IDSes: the misuse or signature-based model and the anomaly model.

The misuse or signature-based is the most-used IDS model. Signatures are patterns that identify attacks by checking various options in the packet, like source address, destination address, source and destination ports, flags, payload and other options. The collection of these signatures composes a knowledge base that is used by the IDS to compare all packet options that pass by and check if they match a known pattern. Later we will discuss a Nimda worm signature example in the Snort IDS.

The anomaly model tries to identify new attacks by analyzing strange behaviors in the network. To make this possible, it first has to “learn” how the traffic in the network works and later try to identify different patterns to then send some kind of alert to the sensor or console. The disadvantage of this model is that you will never know if your network has produced all types of behavior in the IDS learning phase, so it may cause a high number of false-positive alerts.

False-positives are false alerts produced by the IDS to inform of an attack when in fact it is just nonconfigured variables or an application that sent some packet to a different port than usual, instead of a real backdoor, for example. To solve this, the security administrator has to observe the alerts generated by the IDS for some time and then fine-tune it.

Host-based intrusion detection systems usually are located in servers and only detect events related to the machine in which it is installed. The main purpose of the HBIDS is to avoid changes that may compromise the machine and detect malicious queries. Examples of changes that can prove the importance of this kind of IDS are web defacement and rootkits installed in the machine to attack other machines.

Rootkits are packages installed in the compromised machine by the cracker, which contain files used to open backdoors, erase log files to hide their presence and replace binaries like ps and netstat, and also hide any dæmon or open port.

Besides this, the HBIDS has the function of trying to detect attacks before they happen, analyzing logs to point out strange behaviors and also detecting port scans.

Tripwire

Tripwire is an example of an HBIDS for Linux [see Michael Rash's Paranoid Penguin, *LJ* February 2002 for an open-source alternative to Tripwire]. It can be identified as an HBIDS because it fills in for the lack of file-integrity detection tools. With Tripwire, the user can define, in a configuration file, a set of files that he or she wishes to protect against changes, and then Tripwire uses a checksum of these files and attributes. In the case of any changes, it can send alerts to the system administrator. The default configuration file provides a good starting point, but the user also must customize it to reduce the chance of false positives. Pay special attention to the log files. It doesn't make sense to include the log files into the set of files that you select to be checked, since you know that they will grow as soon as any event happens, such as a login.

Tripwire can be used together with the cron scheduler dæmon. In this mode, users can automatize the process and define wherever they want to run it.

PortSentry

PortSentry [see also “PortSentry” by Anthony Cinelli on the *LJ* web site, /article/4751] is part of the Abacus Project, from Psionic Software, whose goal is to “produce a suite of tools to provide host-based security and intrusion detection free to the internet community”. It is an important kind of HBIDS because it detects packets addressed to the host and can be used with TCP Wrappers and iptables. This type of detection is useful because a port scan is often a precursor to an attack. PortSentry can detect TCP and UDP port scans, making you aware of other hosts that run a service in the scanned port. The next step is to verify for new patches or updates, or even configure it to create ACLs (access control lists) to block future connections from the host scanner, using TCP Wrappers. It also can create rules in the firewall, i.e., iptables, to drop everything from the host scanner. The following is an example of PortSentry alerts from Syslog:

```
Dec 9 03:03:17 mobile portsentry[701]: attackalert:
  TCP SYN/Normal scan from host:
  200.185.61.132/200.185.61.132 to TCP port: 111
Dec 9 03:03:17 mobile portsentry[701]: attackalert:
  Host 200.185.61.132 has been blocked via wrappers
  with string: "ALL: 200.185.61.132"
Dec 9 03:03:18 mobile portsentry[701]: attackalert:
  Host 200.185.61.132 has been blocked via dropped
  route using command: "/sbin/iptables -I
  INPUT -s 200.185.61.132 -j DROP"
```

Swatch

Swatch is a log watcher that observes the logs and alerts the security administrator about predefined strings found in the log file, i.e., /var/log/messages. In the example below, I created a very simple Swatch configuration file and chose to define the strings “snort” and “portsentry” and send the alert to screen in different colors (and with a beep) every time that it finds these strings:

```
watchfor /snort/
echo red
bell
watchfor /portsentry/
echo blue
bell
```

I also could ask Swatch to send an e-mail or execute a command when it finds something. As the result of the previous Swatch config file, I received these alerts:

```
Dec 9 03:22:53 flamengo snort[3268]: [1:1256:2]
  WEB-IIS CodeRed v2 root.exe access [Classification:
  Web Application Attack] [Priority: 1]:
  {TCP} 200.31.36.11:2153 -> 200.204.68.154:80
Dec 9 03:03:17 mobile portsentry[701]: attackalert:
  TCP SYN/Normal scan from host:
  200.185.61.132/200.185.61.132 to TCP port: 111
```

LIDS

LIDS stands for Linux intrusion detection system. It is a project that tries to give Linux some extra security features deployed as kernel patches. In these features we can include file and process protection and port-scan detection. The first two deserve a little more explanation. File and process protection will guard even against root superuser changes. This is very useful because when a cracker exploits a bug in your system, such as a buffer overflow, that person will have root access that permits him or her to do almost anything, such as install rootkits, change logs, erase your HTML pages, etc. With these features you can define ACLs to control files and include passwords to access/change them, avoiding changes from unauthorized users, even root. The same is valid for process because it will protect your system from altered binaries/dæmons. Another good feature is that it offers a port-scan detector in kernel space.

NIDS

Network intrusion detection systems are the kind of IDSes responsible for detecting attacks related to the network. One point of discordance is where it should be deployed. You may encounter network topology where it is before a firewall, and you may find it after a firewall. As I said before, there are good arguments for both; it depends on your needs. In these examples I will use the open-source Snort.

Snort

Snort was created by Marty Roesch and currently has over 1,000 rules used to detect attacks like simple port scans and even new attacks such as the SSH CRC32 exploit [see “Snort: Planning IDS for Your Enterprise” by Nalneesh Guar on the *LJ* web site at /article/4668]. One of the greatest advantages of Snort is its flexibility to create new rules on demand. Whereas with some IDS vendors you have to wait until they release new packages, you may customize and create signatures as soon as the attacks are exposed. One good example was the wu-ftpd exploit in mid-December 2001. Just a few hours after the release of the exploit, the Snort filter was released on security mailing lists. Snort also has the capability to interact with firewalls, i.e., Check Point FireWall-1, using the OPSEC feature or using other plugins to interact with Linux's iptables. Besides the fact that Snort has a large signature database and is mainly based on the misuse model, it offers a beta feature to introduce it to the anomaly model. This feature, called SPADE, does a statistical analysis of the data it gathers and tries to find out what the ordinary behavior is. As with many open-source applications, Snort has a lot of additional applications that you may want to use together.

One nice application from Silicon Defense is SnortSnarf, which creates HTML reports based on the data gathered by Snort.

Snort also works perfectly well with just one network interface card (NIC). Instead of other NIDSes, which need two NICs, one to gather the data and other to be used by the administration interface, Snort can work with one NIC in the promiscuous mode and also can be used to administrate it, inserting new rules or upgrading it.

Hybrid-IDS

More recently, another concept of IDS is becoming popular. It is the hybrid intrusion detection system. Marcus Ranum, founder of NFR (Network Flight Recorder, Inc.), wrote that “The shim-type hybrid IDSes are an interesting blend of the strengths and weakness of HBIDSes and NIDSes.” This means that it has most of the features of the NIDS but on a per-host basis. This has a lot of advantages, as it will try to detect attacks to the host exclusively, and the traffic that it will analyze will be only packets with the host destination IP address. The disadvantage of this kind of IDS is that it needs to be deployed in every host.

Prelude-IDS

Prelude is an example of a hybrid-IDS. It is divided into two different parts: the sensor, called Prelude NID, that is responsible for the packet capture and analysis, and the report server, used by the sensor to report an intrusion attempt. Prelude has an interesting feature that deserves some comments: the capability to read rules from Snort IDS. In other words, it has a complete rule set ready to use. From its web site, it is also capable of reading rules from any NIDS. Prelude was built with the cluster concept in mind. This explains the idea of deploying information into a different machine called a report server, which has the job of translating all the information received into a user-friendly format, such as HTML.

Understanding and Creating Signatures

As we learned before, signatures are attack patterns. It's important to understand how they work, so we can create them on-demand or when a new exploit is discovered. In our examples, we will see how Snort handles its signatures. In the second half of 2001 we observed new and powerful worms on the Net, such as Code Red, Code Red II and Nimda. When these attacks started to happen, Linux users (and I was one of them) felt very lucky because the worms mainly were attached to Microsoft's IIS (Internet Information Server). These worms had some particular patterns, for example, trying to access the cmd.exe file through Microsoft's IIS. By knowing this, we easily could create a Nimda Snort rule as mentioned in the section “IDS types and Models”:

```
alert tcp $EXTERNAL_NET any -> $HTTP_SERVERS 80
  (msg:"WEB-IIS cmd.exe access"; flags: A+;
  content:"cmd.exe"; nocase;
  classtype:web-application-attack; sid:1002; rev:2;)
```

Okay, but what does it mean? Snort rules are nothing more than sequential parameters divided in two sections that we use to inform Snort of what we want it to pay attention to. The first section is called rule header and includes everything before the first brackets. The first parameter in this header tells Snort what to do when the packet matches this rule—in this case, “alert” indicates that Snort will generate an alarm and then log the packet. The second parameter tells Snort what kind of protocol we want—in this case, just TCP. The next five parameters indicate the source IP address and port, direction of the packet, destination IP address and port. In this way, we know that a packet from any address outside of our network, with any source port, going to an IP address in our internal network at port 80 (usually web servers listen to this port) will be checked by the internal parameters of the rule, called rule options. The rule option section contains alert messages and information about which parts will be checked in the packet, and then with the result of this inspection, the appropriate action will be taken.

Rule options in our example:

- msg: "WEB-IIS cmd.exe access"—description of the alert.
- flags: A+—logical operator (+) to test all flags in the packet.
- content: "cmd.exe"—sets the specific content (cmd.exe) in the packet payload.
- nocase—will match the specified string with case-insensitivity.
- classtype: web-application-attack—classification of the alert.
- sid:1002—Snort rule ID.
- rev:2—rule revision number.

In the *Snort Users Manual* you can find more than 30 rule options that you can use to satisfy your needs. Too complicated? No, it is not! Let's try to create a simple rule to alert any porn web access attempt from your network using the few rule options above:

```
alert tcp $INTERNAL_NET any -> $EXTERNAL_NET  80
  (msg: "Web Porn Access Attempt"; content:"Free porn";
   nocase; flags:A+);
```

Analyzing the Data Generated

A port scan to a service like portmap (port 111), which is known to have various exploits, would be alerted by PortSentry:

```
Dec 9 03:03:17 flamengo portsentry[701]: attackalert:
  TCP SYN/Normal scan from host:
  200.185.61.132/200.185.61.132 to TCP port: 111
```

Learning how to interpret log files is one of the most important things that an intrusion or security analyst must learn in order to decide what action to take in a given situation. The excerpt from the PortSentry alert above was obtained from the syslog file. This alert states that on December 9 at 03:03, the host called flamengo, which has PortSentry installed, detected an SYN-flag Normal port scan in the TCP port 111 which, in general, runs the service portmap, from host IP 200.185.61.132.

Conclusion

A firewall is a primary security element in a network, but it will not detect attacks on a service that is already opened, such as an attack to your DNS or web server. An IDS by itself will not solve all your problems as a security element, but if you customize it for your needs, it certainly will help alert you to strange behaviors and unauthorized attempts to your host or network. With this information, you should contact the administrator of the network in which the intrusion's IP is located and then inform them of what is going on. Being in contact with the security community is also the best way to keep up to date on new attacks and the signatures to detect them. Be aware—install an IDS!

Resources

email: pbueno@opencs.com.br or bueno@ieee.org

Pedro Bueno (bueno@ieee.org) is a former data engineer from Lucent Technologies and currently is a security engineer at Open Communications Security. He also contributes at Best Linux as a volunteer, and his favorite hobby, besides soccer, is analyzing the alerts generated by Snort.

Archive Index Issue Table of Contents

Advanced search

Tippett Studio and Nothing Real's Shake

Robin Rowe

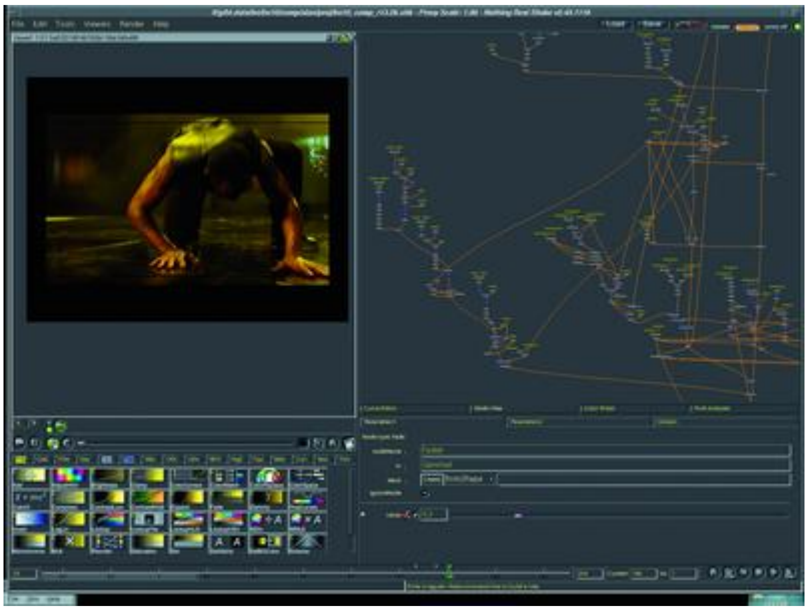
Issue #97, May 2002

Robin describes how Shake on Linux is used to composite special effects.

Video compositing software is used by motion picture studios to combine special effects or animation elements into film sequences. Compositing software may be thought of as doing for moving images what tools like the GIMP and Photoshop do for still images. Nothing Real's Shake seems to be the most widely used high-end compositing package today. Shake runs on Linux, Windows and IRIX, and Apple has just confirmed rumors that it has acquired Nothing Real and Shake.

Tippett Studio specializes in scary effects, such as bugs and creatures. Let's visit Tippet Studio in Berkeley, California for a look at how they use Shake on Linux.

As a visual effects and animation studio for feature films, Tippett Studio has won Academy Awards for visual effects work on *Return of the Jedi* and for *Jurassic Park* and was nominated for effects in *Starship Troopers*, *Dragonheart*, *Willow*, *Dragonslayer* and *Hollow Man*. Tippett Studio currently is creating effects for the vampire movie sequel *Blade II* (release March, 2002). “We've been using Shake on Linux since *Evolution*, about a year and half”, says compositing supervisor Alan Boucek. “I'm running Shake on a dual Athlon. What's exciting is that it is so fast. I can mostly work on my own machine, without going to the renderfarm.”


Shake being used to composite a scene with Wesley Snipes in *Blade II*. Window manager is MWM for continuity of look and feel with SGI IRIX workstations.

Boucek adds that his compositing crew was the first in the studio to move to Linux. “Once we got Shake up and running, the next task was to get our pipeline on Linux”, he says. “For most of our 3-D rendering, we use Pixar's RenderMan. Grey renders are straight out of Maya, but animators run their stuff through a simple Renderman pass for daily review.”

Shake uses a tree-graph interface to manipulate effects. Each input is passed through nodes that represent filters or transforms that create the final composite images. Complex effects take longer to render, and time is always at a premium at a studio. Special effects artists often work at quarter resolution in near real time with Shake on a workstation, then later batch process the effects at full resolution on the server renderfarm.

Two years ago when Tippett needed new computers for compositing they chose $5,000 US SGI x86 PCs over $24,000 US SGI Mips Octanes. “We bought SGI PCs at first, but after those were discontinued we started building our own machines from scratch”, Boucek says. “We've saved a lot of money on machines, but a lot of that is taken up in providing our own support.” Besides PC and SGI computers, they have many Macintoshes. And, some of the PCs are running not Linux but Windows versions of Shake and 3-D modeling package Maya. Boucek prefers Linux:

> We can't tie NT into our renderfarm. When we go home at night the Linux boxes keep working. The NT workstations are dead—can't do double duty as renderfarm servers. We still have a lot of SGIs. Our Linux software conversion is just completing.

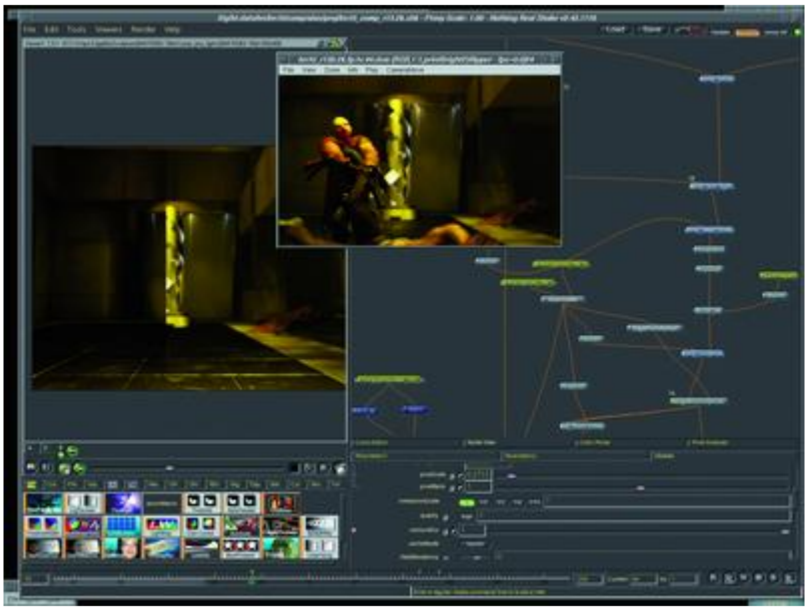
“Just being able to talk to who wrote the original code is a tremendous advantage”, says Director of Technology Christian Rice.

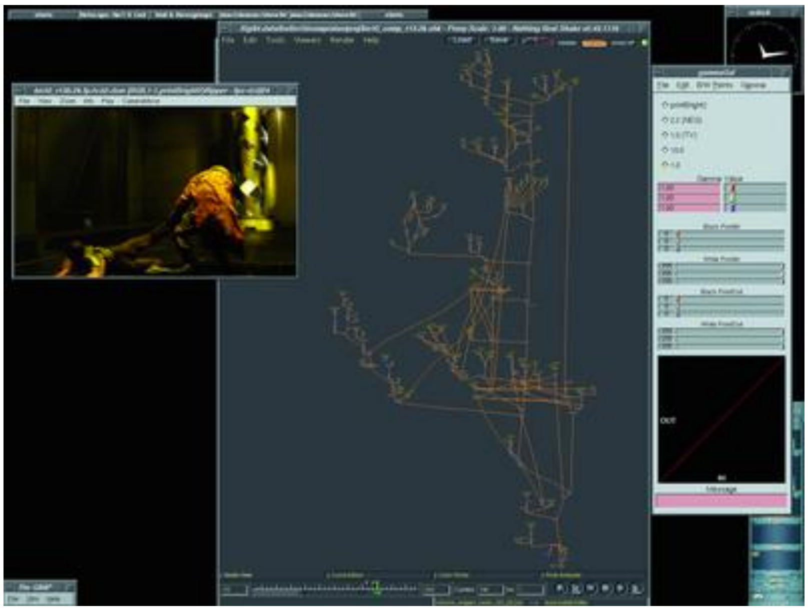
> With Linux we can have a personal dialogue with the author of a package in a noncorporate manner. We've overcome several hurdles in that manner. It does create more responsibility for us though, and more uncertainty. Just because a search of the Web turns up nothing doesn't mean something can't be done. Talking directly to the developer often gives us an insight into special features that we wouldn't otherwise be aware of, not to mention special features being implemented to accommodate us.

“We have ten programmers here in a crew of 150”, says Boucek. “They work on 2-D, 3-D and pipeline support. A lot of what they do takes care of the handoff [integration] between tools not especially intended to work together.” These programmers support about 25 artists doing 2-D work, and about 75 doing 3-D. Although Shake has many built-in effects, studios often create their own that may become a look or specialty associated with that studio. As with the GIMP and Photoshop, programmers may create plugins for Shake written in C or C++.

“Writing plugins for Shake at the beginning is very hard”, warns Software Engineer Qin “Jean” Shen. “There isn't complete developer's documentation. I would copy a simple example from the dev kit then send a lot of e-mail questions to Nothing Real as I worked on extending it. They are good about replying and helpful.” Shen explains that Nothing Real tries to write good documentation, but because each plugin serves a special purpose it is hard to be comprehensive. Plugin developers must take care to note when something is needed in memory so that Shake can automatically fetch it. “Shake is a scanline renderer”, notes Boucek. “It doesn't read the entire image in memory at once. That's what makes Shake so fast.”

Shen disproves the theory that you can't get an exciting job in motion picture software development right out of college. Although she already had an offer from nearby ILM, she choose Tippett because as a smaller studio it offered her more involvement. “When I started in 1999 we were still using Alias|Wavefront Composer. I wrote plugins for that, and then Shake after we switched.” Shen works on development of Flipper (a flipbook movie player) and GammaGal (a tool to adjust monitors), both internal tools. “Flipper and GammaGuy had been written using Motif and IrisGL. I rewrote those in OpenGL and FLTK, and GammaGuy became GammaGal.”


Shake preview on left during split screen transition. Flipper flipbook player on right. Icons (lower left) are for in-house plugins.


Flipper on left, GammaGal on right. Flow graph in Shake to build one entire movie shot (scene).

“An important feature in Photoshop is gamma-level adjustment”, explains Software Developer Darby Johnston. “GammaGal allows us to do something similar very quickly on IRIX and Linux.” Because film has a greater dynamic range than do video monitors, users constantly must play with levels to see details in dark or light areas. Artists working in film play with gamma like an artist will play with magnification while retouching a still image.

“To describe our Linux efforts converting our IRIX tools as porting is too strong a word”, says Johnston. “For the most part, stuff just compiles.” Johnston's work involves mostly traditional C-style coding, without threads or shared memory.

Johnston is part of a team writing glue code for integration, an image-processing suite (e.g., blurs, crops, scales) and batch tools.

“There still are growing pains with Linux”, notes Boucek. “A lot of development in Linux seems to be about beating Microsoft. But, what we need is a reliable platform that runs our tools.” When Tippett started with Linux, the NVIDIA drivers were still in beta and had problems. “Sometimes it wouldn't draw a line in glLines correctly”, says Shen. “I used glStrip instead as a workaround. We had many instances like that.” Boucek adds, “A lot of instability in the early months with Linux was tracked down to the NVIDIA drivers, and Maya is still pretty unstable for us on Linux using NVIDIA GeForce2 and Quadro2 graphics cards.” Despite problems, NVIDIA drivers have been key in the transition to Linux.

“The performance of the proprietary NVIDIA drivers has been important in spurring our move to Linux, although we can tell that the optimization effort is toward *Quake* more than 2-D”, says Johnston.

> We recently tested XiG with an ATI RADEON 7500 on a 1.4GHz Athlon and found it much faster for 2-D. Flipping frames at 1,920 × 1,200, the NVIDIA drivers barely hit 30fps, but the XiG drivers were over 40. However, in 3-D the NVIDIA drivers were perceptibly faster in Maya.

“We're trying hard to get Linux on everyone's desktop”, says Boucek. “But, we're still not there because of stability problems with Maya on Linux.” Boucek wants to see Linux and Linux drivers continue to get more stable and for performance to improve.

Square is another movie studio that has done stunning work using Shake, on the motion picture *Final Fantasy*. “We started with IRIX four years ago”, says Visual Effects Supervisor James Rogers. “Our Linux renderfarm was brought up in the middle of production.” Square used their existing IRIX boxes for workstations, then sent data through a custom render manager to Shake batch render on the Linux renderfarm. “We started testing Linux workstations at the end”, says Rogers. Square recently announced it will cease movie operations and close its offices in Hawaii as of March 29, 2002.

To install Shake we download three files (55MB) from the Nothing Real web site: the license manager (lmutil.Z, 184k), the application (shake-linux-v2.46.0116.tar.bz2, 28MB) and the tutorial (shake-tutorial-v2.46.0116.tar.gz, 26MB):

```
gunzip lmutil.Z
tar xvfj shake-linux-v2.46.0116.tar.bz2
tar tvfz z/shake-tutorial-v2.46.0116.tar.gz
tar xvfz z/shake-tutorial-v2.46.0116.tar.gz
./lmutil lmhostid
```

Just to double-check, we used the tell switch with tar to see what it would do before doing an extract.

The lmutil program echoes a 12-digit license-manager host ID. This host-locked ID code must be sent to Nothing Real in order to receive a two-week trial key code (key.dat). Copy the e-mailed key to the specified directory, start X and run Shake:

```
cp key.dat shake-v2.46.0116/keys/
startx
shake-v2.46.0116/bin/shake
```

Unless you already know Shake, two weeks isn't much time to learn about this complex tool.

As this story is being written, the future of Shake is unclear. Apple confirmed rumors of the Nothing Real acquisition in a short statement released on February 6, 2002. Tippett Studio's Boucek notes:

> They have a core renderer that's faster than anybody's. That could be valuable in QuickTime or Final Cut Pro. Shake is such an important tool for us on Linux we have to be concerned about what would happen if it becomes OS X only. But, it would be good if it was cheaper so we could afford more seats. In general, we're excited about OS X. We have a lot of Macs here.

With BSD-based OS X, the Apple platform gained NFS and other UNIX features that help it integrate into movie studio pipelines.

Apple, in a two-sentence statement confirming rumors that it had purchased the company, only revealed that it “plans to use Nothing Real's technology in future versions of its products”. In 2001, Nothing Real had announced that it would port Shake, which runs on Linux, Windows and IRIX, to Apple OS X. However, Apple is uncommitted as to whether it will release that promised OS X version of Shake.

Since Apple has never continued development of a GUI application for another OS after an acquisition, it seems this may be the last version of Shake for Linux workstations. However, the Shake renderfarm server software will probably continue to support Linux indefinitely. There's a precedent with the Apple Linux Darwin QuickTime server software. If future versions of the Shake GUI only run on Macintoshes, the extra hardware cost for new Macs seems a nonstarter to studios. But, Apple could sidestep that by lowering the cost of Shake. At nearly $10,000 US per seat, Apple could throw in a loaded $2,000 US iMac for free.

Some studio managers are feeling spooked by Apple's total silence on their plans for Shake. But others, such as Boucek at Tippett Studio, express optimism that Apple has something to offer to the professional motion picture editing market.

Because that's been most of the market, Linux companies are more focused on servers than workstations. Despite not really being designed for that, Red Hat is the de facto standard for Linux workstations. Some users say server-oriented Red Hat is not a good distro for the desktop. Red Hat CEO Robert Young admits being booed more than once for telling Linux crowds, “Linux will never be successful on the desktop.” Competition from Apple OS X may spur Linux to become more user-friendly. Thanks to OS X, Apple system Software Product Manager Ernest Prabhakar was able to announce at USENIX BSDCON in February 2002 that BSD is now three times more popular than Linux on the desktop.

Apple may shake up the motion picture effects business. Steve Jobs is the CEO of both Apple and Pixar. Or, up-and-coming competitor Silicon Grail's RAYZ could be a big winner as a result of the Apple's Nothing Real acquisition. Many studios are taking a hard look at RAYZ due to the uncertainty surrounding Apple's plans for Shake. We'll take a look at RAYZ on Linux in detail here next month.

Resources

Cat Sequence

email: Robin.Rowe@MovieEditor.com

Robin Rowe (robin.rowe@movieeditor.com) is a partner in MovieEditor.com, a technology company that creates internet and broadcast video applications. He has written for *Dr. Dobb's Journal*, the *C++ Report*, the *C/C++ Users Journal* and *Data Based Advisor*.

Advanced search

Pessimism or Realism?

David A. Bandel

Issue #97, May 2002

This month David considers antitrust legislation and the possibly bleak future of Linux.

Well, those of you who thought the days of the robber barons were over haven't been paying attention—looks like Microsoft has gone from the software industry into politics. They steamrolled and bullied, and when that didn't work they bought out their competition. And they just found out it works like a charm in political circles too, increasing their $150,000/year in political contributions to over $6.1 million (and buying a lot of influence with that money). I hate to say it folks, but hundreds of millions of our tax dollars have been wasted in antitrust legislation just to subsidize even stronger Microsoft control of the software industry.

But Microsoft leaves no stone unturned. They "give away" software to schools. The price tag is if you want Microsoft for free, you can't run/teach any other operating systems—looks like our children will be saying the pledge of allegiance to Microsoft. And it's not just happening in the US, it's happening everywhere Microsoft sticks its foot in the door. I am certain these deals have been struck with Canadian schools, Australian schools, even South African schools. Microsoft is bound and determined that in five years or less, the only operating system in the world will be theirs. After the incredibly expensive and embarrassing fiasco called an antitrust trial, their stranglehold on the industry will be stronger in the coming years than anyone's nightmares imagined. Our children will learn Microsoft programs and operating systems exclusively. OEMs will revert to offering only Microsoft because that's all folks will order. Because when you get one order for a Linux system and over 10,000 orders for Microsoft, it doesn't pay to preload anything else (and unless a court orders OEMs to offer systems with no OS, they *will* come preloaded).

I also foresee many more useless lawsuits. So I declare the winners of the US antitrust case against Microsoft to be first, the lawyers and second, Microsoft.

The losers are everyone else (unless you happen to be a Microsoft stockholder). In another ten years Linux will be just another memory. Pessimistic? No, realistic. Our courts and elected officials have failed in their duties. And until yet another antitrust lawsuit is won against Microsoft (it will be the third time), things will only get worse. But perhaps the third time (if it ever happens) will be the charm, and innovation and freedom of choice can return to the software industry. I'm not holding my breath. Good guys too often finish last.

Axel www.lintux.cx/axel.html

This utility claims to be a download accelerator. I don't want to get into the debate about accelerating downloads, but the program can run single or multithreaded (opening *X* number of connections to a server). Whether the downloads are faster or not, and in which mode, is for you to decide. What I know is that this utility is very small compared to some others. Requires: libpthread, glibc.

axelq electron.its.tudelft.nl/~hemmin98/axelq.html

The Axel utility mentioned above has no queue to permit you to store URLs for later download. This adjunct allows you to do that and run Axel at a later date against a list of URLs. If you want to run Axel, then you'll also want this utility. Requires: /bin/sh.

integrit integrit.sourceforge.net

This particular application compares itself to the likes of Tripwire and AIDE. As another tool in monitoring your system, it works well enough. It is simple to use and can be set up to check your system against changes to key files or directories. Requires: (integrit is built statically, so requires no runtime libs).

nmbscan gbarbier.free.fr/prj/dev/#nmbscan

If your network has a mix of Windows and Linux/UNIX, you can use this to take a quick look at the local network. It is a simple shell script that makes use of the network tools on your system to identify Windows hosts, Samba servers and provide as much information as possible regarding the Windows side of your network. Requires: /bin/sh, smbclient, nmblookup, arp, host, ping.

myPhile www.rni.net/~geoffm

This is an extremely simple, yet versatile front end to just about any MySQL database to which you care to connect. Newly installed, it creates a small database address book. It is easily modified for your own purposes and is usable by a small company with few changes and almost no training. It's worth a look if you need a quick and simple address book for the masses. Requires: web server with PHP, web browser.

Jmol www.openscience.org/jmol

If you're a college student with chemistry courses that require you to model molecules, this program is great. I remember having to do such modeling (not a few years ago) with a tinker-toy set that looked like jacks with plastic straws. But this program will let you do everything short of touch the model. It comes with many samples, or you can look for the compound you need on the Web. This program supports a lot of different format files for chemical compounds. Requires: Java.

GRPN lashwhip.com/grpn.html

This month's choice from three years ago was frustrating. Keystone was sold to WhitePajamas and abandoned, and most other choices showed little, if any, improvements. While not a favorite of mine from the past, as some others have been, I choose GRPN. Probably not a lot of us remember Reverse Polish Notation (RPN) anymore, but it was used in many calculators, and if I remember correctly, almost all scientific calculators. If you like RPN, then this calculator is for you. It handles general math functions, exponential and logarithmic functions, and trigonometric functions. Requires: libgtk, libgdk, libmodule, libglib, libdl, libXext, libX11, libm, glibc. Until next month.

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

Archive Index Issue Table of Contents

Advanced search

Embedded Linux Targets Telecom Infrastructure

Rick Lehrbaum

Issue #97, May 2002

Reach out and...Tux someone! Rick reveals the many ways Linux is entering the telecom market.

The 22-member nonprofit Open Source Development Lab (OSDL) used the occasion of LinuxWorld 2002 in New York to announce a major new Linux initiative aimed at the telecommunications infrastructure market. OSDL has created the Carrier Grade Linux working group that is chartered to provide “vision and guidance” and to “encourage the development of whatever commercial and open-standard components are needed on top of Linux to implement required platform functionality” for the telecommunications industry. The Carrier Grade Linux working group is made up of some heavy hitters in the telecom market, including Alcatel, Cisco, HP, IBM, Intel and Nokia. Linux vendors MontaVista Software, Red Hat and SuSE are also members of the working group.

OSDL has articulated several reasons why the telecommunications industry needs a new standards-based, carrier-grade operating system platform, and why Linux is ideal to serve as its basis:

- Networks are converging for multimedia communication services.
- More bandwidth and new architectures are needed.
- Open-standards-based, off-the-shelf software components are needed to improve time-to-market of new services.
- An open-standards-based approach reduces development cost/risk of products for the new architectures.
- Linux is the fastest-growing, general-purpose server operating system.
- Fragmentation of the Linux kernel must be avoided for both data center and communications market segments.

The group's initial focus is to collect market requirements and specify the architecture for a Carrier Grade Linux platform (see Figure 1) and also to encourage development of third-party components on top of Linux that implement the required functionality of the platform.

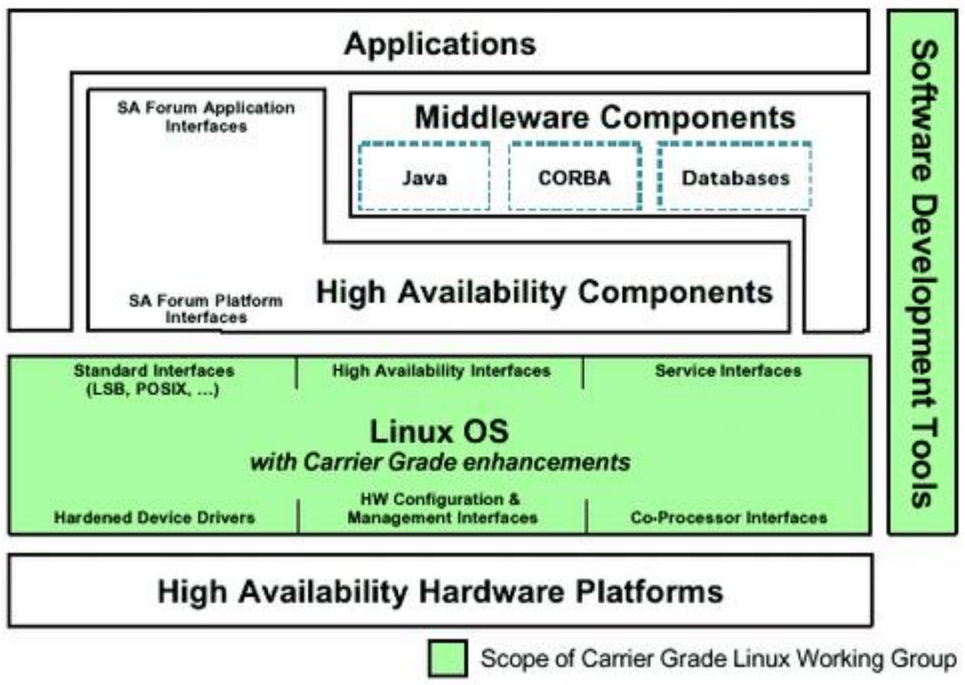

Applications
SA Forum Application Interfaces
Middleware Components
Java
CORBA
Databases
SA Forum Platform Interfaces
High Availability Components
Standard Interfaces (LSB, POSIX, ...)
High Availability Interfaces
Service Interfaces
Linux OS
with Carrier Grade enhancements
Hardened Device Drivers
HW Configuration & Management Interfaces
Co-Processor Interfaces
High Availability Hardware Platforms
Software Development Tools
Scope of Carrier Grade Linux Working Group

Figure 1. Carrier Grade Linux Architecture. Copyright (c) 2002, OSDL. Used with permission.

Other indications of the march of Linux into the telecom market are seen in the following recent headlines:

- **HP: "Linux Is the OS of the Future in Telecom"**: Also this year at LinuxWorld in New York, HP unveiled a range of new Linux-based products and services targeting the internet infrastructure, telecommunications and network equipment provider markets. They include a family of Linux-based, carrier-grade servers and a developer's kit for HP Opencall software. The new telecom-oriented server family will be powered by Carrier Grade Linux, when available. According to Mark Butler, HP operations manager for telecom systems operations, HP is strongly supporting Linux as the OS of choice for the telecom market. "Linux is the operating system of the future in the telecom sector", Butler said. "HP is leading the advance of Linux in the telecom market."
- **Motorola Targets "6NINES" with New HA Linux Platform**: Motorola Computer Group also is promoting Linux into telecom infrastructure. They recently announced the latest version of their telecom-market Linux platform, HA Linux 3.0, which Motorola claims implements "significant steps toward providing the key operating system features required for 6NINES-availability, or the equivalent of 30 seconds or less of downtime annually". Motorola says to achieve that level of uptime, you need

properly designed hardware, not just the right software; Motorola provides HA Linux as an OS for its carrier-grade CompactPCI systems.

- **Nokia Unveils Linux-Based Platforms for All-IP Mobility Networks**: At the 3GSM World Congress in Cannes, France, Nokia announced a new Linux-based platform technology for what is being called “All-IP” mobility networks. The first products based on the new All-IP technology are to be Nokia's open, carrier-grade FlexiServer and FlexiGateway platforms. In January 2002, MontaVista Software announced that they had been chosen by Nokia Networks to help develop Nokia's All-IP infrastructure.

Linus Says “kyllä” to Preemptible Kernel Patch!

The preemptible Linux kernel patch, originally introduced by MontaVista Software and more recently championed by Robert Love, has now been officially merged into the main Linux development-kernel tree, starting with kernel version v2.5.4-pre6 [see Rick's interview with Love in the April 2002 issue of *LJ*].

Although this enhancement came about as a means to provide faster responsiveness of Linux for industrial and embedded control applications, the benefits will be, in the words of Love:

> ...a means to an overall better system. Besides the traditional markets for low latency—audio/video, specialized embedded/real time, etc.—a preemptive kernel can benefit any interactive task. The result is hopefully a smoother, more responsive desktop.

New Linux PDA Being Developed in Bangalore

A Bangalore, India-based development company, perceived an opportunity between the high-end, high-cost Pocket PC PDAs and the low-end, low-cost Palm PDAs and created a new PDA called Kaii (which means “hand” in Kannada, the local language) to fill that gap. According to an article published in the *Bangalore Times*, Infomart (www.infomart.co.in) intends to sell the new Linux handheld devices for around $200 US with a 320 × 240 pixel monochrome LCD and for around $300 US with a TFT color LCD.

Infomart is trying for a high level of compatibility with Sharp's new Zaurus PDA, from both a UI and application software perspective. To accomplish this, they used the identical software stack: Lineo's Embedix Plus, Trolltech's Qt/Embedded and Qtopia, Opera's browser and Insignia's Jeode JVM. In an effort to minimize costs, however, the Kaii uses a 160MHz Hitachi SH3 processor and offers a soft (on-screen), rather than physical, keyboard.

Figure 2. Kaii Linux PDA

Other Kaii features include 32-128MB RAM (depending on version) and 32MB of either ROM or Flash solid-state storage memory, a USB interface with both device and host functions, an RS232 serial port, an IrDA interface and expansion slots for both CompactFlash Type II and MMC cards. Infomart plans to make use of the USB host capability to add external peripherals such as keyboards or specialty interfaces, making the Kaii useful for more than just traditional handheld PDA purposes. The company is looking for global manufacturing and marketing partners.

Question: Could this Zaurus-compatibility be the start of a trend among Linux PDAs?

For the latest information on Linux PDAs, be sure to check out LinuxDevices.com's “Linux PDAs Quick Reference Guide” at www.linuxdevices.com/articles/AT8728350077.html.

Rick Lehrbaum (rick@linuxdevices.com) created the LinuxDevices.com and DesktopLinux.com web sites. Rick has worked in the field of embedded systems since 1979.

Archive Index Issue Table of Contents

Advanced search

It's Elemental—Natural Advantages

Doc Searls

Issue #97, May 2002

Doc contemplates the geology of the West as a metaphor for Linux's role in the infrastructure of the civilized world.

I'm writing from the business class cabin of a United 777 en route from Chicago to Los Angeles. To my left is an LCD display that pulls out of the armrest. On screen is a map that tells me what I'm seeing out the window here on the right side of the plane.

The map scrolls a series of views, like you get in a flight simulator program. Right now it says we're heading toward the Rockies across Fort Collins, north of Denver, Colorado. About 30 miles to the right is Cheyenne, Wyoming, a flat pattern of streets and buildings surrounding the runways of an airport. The town spreads around a convergence of highways and railroads, drawn in faint lines across a landscape worn flat by almost constant winds. The story of the land's geology is told again by snow, which appears to be swept poorly by a gigantic broom, brushing northwest to southeast across endless stretches of ranches and farms.

We pass over the Front Range, the Medicine Bows, the Sierra Madres. In the distance I see the Big Horns, the Uintas—all rough features embossed on the sky above by forces below. Not many millions of years ago much of the scene out this window would have been as flat as Nebraska, but as the mountains came up under the plains, the soft soils were “de-roofed”, perhaps more by wind than by the rivers whose work is far more obvious. Even from up here you can see why they say a Wyoming weathervane is an anvil on a chain.

I take more than a passing interest in all this because I've been steeping myself for the last several years in the works of John McPhee, who does for geology what Shakespeare does for love—except the virtues of geology are not so self-evident. To McPhee, no rock has a story too dull to tell, which he proves, page after page, book after book. McPhee's series of books on American geology were only slightly condensed in 1998 into one fat work titled *Annals of the Former World*. It won the Pulitzer it deserved. My favorite McPhee book is *Rising from the Plains*, which weaves two tales into one: the story of Wyoming's deep past and of geologist David Love, who has lived through the last 89 years of it. “A geological map is a textbook on one sheet of paper”, McPhee writes. And David Love, who grew up on a hardscrabble ranch in the very center of the state, was the primary author of both the 1955 and 1985 editions of the Geological Survey's Wyoming maps. He researched them mostly on foot and guesses he has spent a quarter of his life sleeping outside.

What draws me to people like McPhee and Love is the sense of grounded perspective they give us on a topic that should become increasingly fundamental as both Linux and the computer industry mature. That topic is infrastructure—or, to use a label I prefer, interstructure. I'm talking here about our base-level computing and communications environments. Linux is down there, sitting on top of the deeper and more universal environment we call the Internet. It's infrastructural stuff. Every piece of code we add or change lithifies into solid material we use to build the civilized world.

In geology the term *competent* refers to rock that's dependable. You can build a house on it or with it, and you can trust that it won't break if you climb a steep surface of it. It also has nothing to hide about itself. The same goes for code. Infrastructural code is naturally competent. It is also both open to examination and improvement. The intellectual and creative processes by which we improve infrastructural code are no less natural than the geological forces that turn granite into gneiss, limestone into marble and peridotite into serpentine.

Competence has another aspect. Here's David Love: “Human environment, good and bad, starts with the rock, coupled with the other two major necessities, water and air. Ruin one of these three basic essentials and humanity is in deep trouble.” Now substitute “Internet” for “rock” and “computing” for “humanity”, and Love's point starts to come home.

Infrastructure is what we depend on. And because it is naturally common and abundant, a large number of us understand how it works and what it's good for.

Explaining why the government of China decided to create its own Linux distribution, Red Flag, Matei Mihalca, head of internet research, Asia-Pacific for Merrill Lynch, points to Linux's “transparency”. This is the inherent infrastructural advantage Linux enjoys over Windows. Even if Windows becomes 100% reliable, the reasons why will remain opaque as long as the source code remains closed. Build all you want with it, but don't ask it to serve as bedrock.

This doesn't mean there's no room for commercial developers in the business of making infrastructural bedrock. Dave Winer, the commercial developer behind SOAP and XML-RPC (both open-source protocols) says, “Ask not what the Internet can do for you, ask what you can do for the Internet.” Constantly improving Linux is one answer to that question.

Now the little map tells me we're starting to head past Las Vegas, which looks like a waffle pattern etched almost decoratively on the flat desert floor. In *Basin and Range*, John McPhee explains that the mountains of Nevada were mostly formed by extension: the distance between Salt Lake City and San Francisco is increasing, opening up. This started recently, in the Miocene, only about eight million years ago. The crust here has been spreading and breaking apart, and the heavier edges of each block have been sinking into the mantle while the lighter edges have floated up to form mountain ranges. Between them basins have opened and filled with erosional debris.

Many of Nevada's basins were recently also at the bottom of Lake Lahontan, a body of water as blue and subarctic as any in northern Canada. Many of today's ranges were islands in Lahontan's midst. Lahontan evaporated as the Sierra Nevada pushed out of California's deeps, casting a rain shadow to the east and starving Lahontan and its surrounding lakes of water. The last ice age ended ten thousand years ago, but the Earth continues to warm, leaving vast dry puddles where Lahontan and other Nevada lakes used to be. Las Vegas now reposes, oblivious to the coming deposits of mountain ranges eroding all around it.

As we begin our descent into Los Angeles, I find myself thinking that in the long run—in the period required for computing infrastructure to lithify—the real fight will not be between Linux and Windows, but between those who respect the nature of infrastructure and those who don't. Those who respect it see the Internet is the only platform worthy of the noun. Those who don't respect it see the Net as yet another plumbing system. Our fight with them will be over regulation intended to protect business models that require control over a plumbing system that the Internet's efficiencies threaten to obsolete. But we'll win, because nature will be on our side.

email: doc@searls.com

Doc Searls is senior editor of *Linux Journal*.

The Role of Standards in Open Source

Lawrence Rosen

Issue #97, May 2002

And, on how new standards are often compatible with free and open-source licensing.

Without standards, the Internet would become a Tower of Babel. Our freedom to speak what we wish depends fundamentally on our agreement to speak the same languages. Standards are the common linguistic foundation on which we build our diverse world. To be useful for software, a standard must be available worldwide and be free of encumbrances that prevent its widespread adoption.

Consider the implications for the owner of intellectual property (e.g., the owner of a patent or copyright) who wishes to promote that property as the basis for an industry standard. Or consider the interests of a developer of industry standard software who learns that another person's intellectual property blocks the implementation of the standard. Is private intellectual property compatible with industry standards in an open-source world?

This is not an academic question. Standards organizations everywhere are trying to decide how to incorporate private intellectual property into the framework of the Free Software Foundation guidelines and the Open Source Definition, under which source code must be published and the software must be available for free copying, modification and distribution.

Patents pose the greatest threat to standards and their implementation in open-source software. Any person who owns a patent containing claims that are essential to the implementation of a standard can prevent you from making, using or selling products that implement that standard.

I won't bother defining the phrase “essential claims” here, but consider the effect if the only technically feasible or economically practical way to implement a standard requires the use of patented technology. Since the law generally doesn't mandate compulsory licensing of patents and doesn't define “reasonable” royalties, the standard may be effectively off-limits for those who can't afford to pay or to design around the blocking technology.

Many people complained publicly when the World Wide Web Consortium (W3C) tentatively proposed a patent policy that would allow adoption of web standards based on patented technology for which reasonable and nondiscriminatory (RAND) royalties could be charged. The Free and Open Source communities argued that such royalties—even if they are “nondiscriminatory”, as between rich and poor—are not compatible with software that is distributed with source code under licenses that permit free copying, modification and distribution. As a result of that public outcry, the W3C patent policy currently is being redrafted. By the time this article appears, a new draft patent policy should be available for public comment at www.w3c.org.

One solution to this patent problem for standards is to require that owners of intellectual property license their patents, royalty free, for use in implementing industry standards. Members of organizations such as W3C agree to do just that (under certain conditions that they describe on their web site). Not all standards organizations have such policies. Implementers of standards should verify, by reviewing the patent policies of the organizations that promulgated the standards, that there are no known patent obstacles to the implementation of those standards.

Even when patents are licensed for implementation of the standard, the patent license may not be compatible with the license under which the resulting software will be distributed. For example, some patent license provisions typically say that the patent is licensed only for implementation of the specific standard (a “field of use” restriction).

The GPL is not compatible with patent licenses that are restricted as to field of use. GPL software must be free so that anyone can create derived works, including derived works that are used for other purposes. (Hackers say that the code must be available for “forking” and for “reuse” in other applications.) Patent licenses with field-of-use restrictions run afoul of section 7 of the GPL, which reads in part: “[If] a patent license would not permit royalty-free redistribution of the Program by all those who receive copies directly or indirectly through you, then the only way you could satisfy both it and this License would be to refrain entirely from distribution of the Program.”

Because business motives of patent licensors differ, you will need to read the patent licenses, and the policies of the standards organizations, to make sure that the license you receive to a patent is compatible with your open-source software. Seek out standards organizations like W3C and the Embedded Linux

Consortium that invite input from the Open Source community on their patent policies and procedures.

Be aware that “industry standards” are not always what they seem. Some companies or standardizing organizations attempt to control standards through copyrights on specifications, or by requiring payment for the use of certification marks to demonstrate adherence to the standard. Such restrictive techniques are fundamentally incompatible with open source and free software.

Many of us in the Open Source community are working, often behind the scenes, to convince companies to avoid restrictive standards and to share control over such standards so as to make them truly available under free and open-source terms. The good news is that our input is increasingly being solicited, and that the resulting standards are often now compatible with free and open-source licensing.

Legal advice must be provided in the course of an attorney-client relationship specifically with reference to all the facts of a particular situation and the law of your jurisdiction. Even though an attorney wrote this article, the information in this article must not be relied upon as a substitute for obtaining specific legal advice from a licensed attorney.

email: lrosen@rosenlaw.com

Lawrence Rosen is an attorney in private practice in Redwood City, California (www.rosenlaw.com).

Archive Index Issue Table of Contents

Advanced search

Letters

Various

Issue #97, May 2002

Readers sound off.

Letters

Well-Tamed Demon

Your articles on configuring the pppd are outstanding [see Tony Mobily's “Configuring pppd in Linux” parts I and II in the February and March 2002 issues of *LJ*]. This is the way a HOWTO should be done—excellent hands-on information that works, with complete directions for us beginners. Well done! You can ask Tony back any time.

—Jack Dennon

Propagating Disease?

I read with interest Larry Rosen's “Unbiased License FUD” in your March 2002 issue. I am appreciative that Larry is dispelling the confusion about how the GNU GPL works. He is quite correct that to have obligations under the terms of the GNU GPL, one must actually create and/or distribute a derivate work of some copyrighted software licensed under the GNU GPL.

However, while dispelling FUD in that area, Larry (perhaps unintentionally) helped to spread similar FUD himself. In his article, Larry propagated the idea that the “GPL is an infection”. Infections are (according to “dict” on my Debian GNU/Linux system) things “which taint or corrupt morally” or “cause...disease”.

This sort of fear mongering about the GNU GPL became all too common during summer 2001, when various high-ranking executives of Microsoft called the GNU GPL an un-American cancerous virus that ate up software like a Pac-Man. I doubt that Larry or your magazine want to affiliate themselves with that concerted campaign of FUD.

As a strong advocate of software freedom, I often tell people: “The GNU GPL is not a virus; by contrast, it vaccinates you from harm.” The GNU GPL is designed to share freedom with all who benefit from the software. No one is required to “join the club”, but when you do choose to create derivative works of GNU GPLed software, you have obligations to the community. The GNU GPL is designed to create a software commons, with provisions that help avoid the tragedy of the commons.

I urge your editorial staff and Larry himself to avoid using biased terms like “infection” when trying to give “Unbiased License” information. The unbiased way to describe the GNU GPL's nature in this regard is simple: “The GNU GPL requires that those who distribute derivate works license the source of the derivate work in a GPL-compatible way.”

—Bradley M. KuhnVice President, Free Software Foundation

Larry replies: Bradley, thanks for your letter. I usually try to avoid the term “infection” when speaking of the GPL. However, I used that word in that particular article because, as I quoted directly in my second paragraph, the reader who responded to me used that word. I certainly did not intend to propagate the notion that the GPL has virus-like attributes, in the sense of things “which taint or corrupt morally” or “cause...disease”. I perhaps should have used the word “inheritance” instead, as I have done in my other writings about the GPL, because that word has more positive implications. Indeed, I have recently started to use the term “reciprocity” to describe the effect of the GPL and similar open-source licenses (including the MPL and CPL). Those licenses require licensees to reciprocate by licensing their derivative works under the same license as the work they were given. In contract law terms, I contend, the reciprocity obligation is the “consideration” you pay for the grant to you of the free license to the original code.

A Perl in the Spam

I have a few comments regarding David Bandel's Focus on Software column in the March 2002 issue of *Linux Journal*. First, great job, David. Keep up the good work! For me, your article is second only to Freshmeat when it comes to discovering new software. Second, I was thrilled to see coverage of Vipul's Razor (razor.sourceforge.net). I've found it to be incredibly effective in weeding out my daily dose of spam.

Although I am a long time Procmail user, I've often found its recipes difficult (at best) and cryptic (at worst). I can't begin to count the hours I've spent writing and debugging what should be a fairly simple filter. I think many of even the more hard-core Procmailers would be pressed to disagree with me on this point.

So, it was with no small amount of giddiness that I discovered Mail::SpamAssassin. This module (accessible via CPAN.org) is a plugin to the Mail::Audit module. The true advantage is that, because it's Perl, you can easily filter your e-mail through a Perl script. I have a tutorial covering this at PerlMonks (www.perlmonks.org) under the Tutorials section entitled (appropriately enough) “A Beginner's Guide to Using Mail::Audit and Mail::SpamAssassin”. Using these modules gives you the power of Perl (renowned for its ability to parse text) to filter your e-mail. I've been using this for several months now, and I've found that approximately 99% of my daily spam is filtered appropriately. And, I'm proud to report that none of my e-mail has been lost. Again, you're doing a great job. Keep it up!

—Stephen E. Hargrove

Sex Clarified

Seymour Cray didn't like virtual memory and has been quoted as saying: “Memory is like sex; it's better when it's real.” Whereas Paul Barry (*LJ* March 2002, Letters) apparently wasn't aware of Seymour's quip, Linus certainly was; his comment comparing software to sex is an obvious allusion. Perhaps it's not that Linus needs to choose his words more carefully; perhaps it's that authors should explain the literary significance of these words for the younger generation.

—Collin Park

Netfilter Rescue

After spending a few nights with *LJ* [see David Bandel's “Taming the Wild Netfilter” in the September 2001 issue, available at /article/4815] at hand's reach to configure my firewall, I came across the URL www.boingworld.com/workshops/linux/iptables-tutorial. This is to me the best source I have found on Netfilter. I believe the reference to Rusty Russel's document is a tribute to Rusty's own work, but the document is not usable for a basic user like me, a simple, average Linux user! Please to advertise this URL to your user if they have troubles using the wild, wild Netfilter.

—Thomas Smets

For more on Netfilter, see David Bandel's more advanced article “Netfilter 2: in the POM of Your Hands” in this issue.

—Editor

GT Explained

Gary Bickford's letter in the February 2002 issue of *LJ* states: “GTO is an acronym (Gran Turisimo Omologato), which according to AltaVista means 'great accredited tourism', which I suspect doesn't express the flavor of the phrase.”

He's right—Gran Turisimo refers to a racing class (Grand Touring, or GT) of cars, and Omologato (“homologated” in English) refers to the particular group of cars that was built specifically to fulfill the racing organization's requirements. As it turns out, Ferrari cheated on this; they never actually built the number of cars they were supposed to, but the organization never called them on it.

Most Pontiac dealers, when asked where the GTO moniker came from, were at a loss to explain any of this when the acronym was tacked onto the 389-engined version of the Pontiac Tempest.

—David Spellman

Speedier PHP

I found a small problem when using the caching PHP file [see Bruno Pedro's “Improving the Speed of Web Scripts” in the March 2002 issue of *LJ*]. The md5 hash is generated by using the PHP_SELF name of the called PHP script. But what if I generate content depending on the query string? I changed from PHP_SELF to REQUEST_URI, so it generates a unique cache file per unique parameters. This way I saved 90% execution time compared to running against my MySQL data sources.

—Kai Kretschmann

Erratum

In the March 2002 issue, on page 94, “'Using Mix-ins with Python', *Linux Journal*, April 2000” should read “'Using Mix-ins with Python', *Linux Journal*, April 2001”, i.e., the referenced article appeared in the April 2001 issue, not 2000.

—Chuck Esterbrook

UPFRONT

Various

Issue #97, May 2002

Stop the Presses, *LJ* Index and more.

Who says computer mags have to be dry? We've wetted this issue by hiding a bottle of beer somewhere within its pages. So put on your 3-D glasses (or don't—it will probably make it harder) and look for the bottle. If you find it, send the page number to info@linuxjournal.com by May 31st. The first 100 people to send correct answers will receive a “Powered by Linux” license plate frame and one of each of our popular Linux bumper stickers.

The Largest Open-Source Event in Latin America to Happen May 2-4. 2002

Richard Stallman is one of over 200 speakers set to talk at the largest open-source event in Latin America. Every year, thousands of members of the Open Source community meet at the Fórum Internacional do Software Livre, scheduled this year from May 2-4, 2002 in Brazil.

Rio Grande do Sul, the southernmost state in Brazil and host of the event, is a leader in the adoption of open source by the government and commercial users. The state has specific legislation regulating the use of software by government agencies, giving preference to open-source software.

More information on the Fórum can be found at the official site, www.softwarelivre.rs.gov.br/forum (Portuguese) or by e-mail at contato@softwarelivre.rs.gov.br.

—Marcio Saito

Linux Bytes Other Markets: Linux Helps Make Meals Special

Ypsilanti, Michigan, February 4, 2002—Linux, the open-source operating system, is at the heart of a popular new recipe service, recipesbyemail.com.

Developed by Tap Internet, this service allows anyone with an e-mail client to search hundreds of recipes quickly and easily.

“It was something originally done as a proof-of-concept for a client—a database searchable through e-mail”, said Michael Kimsal, director of Tap Internet.

> But soon after we'd finished the core technology, we decided to put up a “sample” to show other clients. Recipesbyemail.com has quickly become a staple for many users on the Internet to find just the recipes they're looking for.

A sizable percentage of the loyal users have turned out to be blind, due primarily to the fact that many web sites are so wrapped up in JavaScript rollovers and fancy flash animation that users relying on text-to-speech systems cannot use them at all. “It was quite interesting to get e-mails from some of these users because it was something I'd never even considered before”, said Kimsal.

“We intentionally keep the e-mails text-only to ensure maximum compatibility whether you're using Outlook or a Palm Pilot or a cell phone”, said Kimsal. “By building on Linux, we were able to do this on a shoestring and still provide value to the users, many of whom don't have very fast connections or modern browsers.” The system is built on Slackware Linux, using a combination of Perl, MySQL, procmail and PHP.

“We will be introducing a new feature soon to allow users to send in their own recipes, which we hope will help increase interest in the system even more.”

—Recipesbyemail.com

LJ Index—May 2002

1. Pentagon transactions, in trillions of dollars, for which there is no accounting: 2.3
2. Approximate amount, in millions of dollars, wasted by the Pentagon in the last two minutes: 2
3. Number of moons in the solar system with diameters larger than the planet Pluto: 8
4. Number of kilometers by which the diameter of Earth's moon exceeds that of Pluto: 1,176
5. Hours between the crawling of a barely exposed web page by an e-mail-harvesting robot and the arrival of its first spam: 8

6. Estimated ten-thousandths of a second that Earth's rotation will be slowed by global warming by 2100: 1
7. Percent by which the world's over-65 population will grow by 2025: 100
8. Percent by which the world's number of children will grow by 2025: 3
9. Number of wireless public “hot spots” scheduled for deployment in Korea before the Soccer World Cup this summer: 25,000
10. Percentage of users who arrive at web sites by direct navigation or bookmarks (rather than search engines), as of February 6, 2002: 52
11. Same as above, one year earlier: 46
12. Millions of dollars in reported Yahoo profits in 2000: 71
13. Billions of dollars in losses Yahoo would have had in 2000 if option expenses had been factored in: 1.3
14. Percentage of engineers whose career interests solidified before or during 10th grade: 59
15. Percentage of engineers who “decided to pursue their careers because of an affinity for math and science and their desire to innovate and explore new approaches to everyday actions”: 79
16. Billions of dollars in pocket change circulating in the United States: 7.7
17. Number of operating systems bundled with a $399 US PC at Walmart.com: 0
18. Number of Wal-Mart stores worldwide: 4,382
19. Low end of estimated range of Linux users: 2,403,060
20. High end of estimated range of Linux users: 60,076,500

Sources:

1-2: CBS News, January 29, 2002

3-4: Solarviews.com

5-6: DSL Reports (dslreports.com)

6: Astronomy.com

7-8: United States Census Bureau

9: Wireless World Forum

10-11: ZDNet UK, sourcing WebSideStory

12-13: *Fortune*

14-15: MathSoft survey of 1,200 engineers (mathcad.com)

16: *Business 2.0*

17-18: Wal-Mart

19-20: Linux Counter, February 21, 2001

Netcraft: Steady as She Goes

Netcraft's latest (January 2002) web server survey (www.netcraft.com/survey) still has Apache in the lead among active sites with a 63.69% share, up 0.35%. Microsoft IIS holds a 26.07% share, down 0.55%. Both saw increases in numbers of servers. iPlanet was third with 2.99%, and Zeus was fourth with 2.16%. Both were essentially unchanged.

Netcraft also reported “mixed fortunes” for Sun's Cobalt subsidiary, which sells Linux-based web servers. “Although numbers of IP addresses on Cobalt machines has increased over the last year, their share of the total number of sites running on Linux has fallen, almost relentlessly month on month.” Netcraft notes that two large customers have switched from Cobalt to conventional Linux machines. Texas ISP Everyone's Internet recently announced “the largest single purchase ever by an independent North American ISP from Cobalt”. The company bought seven hundred Cobalt RaQ servers.

—Doc Searls

It's Trivial

Questions

Q1. Whose web site is titled “The homepage of a WWW-illiterate”?

Q2. Torvalds commenting on a certain person's rant about open-source software: “I'd rather listen to Isaac Newton than to *X*. He may have been dead for almost 300 years, but despite that he stinks up the room less.”

Simple question: who is Torvalds talking about?

Q3. What's the collective noun for a group of penguins?

Q4. Vinod Valloppillil of Microsoft certainly said quite a few flattering things about Linux:

“Linux represents a best-of-breed UNIX that is trusted in mission-critical applications.”

“Linux has been deployed in mission-critical environments with an excellent pool of public testimonials.”

“I previously had IE4/NT4 on the same box, and by comparison the combination of Linux/Navigator ran at least 30-40% faster when rendering simple HTML + graphics.”

Where did Vinod Valloppillil make these flattering comments about Linux?

Q5. We all know Linus was studying at the Department of Computer Sciences, University of Helsinki when he started working on Linux. But what is Linus' mother tongue?

Q6. Ray Tomlinson, a scientist working at BBN, Cambridge achieved a unique distinction in 1971. What was it?

Q7. A word origin question: William Gibson, in his famous novel *Necromancer*, coined a word that has become very popular. What word?

Q8. What is Linus Torvalds' middle name?

Q9. Which application is *Linux Journal* talking about when it says “You know your program has caught on when people start to use its name as a verb”? Later in the same article *LJ* says, “It's no coincidence that the spread of this application has coincided with Linux distributions finally paring down the menu of potentially exploitable services offered by default.”

Q10. The author of this seminal work ends his acknowledgements with “and AT&T Bell Labs for firing me and making this all possible”. While talking about this book, *Wired* magazine says, “The book the National Security Agency never wanted to be published.” Too many clues already, but name the book and author.

questions and answers.

Answers

A1. Linus Torvalds—check it out at www.cs.helsinki.fi/~torvalds.

A2. Craig Mundie, Microsoft Senior Vice President.

A3. A waddle of penguins or a raft of penguins. A group of penguins in water is called a “raft of penguins”, while a group on ice is called a “waddle of penguins”. This was decided at the 4th International Penguin Conference in Chile in September 2000.

A4. In the (in)famous Halloween Documents. In case you haven't heard of the Halloween documents, go to www.opensource.org/halloween/halloween1.html.

A5. Though Linus grew up in Helsinki, the capital of Finland, his mother tongue is Swedish. Finland has a significant Swedish-speaking population and they call themselves finlandssvensk.

A6. He sent the world's first network e-mail. And according to him, the first e-mail most probably was something as innocuous as QUERTYIOP.

A7. Cyberspace.

A8. Benedict.

A9. Fyodor's Nmap. The article is the Editors' Choice Awards [December 2002 issue of *LJ*, /article/5525], and Nmap was judged the best security tool.

A10. *Applied Cryptography* by Bruce Schneier.

—Sumit Dhar

Stop the Presses: Linuxcare Founders Launch Sputnik Wireless Network

While Boingo (the new national wireless internet system headed by EarthLink founder Sky Dayton) has been getting a lot of attention, the three Linuxcare founders—Dave Sifry, Dave LaDuke and Art Tyde—have been quietly building a system of their own—one based on the sharing model pioneered by the Free Software, Open Source and Linux movements. The company is Sputnik, and after a cautious beginning, the service has finally been launched.

To put Sputnik in context, it helps to see a wireless network (802.11b, or WiFi) as one of three things: 1) a closed wireless Ethernet LAN, 2) a wireless way to get on the Net *for a fee* or 3) a wireless way to get on the Net *for free*. Any one of the three might show up in your wireless client software when your laptop is within range of a “hot spot”. But only the last two offer ways to get on the Net. The main difference between Boingo and Sputnik is that Boingo aggregates businesses offering number two, while Sputnik extends both numbers two and three—especially three. Think of it as a fee-for-use value add-on to all the participating hot spots in the world.

Here's the difference in frame of reference: Boingo comes from the PC world, and Sputnik comes from the Linux/UNIX world. Boingo offers users client software and service for a fee, while it offers service providers a uniform way to interact with those users, along with a straightforward revenue stream. Sputnik offers users a way not only to take advantage of WiFi bandwidth, but to serve it up as well. As with Linux, your laptop is both a client and sever. You become a fully empowered part of the system.

With Sputnik Gateway Software, you distribute bandwidth while you use it. Your laptop becomes an access point—a hot spot. But instead of performing as an indiscriminate hub, Sputnik's gateway acts as an intelligent router, sending traffic by priority to other Sputnik members. The system expands with each new member. Dave Sifry says, “The gateway is a smart edge device, there is no need to run a special client in order to use it. No software to download onto your box! Totally standards-driven!”

Here's how Glenn Fleishmann of 802.11b Networking News puts it:

> This is totally amazing.
>
> The Sputnik folks have solved all the problems Boingo didn't while offering an interesting, viral alternative. They solve the firewall issue (local networks are protected while the AP is open), authentication issue (captive portal without any work) and priority issue (local users and Sputnik affiliates have QoS above random folk).
>
> Because being a member opens the rest of the network to you for free, it plays on both aspects of the Internet in general and wireless community networks in particular: enlightened self-interest, as your adoption increases the size of the network and the likelihood of others to join; and generous selflessness, as you have nothing in particular to gain by allowing others to use your access point and network. Only in combination do these two virtues turn into a viral message.
>
> Of course, some may cavil at this: it co-opts community networks by offering a prefab, PC-based package that looks like a closed box. Money is only taken from outsiders, those not generous or sophisticated enough to become affiliates.

I'm curious if this will provoke a firestorm of criticism.

The gateway comes with a firewall. It also serves as an application platform, does caching, tracks usage, handles authentication, remote management and other things. Its core technologies are also open source and available under the GPL.

Subscribers pay a monthly fee, but for a limited time (as we go to press), subscribers can roam for free. Sputnik is at www.sputnik.com.

—Doc Searls

They Said It

SETI is a science, not a screensaver!

—SETI Institute (www.seti.org)

The present invention relates to the creation and use of synthetic forms of existence, or androids, and more specifically relates to the development of a universal epistemological machine in which any forms of the universe, conventional technologies included, are represented, embodied and realized as eternal moments of an infinitely expanding continuum of enabled existential forms, as an alternative approach to resolving the problems of the human condition.

—Patent No. 6,341,372: “Universal machine translator of arbitrary languages”, January 22, 2002

With open mail relays, lists of 30 million e-mail addresses and a cable modem, it only takes a handful of professional spammers...to deposit ten or more daily pieces of spam into the mailbox of practically every frequent user of the Net.

—DSL Reports

Linux keeps me up at night in terms of the energy that IBM is putting in it. The thing that's a corollary to that—but much more important—is the intellectual issues associated with that, GPL in particular. I worry a lot because it goes to the heart of whether you can have a business selling software. Whether (or not) we're innovative keeps me up at night....The complexity, the legal restrictions keep me up at night. As to why I'm still here, I don't know. I do love technology.

—Jim Allchin

I finally found a really interesting Linux-based alternative to Microsoft's tools and applications. But somehow I couldn't get excited about going after Microsoft's crown jewels.

—Stewart Alsop, venture capitalist and *Fortune* columnist

Business revolutions happen whenever demand acquires the power to supply.

—Doc Searls

If in the real world everybody is going to be famous for 15 minutes, on the Web everybody gets to be famous for 15 people.

—David Weinberger

There is nothing stable in the world; uproar's your only music.

—John Keats

Next Move: Sell It with Linux for the Same Price

Want to know what a headless, vanilla white box PC ought to cost? It's hard to imagine a better place to check than Wal-Mart.

At the Wal-Mart web site (www.walmart.com) you'll find a generic 1GHz Intel white box (the Microtel SYSMAR116) with the usual state-of-the-moment features and figures: 128MB RAM, 40GB drive, keyboard, mouse, modem, amplified speakers, floppy, etc. And in bold uppercase letters, this welcome disclaimer:

MONITOR - NOT INCLUDEDOPERATING SYSTEM - NOT INCLUDED

Commodities don't get much more commodified than that.

The price: $399

The site: www.walmart.com/catalog/product.gsp?product_id=1731327

From the Editor

Richard Vernon

Issue #97, May 2002

Welcome to *Linux Journal*'s kernel issue. Though it's true what Ted Ts'o says in this month's interview about the more exciting work in the Linux community happening in user space than in the kernel, there are still enough intriguing developments in the kernel to merit devoting an issue to it.

In fact, some of the most exciting recent kernel developments are covered in this month's pages. Last month we ran Rick Lehrbaum's interview with the preemptible kernel patch maintainer, Robert Love. This month Robert wrote a feature article explaining just how the patch lowers latency and how this translates to performance benefits, not only for those needing real-time efficiency, but also for regular users.

Greg Kroah-Hartman, the Linux USB and PCI Hot Plug kernel maintainer, reveals how the Linux kernel, as of 2.4.15, handles the kernel-level difficulties associated with hot-pluggable devices by way of the PCI Hot Plug driver core.

While iptables are no longer the latest in kernel development, the Netfilter code is constantly evolving, and many are still struggling with iptables building. Last year for our kernel issue David A. Bandel wrote an introductory-level article on Netfilter. He received a deluge of e-mail requesting further guidance. So to satisfy our readership, David delves into more advanced iptables building. Look for a further sequel to David's article in next month's Kernel Korner.

Continuing in a security vein, Michael Bacarella explains how POSIX capabilities in the Linux kernel can provide a useful middle-ground permission level that grants more liberal permissions than a standard user, but not the potentially harmful level of root.

In our last feature article, David Frascone brings us to the border of user land and the kernel by explaining the benefits of kernel module debugging with

User-Mode Linux. UML provides something of a virtual machine for safer debugging.

Speaking of user land, one of the more intriguing developments there, at least for our production staff at *Linux Journal*, may be the Scribus Project (web2.altmuehlnet.de/fschmid/index.html). Here at *Linux Journal* we try to practice what we preach, and everyone from the accountant to the receptionist, to the marketing and editorial departments do their work on Linux workstations. The only thing we don't do on Linux is magazine layout. Hopefully the Scribus Project will allow us to change that.

Scribus is a GPLed layout program for Linux. It's still in its early stages of development with a team of three—two of whom do the documentation, leaving all the programming to Franz Schmid. Franz is currently working on adding new object types like curves, polygons, etc. The team's goal is to match the quality of programs such as Adobe PageMaker and QuarkXPress.

I'm sure Franz wouldn't object if someone wanted to lend a hand. He can be reached at Franz.Schmid@altmuehlnet.de.

Richard Vernon is editor in chief of *Linux Journal*.

Advanced search

Best of Tech Support

Various

Issue #97, May 2002

Our experts answer your technical questions.

Best of Technical Support

Permissions Change on /etc

I am experiencing random permission 600 applied to /etc. Sometimes I can go a day without being struck, and sometimes only minutes later, after doing **chmod 755 /etc**, it's back to 600. I thought this might be some protection feature in the kernel to protect against bad memory or mass storage. I have changed out memory, leaving only one stick fully qualified by my motherboard maker, but to no avail. I can't play with the disks right now. I reduced the number of processes to those listed in the attached file. Is this a normal feature due to marginal hardware, a misinstalled application or is my system hacked?

—Stan Katz, stan@cloud9.net

Hardware issues will (almost) never cause filesystem changes in this manner and certainly not in such a consistent manner. It's time to look for applications (possibly malignant) that may be causing the problem. You need to examine your other log files—something might be a hint there. You only included messages through the system boot, nothing from its operation. Start by looking in all crontab files, tracing out processes that are run from it. Also compare the system dæmon binaries you are running, as well as other regularly used tools, against those on your installation media to verify that they have not been replaced with a Trojan horse. Finally, check for applications that are setuid root, since those are a sure tip-off that you have been compromised. However, it's unusual for a Trojan horse to restrict the permissions on a directory. Without access to the system, its log files and the ability to examine the files installed, we cannot further analyze the problem.

—Chad Robinson, crobinson@rfgonline.com

You almost certainly have a cron job that is resetting the permissions, or worse, a cracker kernel module that is messing around with your system. You should try **chmod 755 /etc; chattr -i /etc** to make /etc immutable, which hopefully will help make whatever program is resetting the permissions fail (for a cron job, you may even get an error by mail). Typing **rgrep -r chmod /etc /var/spool/cron** may also give you a clue as to what is changing the permissions.

—Marc Merlin, marc_bts@valinux.com

This is not a feature. Sounds like a Trojan or misinstalled program. This is a difficult problem to find. Try to do an **lsof /etc** to see if any program is currently holding the directory open. This may give you a clue. Next, shut down various services/programs until you find the offending program. It might be easier to re-install your version of Linux from scratch.

—Christopher Wingert, cwingert@qualcomm.com

I Do Believe in /dev/st0!

I can't access my tape! Running **dmesg** shows:

```
scsi0 : Adaptec AHA274x/284x/294x
        (EISA/VLB/PCI-Fast SCSI) 5.1.33/3.2.4
       <Adaptec AHA-294X Ultra SCSI host adapter>
scsi : 1 host.
  Vendor: ARCHIVE  Model: Python 28454-XXX  Rev: 4ASB
  Type:   Sequential-Access    ANSI SCSI revision: 02
  Vendor: FUJITSU   Model: M1606S-512       Rev: 6236
  Type:   Direct-Access        ANSI SCSI revision: 02
Detected scsi disk sda at scsi0, channel 0, id 3, lun 0
scsi : detected 2 SCSI generics 1 SCSI disk total.
(scsi0:0:3:0) Synchronous at 10.0 Mbyte/sec, offset 15.
SCSI device sda: hdwr sector= 512 bytes.
Sectors= 2131992 [1041 MB] [1.0GB]
```

I can access the SCSI disk but not the tape. Both eth0 and aic7xxx are on interrupt 9. I have SCSI tape support compiled in the kernel. This is all on a P133.

—Andy Prowse, azp80@amdahl.com

Normally, dmesg(8) should show a line referencing st0 as a device found in the summary, just below the entry about sda. See the example below from my own system:

```
Detected scsi tape st0 at scsi0, channel 0, id 6, lun 0
```

Your tape drive is being discovered as a “generic” device, but you cannot use mt(1) without the tape driver active. Check your kernel compilation options.

Depending on your kernel version, this should be under “SCSI support” and named “SCSI tape support”.

—Chad Robinson, crobinson@rfgonline.com

Routing to ppp0

I installed Red Hat 7.2 with a network card and successfully connected to the Internet with a cable modem connection hosted on another machine in the network. Now I need to connect to a dialup account. The modem is correctly configured (I can log in successfully). But when I try to work with the dialup account, I am unable to connect to its mail server because my Linux box is using eth0 routes as the default routes. How do I change the routing tables to fix this? I already checked the box to make ppp0 the default connection, but that did not solve the problem.

—Kelvin Barnes, Kelvin.Barnes@att.net

You have a default route going to your Ethernet interface that is superseding the default route added by PPP. You can try **route del -net default** before bringing your PPP interface up, and then it should work. You also can use the Red Hat GUI to bring eth0 down and back up.

—Marc Merlin, marc_bts@valinux.com

Die Process, Die, Die

Recently, several attempts to run **vi /etc/*filename*** resulted in vi freezing and Telnet/SSH not responding to break commands (Ctrl-C, Ctrl-B, Ctrl-D). I logged in to the server again and tried to kill the process. The man pages said that if **kill** didn't work, it was probably a result of the kill command being a part of the shell. So I tried **/bin/kill <*pid*>**, **/usr/bin/skill <*pid*>**, **/usr/bin/killall vi** with 9, 15 and several other signals. Running **top** and killing the process via top didn't work either. It has been a couple of days and about a dozen vi processes are still running. I need a license to kill!

—Peter D'Souza, souza@broadleaf.net

If a process is stuck in kernel state due to a kernel or network problem, you will not be able to kill it, even with **kill -9**. In that state, you can usually only reboot to get rid of the process.

—Marc Merlin, marc_bts@valinux.com

I Have No Valid Modes and I Must startx

Why can I not get startx to work on a Toshiba Satellite Pro 4600 that uses a Trident CyberBladeXP video card? I am running Red Hat 7.2., and I get the following error:

```
Fatal server error:
No Valid modes found.
```

—Troy, coder@starmail.com

There is one update from Red Hat's site that mentions solving a problem with your video card. Please try to upgrade the necessary packages and try again. You can find more information about it at rpmfind.net/linux/RPM/redhat/updates/7.2/i386/Xconfigurator-4.9.39-2.i386.html.

—Mario Bittencourt Neto, mneto@buriti.com.br

DragonLinux almost Boots

When I run DragonLinux everything loads perfectly until **Space Freed:**. Then it says:

```
Warning:Unable to open an initial console
Kernel Panic:no init found.
Try passing init= option to kernel.
```

—Alok Bhatt, dkbhatt@eth.net

You may have corrupted the root filesystem after you finished installing. I suggest that you reinstall Linux and make sure that you make a filesystem on all Linux partitions. If you experience the same problem again, then you need to contact the DragonLinux people.

—Usman Ansari, usmansansari@yahoo.com

Teaching Red Hat Japanese

I thought I'd take the opportunity to e-mail you about configuring Red Hat 7.0 to be able to use Japanese. I have a 109-key Japanese keyboard and am using a Japanese 106 keymap that works okay, despite being unable to use the Japanese-English switch key. I enabled deadkeys and installed all the Japanese language packets during installation. These packets, WNN and Kanna packets, among others, start up on boot but don't activate for some reason. Any help would be appreciated. I think I could solve it by getting the right keyboard map. Japanese characters will appear on most software in KDE and GNOME, but getting output from the keyboard is the problem.

—Graeme Jensen, magic@zae.att.ne.jp

I recommend you use a distribution that has been widely tested with Japanese. You'll have better odds with those things working out of the box. I have to admit I'm not sure which is the Japanese distribution of choice today, but you may want to give Turbolinux a try.

—Marc Merlin, marc_bts@valinux.com

Archive Index Issue Table of Contents

Advanced search

New Products

Heather Mead

Issue #97, May 2002

VT100 Set-Top Box, Port Server CM, CrossOver Plugin v1.1 and more.

New Products

VT100 Set-Top Box

VT Media Technologies has added the VT100, the Edge, to its line of set-top boxes. The Edge enables several data streams, including Ethernet 10/100 to be converted to composite analog RF data streams or digital S-Video data streams compatible with all standard television sets. A full browser and support for all plugins are provided, and the Edge also is compatible with CRT monitors. Options available for the Edge include a DVD player, CD-RW, floppy and standard IDE hardware. It also supports the Web-Media software announced by National and Century Embedded Software. Developer kits for the Edge are also available on the VT web site.

Contact VT Media Technologies, 6307 County Road 87 SW, Alexandria, Minnesota 56308, 320-763-8491, sales@vtmt.com, www.vtmt.com.

PortServer CM

PortServer CM is a new line from Digi International that offers products for data center management. Available in a 1U, 32-port design, the rackmountable PortServer CM allows administrators to monitor and control any mix of connected devices from anywhere on the corporate network, including standard TCP/IP connections over Ethernet LANs or dial-up modem connections. SSH v2 is used for security, and other supported protocols include DHCP, PPP, SLIP, NTP and FTP. Memory specs are 64MB SDRAM and 4MB Flash (upgradeable).

Contact Digi International, Inc., 11001 Bren Road East, Minnetonka, Minnesota 55343, 1-800-344-427 (toll-free), www.digi.com.

CrossOver Plugin v1.1

Version 1.1 of the CrossOver Plugin from CodeWeavers, Inc., is designed as a Windows-to-Linux adapter for Windows browser plugins and e-mail clients. In addition to opening MS Office documents and eFax files in any KDE or GNOME application, version 1.1 supports Windows Media Player streams. Other improvements in CrossOver 1.1 include acceptance of all TrueType fonts, RealPlayer files, the Trillian plugin, Yahoo Messenger, the iPIX plugin and Chime, a plugin for the chemical industry. CrossOver software supports most browsers, including Netscape, Mozilla, Konqueror, Opera, SkipStone and Galeon.

Contact CodeWeavers, Inc., 2550 University Avenue West, Suite 493S, St. Paul, Minnesota 55114, 651-523-9300, sales@codeweavers.com, www.codeweavers.com.

Desktop/LX

Standard and Deluxe box sets of Desktop/LX, a Linux distribution from Lycoris (formerly Redmond Linux Personal), are now available. The Standard box set includes the Desktop/LX CD-ROM, a 30-page installation manual and 60 days of e-mail support. The Deluxe box set also includes a source code CD-ROM and a DevTools CD-ROM, allowing Desktop/LX Deluxe to be used as a development platform. A graphical installation autodetects supported video, audio and network hardware, as well as attached printers. Desktop/LX offers an easy and fast system setup and a wizard that allows internet-connected users to get the latest versions and upgrades.

Contact Lycoris, PO Box 2313, Redmond, Washington 98073, 425-869-2930, sales@lycoris.com, www.lycoris.com.

PS-R1242, Dual Xeon 1U Server

Based on Intel's E7500 chipset, the PS-R1242 1U server supports dual Pentium Xeon processors up to 2.2GHz, providing supercomputing power aimed at the media/entertainment and scientific industries. The PS-R1242 has optional SCSI Ultra160 or ATA 100 RAID controllers and can be configured with either two IDE or three SCA hot-swap drive bays. It also supports up to 8GB of two-way interleaved DDR SDRAM. Additional features include a 1GB Ethernet port, one Fast Ethernet port incorporated on the motherboard with two available PCI 133/100 slots and a 350/400W cold-swap power supply.

Contact Promicro Systems, Inc., 12635 Danielson Court, #203, Poway, California 92064, 858-391-1515, sales@promicrosystems.com, www.promicrosystems.com.

NASAS-2040

Ateonix Networks introduced the NASAS-2040, a network-attached storage appliance with a feature set designed for small to medium-sized businesses. It supports any combination of up to four ATA/100/133 disk drives from any manufacturer. The storage capacity of each disk can be up to 128 petabytes, with disks of varying capacity. Upgrades can be done on-site by in-house personnel, and hard disk can be added or replaced via front-panel access. All software updates, including the network OS, drivers and user configuration data, can be updated via a web interface. All of the NASAS-2040 software resides on Flash memory, and it supports RAID 0, 1 and 5.

Contact Ateonix Networks, Inc., 42618 Christy Street, Fremont, California 94538, 510-656-8400, sales@ateonix.com, www.ateonix.com.

SnapGear PRO+

With an integrated V.90 modem for automatic failover for ADSL and cable-connected customers, the SnapGear PRO+ VPN router appliance offers the ability to switch to a conventional modem connection in the event of an outage. PRO+ has a 3DES capacity of 35Mbps and specific optimizations for the PPPoE protocol used by ADSL. It also can support network throughput up to 18Mbps. Two 10/100 Ethernet interfaces are standard for WAN and LAN traffic, in addition to the modem. A built-in PPTP client and server allows Windows users to connect to the PRO+ without a third-party client. Other features include RADIUS/TACACS+ authentication and dynamic DNS support.

Contact SnapGear, Inc., 7984 South Welby Park Drive, #101, Salt Lake City, Utah 84088, 801-282-8492, contact@snapgear.com, www.snapgear.com.