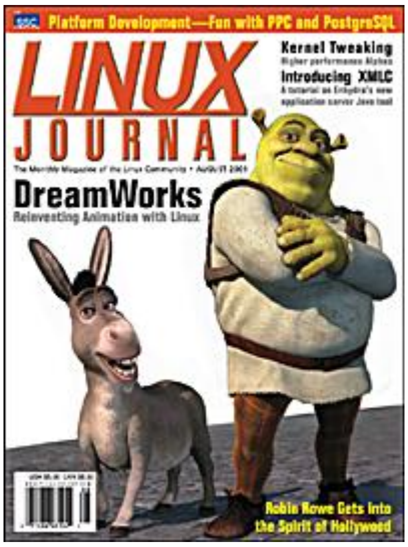
LINUX JOURNAL

Advanced search

Linux Journal Issue #88/August 2001


Platform Development—Fun with PPC and PostgreSQL
LINUX JOURNAL
Kernel Tweaking
Introducing XMLC
DreamWorks
Reinventing Animation with Linux
Robin Rowe Gets Into the Spirit of Hollywood

Archive Index

Advanced search

Focus: Platforms

Richard Vernon

Issue #88, August 2001

With this month's feature articles we cover both hardware and software.

The topic “platforms” is almost as broad as “computers” because anything upon which something else is dependent can be considered a platform. With this month's feature articles we cover both hardware and software.

Frequently, when people think platform they think processor architecture. And for years, as far as Linux was concerned, that meant x86, despite the fact that some courageous individuals began very early the work of porting Linux to other platforms. Some of us can remember as far back as 1996 in the precivilized days before 20GB hard drives when Linux was supported on only few platforms other than x86, such as the IA32, the Amiga and Atari. Now every major processor (and a whole lot of minor ones) have been ported to Linux.

In his article, “The Trials and Tribulations of LinuxPPC 2000 Q4”, Paul Barry discusses his experiences with the highly touted PPC processor, the one believed by many to have the best chance of taking the “tel” out of “Wintel” (see page 60). While many distros continue to support only Intel, a growing number are offering support for the PPC. Besides the usuals—Yellow Dog, MkLinux and LinuxPPC—SuSE, Mandrake and Debian also have distros for the PPC. In our August 2000 issue we ran an article on installing LinuxPPC, and Barry's article, almost a year later, provides a good measure for how far it's come and the distance still to go.

In our second feature article, “PostgreSQL Performance Tuning”, Bruce Momjian discusses what can be done on the hardware end to improve the performance of tasks involving the PostgreSQL database (see page 66). Momjian provides an illustration of memory types and uses and how to make the most of PostgreSQL by modifying cache size and sort size and spreading disk access across drives.

Also, see Stephanie Black's book review (page 76) on Momjian's *PostgreSQL: Introduction and Concepts*. Momjian's book includes tips on maximizing performance through optimizing the queries sent to the database. Between the article and the book, you should be able to get your PostgreSQL running at its maximum potential.

—Richard Vernon, Editor in Chief

The Trials and Tribulations of Installing LinuxPPC 2000 Q4

Paul Barry

Issue #88, August 2001

Read the manual before turning your Mac over to Linux.

I recently wrote about using Mac OS X as my main desktop OS (see “Mac OS X: First Impressions” at http://www.linuxjournal.com/articles/misc/0035.html). Prior to working with Mac OS X (the X is read as ten), I ran a Mac OS 9/LinuxPPC dual-boot system. A longtime Mac user but a UNIX geek at heart, I was thrilled with the prospect of Mac OS X—a Mac-like GUI on top of a UNIX core. I enthusiastically installed the Public Beta of Mac OS X on my desktop machine, a Macintosh Server G3.

Initially, all went well with Mac OS X. However, once the honeymoon was over, I became increasingly frustrated with Apple's latest OS. The main problems are it is terribly slow, it has no support for floppy diskettes, there are GNU software compatibility issues and it has poor “classic” Mac OS emulation. In particular, there is no support for AppleTalk file sharing with older Macintosh computers. The lack of backward-compatible AppleTalk support was the hardest to take. After all, this was Apple's new OS not talking properly to its old OS using Apple's own networking protocol.

Late last year I decided to dump Mac OS X in favor of LinuxPPC. General availability of LinuxPPC 2000 Q4 (LPPC2000) was announced in time for the January 2001 MacWorld Expo, and pre-orders were being taken on the LinuxPPC web site. I wasted no time in ordering a copy.

LinuxPPC Arrives

In early February a LinuxPPC box-set arrived in the mail. For $30 US (which included shipping), I received four CDs, a printed user's guide and my very own LinuxPPC “Think Linux” t-shirt. The four CDs were *Install*, *Source*, FWB

Software's *Hard-Disk Toolkit-PE* and *Extra Software*. The CDs were all dated December 2000.

I wanted to install on three computers: my G3 (128MB RAM, 4GB HDD), an original translucent green iMac (32MB RAM, 4GB HDD) and a cherry iMac (32MB RAM, 6GB HDD). Of the three, only the cherry iMac would require the services of the FWB-Toolkit (third-party software that allows an existing Mac hard disk to be repartitioned while preserving the original contents). This wasn't a concern with the other machines—the G3 was running Mac OS X, so repartitioning and reformatting were the orders of the day, and the green iMac was running an older version of LinuxPPC, the 1999 Q3 release. This machine's hard disk would also be wiped prior to installing LPPC2000. The cherry iMac is at home and is used by my kids to run a collection of Mac-based educational software.

The Green iMac

I started with the green iMac. The LinuxPPC web site and the user's guide made a big deal of the fact that this release was shipped on a bootable CD-ROM. It was simply a matter of dropping the *Install* CD into the CD drive, rebooting and holding down the C key while the iMac started up. What could be easier? I did this on the green iMac, and after a few moments Linux booted, and the LPPC2000 graphical installer (X Linux Installer) loaded and appeared on screen. So far so good. All I had to do was follow the instructions on screen.

Up popped four large buttons and two small ones. The large buttons were labeled Instructions, Language, Select Partitions and Choose Packages. The smaller buttons were Options Menu and Reboot. The Instructions button gives a brief introduction to the install process, and the Language button lets you choose a language to work in. The real work starts with the Select Partitions button. Clicking here brings up the Mount Partitions and Setup Swap window, providing a means of specifying the mount points and names of any partitions. The partitions can then be formatted and mounted from this window. Of course, prior to doing this I needed to get my partitions in order, and LPPC2000 ships with partitioning software called Perldisk. Now Perldisk isn't pretty, but it does work, even though it can be a little strange to use. I didn't like the Add Partitions dial control for specifying the size of a newly created partition. It was, for example, hard to move the dial to *exactly* 20MBs, so I ended up manually calculating the exact size of the partition and entering the value into the text field under the dial. It is important to calculate the size using the proper formula. For instance, to request a 20MB partition, use

```
   (1024 * 2) * 20
```

to get the correct value.

The partition information is then written to disk; then the Installer requires a reboot (remember to hold down the C key while the Mac starts up to continue with the install). With the Installer back on screen, I selected the Select Partitions button again and specified the partitions to format and mount prior to starting the install. For the green iMac, my partitioning strategy was to have /boot (20MB), swap space (128MB), /root (1GB) and /home (1.8GB). A further 1GB was set aside as a Mac OS formatted drive. I clicked on Done, and the partitions window disappeared.

The list of package groups then appeared in the Choose Packages window. Over and above the default selections, I selected the Editors, Interpreters, Network Servers, Network Apps/Utilities and Development package groups. A word of warning: don't unselect any of the default selections from this list—things tend to come out muddled as a result. I clicked on the Install button, and the process of copying LinuxPPC and its packages was underway.

It took a long, long time to copy over the 600MBs worth of packages, and I attribute this to the awful CD drive shipped with the original crop of iMacs. Some four hours later I was presented with a message box detailing the name of my root partition (a nice touch) and offering two further buttons: Reboot and Additional Options.

Clicking on Additional Options brings up the Configure LinuxPPC window that lists a selection of Post Install Options. There are buttons labeled Set Password, Set Date & Time, Set Timezone, Configure inetd, Set Network Settings, Configure Modem Port, Configure PPP Settings, Configure Runlevels, Setup Users and Groups, Configure X Modifier Keys and Run linuxconf. I made sure the clock settings were correct, set my root password, set the timezone to “Eire”, removed any unnecessary network processes from **inetd** and then configured the iMac for use with IP. (Note that I prefer to use **chkconfig** over the Runlevel Editor.) I closed the Configure LinuxPPC window and then clicked on Reboot.

The green iMac restarted, then the dreaded Apple flashing question mark appeared in the middle of the screen. The green iMac couldn't find an OS to boot!

Direct Booting the Green iMac

The program that controls booting into LPPC2000 is called **yaboot**. The users' guide does a good job of describing how to use this program, but check the LinuxPPC web site for a fix to a small but important typo on page 40.

From the users' guide, I followed the instructions for selecting LPPC2000 as the green iMac's default OS. This involved booting from a Mac OS CD and then running the Startup Disk control panel. I selected /boot as my default choice and then restarted. The iMac continued to refuse to boot. I checked the documentation for clues as to why this was happening and was told that /boot should contain a “fake” Mac OS system folder copied there during installation. But this had not happened. On the green iMac, /boot contained the yaboot binary and a collection of kernel files but no system folder. I thought I was stumped until I remembered that the *Install* CD was itself bootable. I found a Mac system folder on the CD, copied it from the *Install* CD onto /boot, edited yaboot.conf, restarted the green iMac and watched while the iMac booted directly into Linux. Cool.

The G3 Install

Despite having gone through a successful, if somewhat involved, installation on the green iMac, installing on the G3 was to present even more hurdles. For starters, the holding down the C key while booting trick did not work. Only Macs with a “New World ROM” can boot directly from the *Install* CD. If your Mac is an iMac or newer, you are probably going to be fine. If it isn't, you won't be. The problem for me was that the G3 can boot from Mac OS bootable CDs, so I was initially confused as to why the *Install* CD did not boot. Of course, if I'd read the entire printed manual I would have seen that the G3 had problems in this area.

It certainly pays to read through the entire printed manual prior to attempting to do anything. Although the install process is made easier by the graphical installer, it is not straightforward, and this can result in a number of false starts. This is what happened to me with the G3.

As the G3 could not boot from the CD, the manual suggests booting into Mac OS and running the Mac-based Installer application from the *Install* CD. I had already booted from a bootable Mac OS CD and partitioned the G3's hard disk into two Mac OS partitions (750MB and 250MB), /boot (20MB), swap space (128MB) and /root (3.1GB). I installed a minimal subset of Mac OS 9.0 onto the Mac OS system partition, as my intention was to keep the OS available should I wish to access any legacy data from when I was using Mac OS exclusively. With Mac OS 9 installed and running, I popped the *Install* CD into the CD drive, double-clicked on it and looked for the Installer icon. To my dismay, the Installer application wasn't on the CD. I checked the manual and was told that it should have been there. It wasn't. Heading rapidly toward panic, I searched the other three CDs in the set, all to no avail. The Installer application was missing from the distribution CDs! Of course, the Installer application is available for download from the LinuxPPC web site, but as I'd only installed the minimal Mac OS setup, the internet connectivity software wasn't available to me.

The printed manual came to my rescue. Page 45 details the steps to perform if the Installer program doesn't work. I copied a selection of files and folders from the *Install* CD to various locations on the hard-disk and then restarted the G3. Up popped the Mac OS-based BootX OS chooser. I was in business.

I configured BootX to start the graphical installer and then clicked on the Boot button. Things went pretty smoothly from here on in. After about 90 minutes, I had a working installation of LPPC2000 running on my G3.

Booting on the G3

I decided to opt for a direct boot on the G3 as opposed to configuring BootX to provide a choice between Mac OS 9 and LPPC2000. I would never have considered this 18 months ago. But now, with more and more of my daily activities handled by Linux, I was ready to let go of the Mac OS crutch and run Linux as my main desktop OS. For those times when I needed to access Mac OS 9, I planned to run the Mac-On-Linux emulator within an X window.

As with the green iMac, I had to copy the system folder from the *Install* CD onto /boot and edit the yaboot.conf file. Upon rebooting, yaboot brought me directly to a graphical Linux login. LPPC2000 defaults to GNOME as the preferred desktop with KDE installed as an option. I left GNOME as the default.

The LinuxPPC Perl Shocker

I logged in as root and created an account for myself. Under user “barryp”, I ran a small set of commands to see what was installed. Typing **uname -a** confirmed that LPPC2000 was shipped with version 2.2.18 of the kernel. I checked to see if **vi**, **latex** and **ispell** were installed. As expected, they were. I then issued the **perl -v** command to see which version of Perl was running. What I saw shocked me. LPPC2000 ships with release 5.005_03 of Perl, not the 5.6.0 release which has been available (and stable) since mid-2000! It turns out that LPPC2000 is based on the software packages shipped with Release 6.1 of Red Hat. And of course, Red Hat 6.1 shipped with release 5.005_03 of Perl. Since 6.1, the folks at Red Hat have gone through 6.2, 7.0 and are now shipping release 7.1 of their distribution (as of April 2001). Even though LPPC2000 was new, it was already quite outdated. As a Mac user, this is something that I had become accustomed to: seeing new versions of software appear first on the Intel platform and then waiting for updates to appear on the Mac (if ever). I thought that moving to Linux on the Mac hardware platform would protect me from this. I was annoyed and disappointed.

Now as any Linux user knows, the latest release of open-source software is never more than a download and a “tar -zxvf” command away. However, for the vast majority of Mac users (who must constitute a large chunk of LinuxPPC's target market), this is already too much to ask of them.

The Challenge of Printing

My next task was to install and configure a printer. This is easily accomplished using the Printer Configuration control panel from within X. I'd done this before, and it has always worked fine. But not this time. Everything looked as it should, but printing did not work. I checked the LinuxPPC web site for any help and found a support item stating that LPRng needed to be installed. I downloaded LPRng and tried to install it. It complained that the rhs-printfilters package was missing. I found an RPM for this package on the Net, and after installing it I was able to complete the install of LPRng. Unfortunately, printing still did not work. I searched my hard disk for copies of the printcap file. It turned out I had two: one in /etc and another in /usr/etc. Checking the contents of these files confirmed that LPRng was using the empty printcap file in /usr/etc, while the Printer Configuration Control Panel was using the file in /etc. As root, I copied the printcap from /etc over the file in /usr/etc, restarted LPRng and, lo and behold, printing worked! Of course, it would have been nicer to have printing work out-of-the-box.

Other Included Software

The *Extra Software* CD has a load of LinuxPPC-ready RPMs of many different software packages. I was initially very interested in the Bochs x86 emulator and thought I might install the RPM to see what the emulator was capable of. Using the graphical RPM installer, I located the RPM on the CD. When I clicked to install, I got a message telling me that user/group jcarr was missing from my system and that I couldn't install the RPM as a result (Jeff Carr is a key player at LinuxPPC). I added an entry for jcarr to the /etc/passwd and /etc/group files, and the RPM installation problem went away. Really though, the techies at LinuxPPC should be catching things like this long before CDs make their way into the hands of users.

I was interested in the x86 emulator, but not as much as I was in the Mac OS emulator, Mac-On-Linux (MOL). Having gone to the trouble of installing Bochs, I have yet to look at it in any detail. The same cannot be said for Mac-On-Linux. Here is a piece of software that lets you run a copy of Mac OS (pre-X) within an X window. MOL does a better job of this than the similar technology built into Mac OS X. It supports more versions of classic Mac OS, including those based on releases 8 and 9. Mac OS X's classic emulation only works with the 9.1 release. MOL also has great support for AppleTalk—I was able to work with drives on remote Macintoshes within MOL, something that is not possible with Mac OS X. Printing was better integrated, too. All in all, MOL is another great advertisement for why open source is a better way.

There's Just No Shrinking a Cherry iMac

I was worried about installing onto the cherry iMac, primarily due to the fact that the kids would never forgive me if I upset their Mac OS installation.

It was time to take a look at FWB Software's Hard Disk Toolkit-PE. The cherry iMac used to run LinuxPPC 1999 Q3, but I had never really used it. So the 6GB was already partitioned as 4GB for Mac OS and 2GB for LinuxPPC. As one of the Toolkit-PE's main features is to rearrange existing partitions, I wanted to shrink the 4GB Mac OS partition to 3GB and use the extra 1GB as the cherry iMac's / root . The other 2GB would be used by /home. As no instructions on the use of FWB's tools came with LPPC2000, I downloaded an article from http://www.linuxppc.org/. I followed the instructions but could not shrink the partition. The software complained that the driver needed to be updated and suggested I choose the Update Driver option from the FWB menu. Every time I tried, Toolkit-PE crashed with a Mac OS error “of type 2”. When I tried to install the software onto the Mac OS partition, I was asked for a serial number. I did not get a serial number with LPPC2000, and no mention of the serial number was to be found on the http://www.linuxppc.com/ web site. The FWB article did mention that the serial number wasn't needed if the software was used directly from FWB's bootable CD. This was how I was using it, and it was crashing every time.

I had to make do with installing on the old LinuxPPC partition, and the install went smoothly. I'm not sure what I would have done had I needed to shrink the partition before the install, as reformatting, repartitioning and re-installing the Mac OS setup was not something I wanted to do. Luckily for me, the partitions already existed.

Concluding Remarks

This release of LinuxPPC is a huge improvement over previous releases. However, I am disappointed by the amount of effort I had to expend in order to get the software installed. Most of what I went through will be so alien to the vast majority of Mac folk that they may not bother with LinuxPPC at all, which would be a shame. Apple definitely has the upper hand here, as all it takes to install Mac OS X is a CD insertion and a reboot. Of course, the provided functionality leaves a little bit to be desired.

In defense of LinuxPPC, they do state that this release of their distribution is the last before a “major reorganization”. As a company, LinuxPPC is transitioning from a for-profit business to a nonprofit organization, and this is to be applauded and supported. Despite my initial disappointments and configuration problems, I now have LPPC2000 running smoothly on my G3. The gain was well worth the pain. I see no reason to change to another OS, and I keep a regular eye on the Updates page on the LinuxPPC web site. I also look forward to the next release of LinuxPPC. But be warned: read the manual before attempting your install and check the LinuxPPC web site for hints, tips, fixes and updates.

Paul Barry (paul.barry@itcarlow.ie) lectures in Computer Networking at The Institute of Technology, Carlow in Ireland (http://www.itcarlow.ie/). He is the author of Programming the Network with Perl, which will be published by John Wiley & Sons. The green iMac referred to in this article hosts Paul's home page at glasnost.itcarlow.ie/~barryp/index.html.

PostgreSQL Performance Tuning

Bruce Momjian

Issue #88, August 2001

Tweak your hardware to get the most from this open-source database.

PostgreSQL is an object-relational database developed on the Internet by a group of developers spread across the globe. It is an open-source alternative to commercial databases like Oracle and Informix.

PostgreSQL was originally developed at the University of California, Berkeley. In 1996, a group began development of the database on the Internet. They used e-mail to share ideas and file servers to share code. PostgreSQL is now comparable to proprietary databases in terms of features, performance and reliability. It has transactions, views, stored procedures and referential integrity constraints. It supports a large number of programming interfaces, including ODBC, Java (JDBC), Tcl/Tk, PHP, Perl and Python. PostgreSQL continues to improve at a tremendous pace thanks to a talented pool of internet developers.

Performance Concepts

There are two aspects of database-performance tuning. One is improving the database's use of the CPU, memory and disk drives in the computer. The second is optimizing the queries sent to the database. This article talks about the hardware aspects of performance tuning. The optimization of queries is done using SQL commands like CREATE INDEX, VACUUM, VACUUM ANALYZE, CLUSTER and EXPLAIN. These are discussed in my book, *PostgreSQL: Introduction and Concepts* at www.postgresql.org/docs/awbook.html [see also Stephanie Black's review on page 76].

To understand hardware performance issues, it is important to understand what is happening inside the computer. For simplicity, a computer can be thought of as a central processing unit (CPU) surrounded by storage. On the same chip with the CPU are several CPU registers, which store intermediate results and various pointers and counters. Surrounding this is the CPU cache, which holds the most recently accessed information. Beyond the CPU cache is a large amount of random-access main memory (RAM), which holds executing programs and data. Beyond this main memory are disk drives, which store even larger amounts of information. Disk drives are the only permanent storage area, so anything to be kept when the computer is turned off must be placed there (see Table 1). Figure 1 shows the storage areas surrounding the CPU.

Table 1. Types of Computer Storage

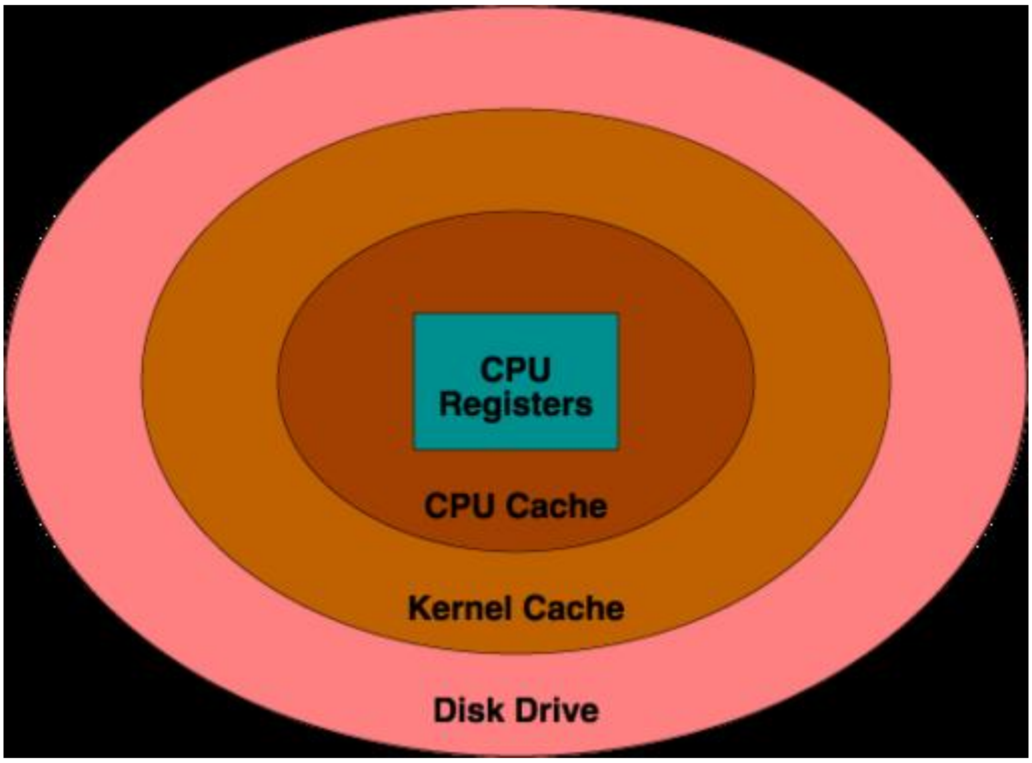


Figure 1. Storage Areas

You can see that storage areas increase in size as they get farther from the CPU. Ideally, a huge amount of permanent memory could be placed right next to the CPU, but this would be too slow and expensive. In practice, the most frequently used information is stored next to the CPU, and less frequently accessed information is stored farther away and brought to the CPU as needed.

Keeping Information Near the CPU

Moving information between various storage areas happens automatically. Compilers determine which information should be stored in registers. CPU chip logic keeps recently used information in the CPU cache. The operating system controls which information is stored in RAM and shuttles it back and forth from the disk drive.

CPU registers and the CPU cache cannot be tuned effectively by the database administrator. Effective database tuning involves increasing the amount of useful information in RAM, thus preventing disk access where possible.

You might think this is easy to do, but it is not. A computer's RAM contains many things, including executing programs, program data and stack, PostgreSQL shared buffer cache and kernel disk buffer cache. Proper tuning involves keeping as much database information in RAM as possible while not adversely affecting other areas of the operating system.

PostgreSQL Shared Buffer Cache

PostgreSQL does not directly change information on disk. Instead, it requests data be read into the PostgreSQL shared buffer cache. PostgreSQL backends then read/write blocks, and finally flush them back to disk. Backends that need to access tables first look for needed blocks in this cache. If they are already there, they can continue processing right away. If not, an operating system request is made to load the blocks. The blocks are loaded either from the kernel disk buffer cache or from disk. These can be expensive operations.

The default PostgreSQL configuration allocates 64 shared buffers. Each buffer is eight kilobytes. Increasing the number of buffers makes it more likely that backends will find the information they need in the cache, thus avoiding an expensive operating system request.

How Big Is Too Big?

You may think, “I will just give all my RAM to the PostgreSQL shared buffer cache.” However, if you do that, there will be no room for the kernel, or for any programs, to run. The proper size for the PostgreSQL shared buffer cache is the largest useful size that does not adversely affect other activity.

To understand adverse activity, you need to understand how UNIX operating systems manage memory. If there is enough memory to hold all programs and data, little memory management is required. However, if everything doesn't fit in RAM, the kernel starts forcing memory pages to a disk area called swap. It moves pages that have not been used recently. This operation is called a swap pageout. Pageouts are not a problem because they happen during periods of inactivity. What is bad is when these pages have to be brought back in from swap, meaning an old page that was moved out to swap has to be moved back into RAM. This is called a swap pagein. This is bad because while the page is moved from swap, the program is suspended until the pagein is complete.

Pagein activity is shown by system analysis tools like **vmstat** and **sar** and indicates there is not enough memory available to function efficiently. Do not confuse swap pageins with ordinary pageins, which may include pages read from the filesystem as part of normal system operation. If you can't find swap pageins, many pageouts is a good indicator you are also doing swap pageins.

Effects of Cache Size

You may wonder why cache size is so important. First, imagine the PostgreSQL shared buffer cache is large enough to hold an entire table. Repeated sequential scans of the table will require no disk access because all the data is already in the cache. Now imagine the cache is one block smaller than the table. A sequential scan of the table will load all table blocks into the cache until the last one. When that block is needed, the oldest block is removed, which in this case is the first block of the table. When another sequential scan happens, the first block is no longer in the cache, and to load it in, the oldest block is removed, which in this case is now the second block in the table. This pushing out of the next needed block continues to the end of the table. This is an extreme example, but you can see that a decrease of one block can change the efficiency of the cache from 100% to 0%. It shows that finding the right cache size can dramatically affect performance.

Proper Sizing of Shared Buffer Cache

Ideally, the PostgreSQL shared buffer cache will be large enough to hold most commonly accessed tables and small enough to avoid swap pagein activity. Keep in mind that the postmaster allocates all shared memory when it starts. This area stays the same size even if no one is accessing the database. Some operating systems pageout unreferenced shared memory, while others lock shared memory into RAM. The *PostgreSQL 7.2 Administrator's Guide* has information about kernel configuration for various operating systems (www.postgresql.org/devel-corner/docs/admin/kernel-resources.html).

Sort Memory Batch Size

Another tuning parameter is the amount of memory used for sort batches. When sorting large tables or results, PostgreSQL will sort them in parts, placing intermediate results in temporary files. These files are then merged and resorted until all rows are sorted. Increasing the batch size creates fewer temporary files and often allows faster sorting. However, if the sort batches are too large, they cause pageins because parts of the sort batch get paged out to swap during sorting. In these cases, it is much faster to use smaller sort batches and more temporary files, so again, swap pageins determine when too much memory has been allocated. Keep in mind that this parameter is used for every backend performing a sort, either for ORDER BY, CREATE INDEX or for a merge join. Several simultaneous sorts will use several times this amount of memory.

Cache Size and Sort Size

Both cache size and sort size affect memory usage, so you cannot maximize one without affecting the other. Keep in mind that cache size is allocated on postmaster startup, while sort size varies depending on the number of sorts being performed. Generally, cache size is more significant than sort size. However, certain queries that use ORDER BY, CREATE INDEX or merge joins may see increases in speed with larger sort batch sizes.

Also, many operating systems limit how much shared memory can be allocated. Increasing this limit requires operating system-specific knowledge to either recompile or reconfigure the kernel. More information can be found in the *PostgreSQL 7.1 Administrator's Guide* (www.postgresql.org/docs/admin/kernel-resources.html).

Disk Locality

The physical nature of disk drives makes their performance characteristics different from the other storage areas mentioned in this article. The other storage areas can access any byte with equal speed. Disk drives, with their spinning platters and moving heads, access data near the head's current position much faster than data farther away.

Moving the disk head to another cylinder on the platter takes quite a bit of time. UNIX kernel developers know this. When storing a large file on disk, they try to place the pieces of the file near each other. For example, suppose a file requires ten blocks on disk. The operating system may place blocks 1-5 on one cylinder and blocks 6-10 on another cylinder. If the file is read from beginning to end, only two head movements are required—one to get to the cylinder holding blocks 1-5, and another to get to blocks 6-10. However, if the file is read non-sequentially, e.g., blocks 1,6,2,7,3,8,4,9,5,10; ten head movements are required. As you can see, with disks, sequential access is much faster than random access. This is why PostgreSQL prefers sequential scans to index scans if a significant portion of the table needs to be read. This also highlights the value of the cache.

Multiple Disk Spindles

The disk head moves around quite a bit during database activity. If too many read/write requests are made, the drive can become saturated, causing poor performance (Vmstat and sar can provide information on the amount of activity on each disk drive).

One solution to disk saturation is to move some of the PostgreSQL data files to other disks. Remember, moving the files to other filesystems on the same disk drive does not help. All filesystems on a drive use the same disk heads. Database access can be spread across disk drives in several ways:

- Moving Databases—**initlocation** allows you to create databases on different drives.
- Moving Tables—symbolic links allow you to move tables and indexes to different drives. Movement should only be done while PostgreSQL is shut down. Also, PostgreSQL doesn't know about the symbolic links, so if you delete the table and recreate it, it will be created in the default location for that database. In 7.1, pg_database.oid and pg_class.relfilenode map database, table and index names to their numeric filenames.
- Moving Indexes—symbolic links allow moving indexes to different drives from their heap tables. This allows an index scan to be performed on one disk while a second disk performs heap lookups.
- Moving Joins—symbolic links allow the movement of joined tables to separate disks. If tables A and B are joined, lookups of table A can be performed on one drive while lookups of table B can be done on a second drive.
- Moving Log—symbolic links can be used to move the pg_xlog directory to a different disk drive. (Pg_xlog exists in PostgreSQL releases 7.1 and later.) Unlike other writes, PostgreSQL log writes must be flushed to disk before completing a transaction. The cache cannot be used to delay these writes. Having a separate disk for log writes allows the disk head to stay on the current log cylinder so writes can be performed without head movement delay. You can use the postgres -F parameter to prevent log writes from being flushed to disk, but an operating system crash may require a restore from backup.

Other options include the use of RAID features to spread a single filesystem across several drives.

Conclusion

Fortunately, PostgreSQL doesn't require much tuning. Most parameters are automatically adjusted to maintain optimum performance. Cache size and sort size are two parameters administrators can control to make better use of available memory. Disk access can also be spread across drives. Other parameters may be set in share/postgresql.conf.sample. You can copy this file to data/postgresql.conf to experiment with some of PostgreSQL's even more exotic parameters.

Bruce Momjian is cofounder of the PostgreSQL Global Development Team. Momjian currently serves as vice president of database development for Great Bridge, LLC (http://www.greatbridge.com/) and is responsible for leading critical database development projects, particularly for PostgreSQL. Momjian is also the author of PostgreSQL: Introduction and Concepts (Addison-Wesley), the first comprehensive guide for this complex open-source database system.

Archive Index Issue Table of Contents

Advanced search

freeVSD Enables Safe Experimentation

Randall Embry

Issue #88, August 2001

Using freeVSD enables self-sufficient systems and can save software headaches.

I work in an environment where software developers and system administrators with varying capabilities aggressively and routinely use Linux. We frequently explore new applications but sometimes hesitate to actually initiate an installation because we lack confidence in the software. Our concern is that rogue software might disrupt essential services on production servers. In extreme cases, it is even possible that a poorly written installer might corrupt a workstation's operating system installation.

Typically, implementing even a slightly complicated application that interacts with the web server requires installing the application, adding a new user (for suid operation), adding lines to httpd.conf, restarting the web server and creating and manipulating files in root-owned places like /etc or /usr/local/. All of this has to be undone if we later decide not to put the system into production use. While uninstall scripts can assist in this, these scripts could fail, leaving the system in an indeterminate state.

freeVSD is a GPL product initially conceived to enable an ISP to provide virtual server hosting. It can also transform a stock Red Hat installation into a powerful, low-cost testing environment. freeVSD works by simulating up to 250 full-featured private servers. Hard links to system files facilitate compact and homogeneous environments for each virtual server. Logins to the virtual servers are restricted via the native **chroot** facility, effectively creating a secure sandbox.

Now, we can experiment recklessly, hand the keys over to inexperienced juniors or casually grant root privileges to strangers, with little concern for negative consequences.

From the system administrator's standpoint, freeVSD enables you to create multiple self-sufficient systems, each with its own administrative account and the ability to manage user accounts, as well as the ability to configure their own web services, mail services, database server—a “Lite” version of Linux, if you will.

freeVSD was originally developed for an ISP in the United Kingdom and has been under development for three years. Based on mailing list archives, freeVSD seems popular and well supported. Questions are answered quickly, either by users or the developers.

Many significant functions of each virtual server can be administered by a rootlike account named Admin. For example, the Admin account can add users, manipulate their privileges, make changes to httpd.conf, restart various aspects of the server and so on.

Installing freeVSD

Installing freeVSD can be a bit tricky. You need to be especially careful if you intend to restore the hostsystem back to its original configuration. As always, it is imperative to back up anything you are not comfortable with catastrophically losing. According to the web site, support for Debian, Mandrake and Slackware is forthcoming, but so far only Red Hat 6.x and 7.x. are officially supported. Version 1.4.6 introduces support for Red Hat 7.0, but Red Hat 6.2 seems to have more of the kinks worked out.

It is recommended that freeVSD be installed on a nearly pristine system. Start with a freshly installed Red Hat 6.2. Then decide whether you want any special server software available, such as MySQL, Postgres or PHP. Apply patches. Ideally, all applications should be installed before configuring freeVSD. Note that freeVSD works quite well under VMware, which might prevent a bit of stress during the first few installs. You'll probably need around 800MB of free disk space to accommodate the filesystem skeleton.

I assume you have or can obtain a FQDN or dedicated IP number for your first virtual host (freeVSD uses IP aliases). Of course, you need to be sure to obtain permission from whomever is in charge of your network before engaging in behavior that might be considered aggressive.

Then choose a name for your first virtual host. A good idea might be the hostname, (e.g., “myhost” if your FQDN is myhost.mydomain.com) or the domain name (mydomain), if you are providing hosting for multiple domains.

Here's an overview of the freeVSD install process, described in detail by the file /usr/doc/freevsd-x.y.z/user-guide.txt.

1. Install main RPM (e.g., freevsd-1.4.6-2.i386.rpm).
2. Install pkgs RPM (e.g., freevsd-pkgs-1.4.6-1.i386.rpm).
3. Run **/usr/sbin/vsd-install.pl**.
4. Run **/usr/sbin/vsd-genskel.pl** (several hundred megabytes will be copied during this process, so be patient). It is simple to customize this installation process. The file /etc/freevsd.conf provides several customization opportunities to specify files to include and exclude during skeleton generation. Red Hat 7.x users may need to tweak /etc/xinetd.conf and/or restart xinetd at this point.
5. Create first virtual host with a command such as **/usr/sbin/vsdadm vs_create localhost *name-of-virtual-server IP-of-Virtual-Server FQDN-of-virtual-server* 200 0**.
6. Execute batch by running **/usr/sbin/vsd-vsbatch.pl**.
7. Start the virtual server(s) with **vsboot --start**.
8. Try out the virtual shell with **/usr/bin/bevs -r [*name of virtual*]** (become the virtual shell).
9. Set the administrative password with **passwd -u admin**.
10. Exit the virtual shell by typing **exit**.

At this point, assuming nothing went wrong, you will have a functioning virtual server to which you may connect via Telnet or FTP.

To uninstall, stop all virtual servers with **/usr/sbin/vsboot --stop**. Next, optionally delete existing virtual hosts with

```
/usr/sbin/vsdadm vs_delete localhost myhost
```

Then, run **/usr/sbin/vsd-uninstall.pl** to restore configurations and optionally delete files. Take care to answer these questions correctly the first time as you won't get a second chance, and you will have to restore configurations manually. Finally, remove the pkgs and main RPMs.

Under the Hood

The /usr/sbin/vsd-genskel.pl installation script copies system files from the host into a directory referred to as the skeleton. This process is controlled by entries in /etc/freevsd.conf, which specifies files to include and delete. The copies are placed in /home/vsd/skel/ by default:

```
$ ls /home/vsd/skel
bin  dev etc  home  lib  proc sbin  tmp  usr
```

The root filesystem of each virtual server relies on hard links to corresponding binaries in this skeleton. A hard link is an additional directory entry pointing to a particular file. Note that hard links differ from symbolic links in that every hard link must be deleted before the file is removed from the system; if the file that a symbolic link points to is removed, the symlinked file becomes unresolvable. Since each server shares the same copies of files that comprise most of the default filesystem, relying on hard links creates a tremendous savings of disk usage.

As far as ps and top are concerned, the freeVSD user processes seem to run as root; however, they don't start with root privileges. By default, UIDS start at 1,000 and increment by 200 for each subsequent virtual machine (i.e., the first vm starts at 1,000, the next at 1,200, etc.). This behavior can be modified by changes to the configuration file, /etc/vsd.conf.

As mentioned earlier, the host machine assumes multiple IP addresses using IP aliasing. A dæmon, /usr/sbin/virtuald, works with **inetd** (or **xinetd**, which replaces inetd in Red Hat 7.0) to intercept client connections to services such as Telnet or FTP. The incoming connection is handed to the appropriate dæmon in the virtual environment by using chroot to the host's filesystem, by default a directory located in /home/vsd/vs/.

Because of potential security exploits, the virtual web server does not run directly on port 80. Instead, a host server process called **vsredirect** forwards traffic from port 80 to port 8080 and moves https traffic on port 443 to port 8443. The file documentation file security.txt details how a malicious user could gain root access without this safeguard in place. Redirection is recommended for all privileged ports below 1000.

Within the freeVSD filesystem, several common commands such as **rm**, **ls** and **passwd** have been modified slightly so that the Admin account has the additional privileges required to administer accounts on the virtual server. This includes the ability to establish user accounts and manage their files.

Separate dæmon processes (httpd, pro-ftpd, sendmail and so on) are created for each virtual host. A suid script allows the Admin of the virtual host to start and stop the dæmons via the /usr/sbin/rebootvs command.

The Admin Experience

The Admin account has simulated root privileges, enabling the Admin user to perform various administrative tasks without seriously compromising the host. The Admin is not a diluted root account; rather, it is an enhanced user account with the limited ability to manage files and accounts on a particular virtual server.

Upon logging in, whoami reports admin and your home directory is /root. Examination of the root (/) directory reveals that Admin owns /root, /home and /tmp. Other Admin-owned files can be determined by running

```
find / -name admin -print
```

You are really running Linux, and it really is the bash shell. It's not a watered-down experience. You can run Python or Perl or even compile new apps with **gcc**. You are in a standard Linux shell environment, and it appears to be your own system.

In /home/httpd/docs one can find the DocumentRoot for the default web server, and /home/web/log contains the log files. The httpd.conf file is located in /etc/httpd/conf/ and may be modified by Admin. As mentioned earlier, /usr/sbin/rebootvs will effectively restart the virtual server by restarting its processes.

Examining /etc/passwd shows the regular system accounts and the admin account; note that this file is read-only. However, **/usr/sbin/useradd *<newuser>*** works as expected, including the addition of the new user to the /etc/passwd file.

The file /etc/vsd/priv has a format similar to /etc/groups. It controls which users have privileges such as login, Telnet and FTP access, as well as whether they are allowed to run Perl or gcc. The /usr/bin/listrights command will show the rights granted to a named user. The /usr/sbin/setrights command is a utility to manage this file; however, from a review of the source code (setrights.c), it doesn't appear that the login privilege may be granted with this utility. It is possible to edit /etc/vsd/priv manually and grant this privilege; since Admin doesn't have write privileges in /etc/, the login privilege can only be granted by root.

Administering freeVSD from the Host

The freeVSD dæmon follows the Sysvinit startup conventions and can be controlled via:

```
/etc/rc.d/init.d/vsd start
/etc/rc.d/init.d/vsd stop
/etc/rc.d/init.d/vsd restart
/etc/rc.d/init.d/vsd status
```

(though this may not yield accurate results).

Several options are configurable through the file /etc/vsd.conf. You may also start or restart a given virtual server with the /usr/sbin/vsboot command.

The /usr/sbin/vsdadm command, introduced during the installation process, can be used to create and delete virtual machines and to manage user accounts. It works by issuing commands to the vsd dæmon that runs on port 1725. This approach enables alternative front ends to manage freeVSD. This includes the administration suite, which consist of tools developed by Idaya, Ltd., some of which run in a web browser or under Microsoft Windows.

The program /usr/bin/bevs (become a virtual server) allows a user on the host to quickly assume the role of root on the virtual machine, without Telneting or otherwise connecting to the virtual machine. Most notably useful is the ability to set the Admin password initially or manage files not owned by Admin or a virtual user.

The following scripts, while useful during install, may become liabilities afterward. You may wish to issue the command **chmod a-x** on them to avoid inadvertently executing them again:

```
/usr/sbin/vsd-genskel.pl
/usr/sbin/vsd-install.pl
```

The program /usr/sbin/**vsd-refreshskel.pl** will update the freeVSD skeleton and then relink files in the root filesystems on each virtual machine. This enables you to add, remove or update applications available to the virtual servers. Be sure to read the documentation or scan the source code before trying this command. Packages you wish to make available to the virtuals, but not to the host, may be installed with the command

```
rpm -ivh --force --root=/home/vsd/skel/ file.rpm
```

Similarly, you may use

```
rpm -ivh --force --root=/home/vsd/vs/some-vs file.rpm
```

to install an RPM into a single virtual server. Unfortunately, since the bulk of the skeleton is created by copying files and not from RPM installations, the RPM database will not accurately reflect the installed packages, so the force option is necessary.

Since /usr/sbin/vsd-refreshskel.pl will refresh the skeleton, it's useful to propagate upgrades to system software on every virtual machine. Please note that this utility seems to be broken for Red Hat 7.x in freeVSD version 1.4.6.

The /usr/share/doc/freevsd-1.4.6/ directory contains documentation and is made available to each virtual machine by default. Finally, /usr/share/freevsd/ contains scripts useful for site customization.

Informal Security Analysis and Discussion

Giving superuser capabilities to regular users is serious business. We frequently see root exploits via shell accounts, so one must wonder if these problems could be compounded in the chroot environment provided by freeVSD.

On the other hand, one operating system patch can instantly improve security for up to 250 web site administrators. Further, having dedicated httpd processes prevents many of the problems usually present in a typical shell environment, where one user's actions could inadvertently disrupt another user's service.

An ordinary user on the host can see the processes of every user on every virtual machine. Further, using the bevs -r command, they can become root on any of the virtual servers by default! The bevs command is suid root, so apparently this is intentional. I recommend changing the permissions in order to disable this (e.g., **chmod o-xr /usr/bin/bevs** will do the trick), while still enabling members of the root group to run bevs.

The virtual server administrative dæmon /usr/sbin/vsd listens on port 1725. Out of the box, it is a trivial task to query a virtual server for user lists, change user privileges—in effect, fully control the vsd dæmon remotely, with no authentication. As suggested in the file security.txt, it is imperative that you control this with commands on the host, such as these (which should be placed in rc.local):

```
ipchains -A input -p tcp -s W.X.Y.Z
<@cont_arrow>å<@$P>--dport 1725 -j ACCEPT
ipchains -A input -p tcp -s 0/0 --dport 1725 -j REJECT
```

Because of the complexity of propagating operating system updates to the virtual machines, it is prudent to run a stable distribution.

Alternatives: Application Testing without freeVSD

Perhaps the simplest way to test out a new application is to keep a spare machine around on which you can safely format the drive and re-install it. While reinstallation can be a good experience builder, continually installing and tweaking becomes a nuisance.

Another option is to create disk images. After you have freshly installed the operating system and have it configured to your satisfaction, a command such as

```
dd if=/dev/hda1 of=/tmp/image
```

can create an image file that can be used to restore the operating system at a later date. Beware of the 2GB file size limitation if using this strategy—the **split** utility will be helpful.

The commercial application, VMware, fully simulates an x86 PC, right down to a full-featured Phoenix BIOS. For maximum performance, VMware directly utilizes the host processor to execute most instructions, instead of emulating the CPU. The VMware approach is not particularly efficient from a disk utilization standpoint: each session must have the operating system completely installed. VMware can create virtual disks from files, so multiple drives are not necessary. In addition to being disk-intensive, memory must be dedicated to each running virtual. However, with VMware, you are not limited to a single Linux distribution or even to the Linux operating system. It offers a more complete virtual server environment but is more resource-intensive, including the need to administer and update each operating system separately. Note that Plex86 (http://www.plex86.org/) is a free alternative to VMware under development.

User Mode Linux offers yet another means. In short, the Linux kernel has been ported to another architecture—a process running inside a regular Linux kernel. Full details are available at http://user-mode-linux.sourceforge.net/.

Conclusion

freeVSD is a promising product. The technique of running multiple server processes for each virtual server results in a robust and reasonably efficient solution. The illusion that the virtual server is a dedicated machine is fairly convincing, but the lack of full control over the virtual's filesystem can be occasionally frustrating. Plus the Admin user's inability to install RPMs is somewhat limiting. But for moderate development, especially where the primary tools you need are already installed and compatible with freeVSD, this is one of the quickest ways to create a fresh Linux system. The chances of inadvertently goofing up the host from a virtual server are negligible.

Randall Embry (randall@embry.com) lives with his wife, four-year-old daughter and two cats in Bloomington, Indiana. Last year, he traded in his laptop for a 28-pound “luggable” with a 15-inch LCD screen. (It runs Linux, of course.) In addition to programming and aggressively enjoying Linux, Randall manages a team of programmers and network engineers at Dataworks, the IT consulting division of Fine Light. See http://www.embry.com/randall/ for more information.

Kernel Tuning Gives 40% Gains

Rick Gorton

Issue #88, August 2001

Adjusting assembly language routines for higher performance from Alpha processors.

API NetWorks is a leading developer of high-performance, high-density servers. As such, we (and our customers) are acutely sensitive to system performance issues. With the majority of our customers running Linux on Alpha-based systems, we pay close attention to key open-source components and packages to ensure we don't inadvertently leave performance “on the table”.

In the proprietary world, improvements unearthed after software product releases are saved for the next version. In the world of open-source software, the time between improvements can be measured in hours or days. In addition, tuning techniques are shared quickly for the enhancement of the communal knowledge base. With this in mind, we examined the assembly language routines in the Linux kernel, with an eye toward taking advantage of the micro-architectural features of the latest Alpha 21264 (also known as ev6) processor.

We discovered that by rewriting a handful of routines in the architecture-specific portion of the Linux kernel to take advantage of the features of the 21264, we could attain significant speed improvements, amounting to a 40% reduction in overhead for some workloads. This article describes how and why these optimizations work, so that others may be able to use these techniques to boost their Alpha applications performance. For an e-mail server running on a 21264-based Alpha, for example, applying these tuning tips will enable it to scale to higher loads, extending performance beyond where it would normally tend to taper off. End users will feel as though their system is running faster; systems managers will be pleased because they won't have to buy another system as soon to handle increased traffic.

In the case of the Linux kernel, the performance-critical routines are carefully (and conveniently) separated by architecture and mostly implemented in assembly language. While there is a large body of developers continually working to improve the algorithmic performance of the kernel, there are only a handful of developers with the requisite interests, skills and knowledge to enable them to tune these assembly routines to extract the maximum performance from a processor. Most of these developers are tuning code for x86 systems; the relevant Alpha code base had been dormant for quite some time prior to our tuning.

Anecdotal evidence and previous measurements suggest a disproportionately large amount of time was spent inside the Linux/Alpha kernel when running various web and networking servers. Given that the number of known performance-critical routines in the Linux kernel is relatively small (20-30 assembly language routines), it made sense to sort through them to make sure the code was written with the 21264 in mind. A quick reading of these routines revealed they had been carefully hand-scheduled for the previous generation of Alpha CPUs, the 21164 (also called ev5).

While the difference may seem minor, the 21264 differs significantly from the 21164 at the micro-architectural (chip implementation) level, having features that result in roughly double the performance of the 21164 at the same clock speed.

Before detailing the performance improvements, it is useful to note some of the differences between the 21164 and 21264 processors (see Table 1).

Table 1. Comparison of Alpha 21164 and 21264 Processors

After realizing a careful rewrite could yield significant performance gains, we undertook the tuning of the Alpha assembly language routines in the Linux kernel. (There have been attempts to solve dynamically and automatically the problem of efficiently running binaries compiled for older versions of Alpha CPUs on new Alpha CPUs1, but that is beyond the scope of this article.) Specifically, it was clear that about 20 routines in linux/arch/alpha/lib and four to six routines in linux/include/asm-alpha needed to be rewritten to take full advantage of the features of the 21264.

Improvement Sources

From an internal perspective, the 21264 is a significantly more complex and powerful CPU than the older 21164. Because of these differences, the nature of the performance improvements can come from transformations (discussed below) involving the rescheduling of the code in order to keep the processor from stalling on fetch, avoid branch and branch target penalties, avoid replay traps from occurring where possible, use the instruction latencies and scheduling rules of the 21264, use instructions available in the 21164 and 21264 that had not been used, use instructions newly available in the 21264 and not use some instructions if possible on the 21264.

Preventing Fetch Stalling

When writing code to avoid fetch stalling, one must ensure the instructions dynamically encountered in a fetch block do not result in one or more of the functional units being over-subscribed. On the 21264 it is also important to ensure that branch targets are aligned on a fetch block boundary. Given that it fetches four instructions at a time, branch targets should have an alignment of 0mod4 instructions, equivalent to an address alignment of 0mod16.

Avoiding Branch Penalties

It is particularly important to avoid branch penalties on the 21264. Sophisticated, trainable branch prediction logic is built in and works effectively if only one control flow change instruction is in a fetch block (a “quad-pack”). In the 21164-tuned kernel assembly language routines, there are a number of places where multiple control-flow change instructions occur within a quad-pack. Additionally, branch targets were aligned to 8mod16 addresses, which often resulted in branch target labels appearing in the middle of a quad-pack. While these sequences run quite well on a 21164, they run relatively slow on a 21264.

Avoiding Replay Traps

Replay traps occur when the processor must roll back the state of memory to force accesses to a particular memory location in order to be sequential, or when there are different-sized accesses to the same memory location. But the code context in the modified routines was such that replay traps were not an issue, so rewriting sequences to avoid replay traps was unnecessary.

Instruction Latencies of the 21264

The instruction scheduling and slotting rules for the 21264 are too complex to list here, but for those interested in the details, the 21264 *Compiler Writer's Guide* is an excellent reference.

Available but Unused Instructions

Byte- and word-sized loads and stores were introduced in the 21164A (ev56) processor, but they were not used in the original versions of the assembly language routines. Prior experience (in the context of static binary translation of applications) has shown performance can be typically improved 10% to 20%

by utilizing these instructions. This is particularly true for the stw (store word) and stb (store byte) instructions, as it eliminates memory traffic in a way guaranteed to cause replay traps on the 21264. In the context of the tuned-up kernel routines, these instructions were helpful, but it was typically localized to the tail code of large region copies, while the bulk of the data movement used eight-byte granularity load and store instructions.

The Alpha architecture also features various forms of pre-fetch instructions. Pre-fetch instructions are hints to the memory subsystem to fetch a block of memory to the data cache for future consumption. These do not normally appear in compiled code, as few compilers have enough context available to permit their generation; Compaq's compilers do generate pre-fetch instructions. In the context of moving large amounts of data, it is possible (and desirable) for the assembly language programmer to utilize pre-fetches. The __asm__() feature of gcc enables programmers to insert relevant pre-fetch instructions at key points in routines when rewriting entire routines in assembly language is undesirable. Because they can minimize or prevent data-cache stalls, using these instructions can significantly boost performance.

New Available Instructions

The 21264 is the first Alpha implementation to include support for three instructions useful for boosting performance: CTLZ, CTTZ and WH64.

The CTLZ and CTTZ instructions count the number of leading/trailing zeros in a 64-bit register and are handy for string manipulation. When a program performs string operations involving pattern matching (strlen() matches on NULL), it is often the case that the byte-number index of the pattern match in an eight-byte value in a register is needed. Without CTTZ, it takes about ten instructions involving multiple CMOVxx (conditional move) instructions to determine this index. The result is a reduction in code size (always useful), as well as a decrease in the number of cycles needed to perform string operations. Also, there are some filesystem primitives involving finding holes in a bitfield where these instructions are useful.

WH64 (write hint for 64-bytes) is a memory subsystem hint that a specified 64-byte region is going to be written to in the near future. The processor can pass this information to the memory subsystem, which can invalidate the target contents and avoid some number of memory system cycles to keep the memory state coherent. Since a process context switch entails moving large amounts of information in memory from one place to another, any improvement in copying performance between kernel-space memory and user-space memory is good news. Meanwhile, program load time is another place in the operating system that depends upon doing a lot of memory-to-memory traffic. The program bits all have to get mapped, and all of the zeroed memory (.bss in executables) must have zeros written to it.

Instructions to Avoid

The CMOVxx conditional move instructions on the 21264 have been implemented in the hardware by decomposing them into two separate instructions inside the processor. The result of the latency of a CMOVxx instruction is a minimum of two cycles, and can take up to five cycles, depending upon the number of CMOVs in a given fetch block. In some situations, replacing CMOV instructions with highly predictable conditional branches can result in a performance gain on the 21264. Overall, a good rule of thumb is to try to minimize the number of CMOV instructions if possible.

Data Collection and Test Methodology

The data was collected from an API NetWorks' CS20 server, which has dual 833MHz processors with 4MB DDR cache, 1GB of SDRAM and Ultra-160 SCSI disks. Two load-generation tests were run: five builds of the 2.2.18 kernel and five builds of gcc-2.95.3. The average system time (as reported by /usr/bin/time -p) was recorded, using various levels of parallelism with **make** (see Tables 2 and 3).

Table 2. Load-Generation Test Results of the 2.2.18 Kernel

Table 3. Load-Generation Test Results of gcc-2.95.3 Kernel

A similar version of the experiment was run using the 2.4.2 kernel in default mode (all of the performance patches exist). The results were compared to an unpatched 2.4.2 kernel with most (but not all) of the performance changes reverted.

Table 4. Load-Generation Test Results of Default 2.4.2 Kernel

Table 5. Load-Generation Test Results of gcc-2.4.2 Kernel

This experiment was initially performed on an API NetWorks' UP1000 motherboard system, which has a 700MHz processor with 4MB cache, 128MB SDRAM and IDE disks. Again, five builds of the kernel and gcc were run, and the average times were recorded. The kernel used was 2.4.0-test6, with and without the patches.

Table 6. Test Results of UP1000 System Running 2.4.0 Kernel

Table 7. Test Results of UP1000 System Running gcc-2.4.0 Kernel

On a modestly configured 21664 system (the UP1000), the performance increase is significant in terms of reducing the amount of time spent in the kernel, with improvements in the 40% range for some activities (kernel builds). On a more generously configured CS20, we consistently attained speed increases of 14-15% for the measured loads.

We attribute the differences between the UP1000 and CS20 systems to be related to their memory: the UP1000 has an 800MB/sec, 64-bit bus, while the CS20 has a 2.65GB/sec, 256-bit bus.

Summary

All of the rewritten routines have appeared in one form or another (some have undergone subsequent rewriting) as part of the 2.4.2 kernel. Additionally, we have put together a patch for 2.2.17 of the kernel and made it available on our corporate web site, http://www.api-networks.com/products/downloads/developer_support/ under "Performance". Through additional efforts, these improvements have also migrated into glibc and will eventually help improve application performance of user-mode code.

Rick Gorton, a member of the technical staff at API NetWorks, was first introduced to Linux in the form of the original 32-bit port to Alpha (BLADE). He has been developing binary translators, optimizers and other binary manipulation tools for Alpha since 1992. He can be reached at rick.gorton@api-networks.com.

XMLC

Reuven M. Lerner

Issue #88, August 2001

Reuven introduces XMLC, part of the Enhydra application server.

Over the last few months, we have looked at a variety of methods for creating web applications using server-side Java. We started with simple servlets and then moved onto JavaServer Pages (JSPs). In order to remove Java code from our JSPs, we began to use JavaBeans, objects whose methods are automatically available to our pages.

But you can only go so far with JavaBeans, which is where custom actions come in. These actions, which look like XML tags and attributes in our JSPs, are tied to the methods of a Java class. In other words, placing a tag in our JSP can effectively invoke one or more methods. Combining custom tags with beans allows us to remove quite a bit of the Java code from our JSPs.

But in the end, what have we accomplished? As we saw last month, intelligent use of custom actions means creating our own mini-language, with its own loops, conditionals and variables. Writing our own tags saves graphic designers from having to use Java and allows us a greater separation between form and content. But it does not go nearly far enough in solving problems.

One clever solution is part of the Enhydra application server, about which I will be writing over the next few months. XMLC, or the XML compiler, turns XML files (including HTML and XHTML files) into Java objects. By invoking methods on these objects, we can modify the HTML that is eventually produced.

XML and XHTML

XML, as you have probably heard by now, is the extensible markup language. What began as a simple and small standard several years ago has ballooned into a veritable alphabet soup of standards and proposed standards.

But the core of XML has remained the same, allowing people to create their own markup languages using a uniform syntax. XML is not meant to be used directly; rather, it is meant to let you create your own markup languages. Because those markup languages are based on XML, they have a well-understood syntax that can be verified by any XML parser. Moreover, if you define a data type definition (DTD) for your markup language, a verifying parser can ensure that the elements and attributes are within accepted norms.

HTML and XML are both standards of the World Wide Web Consortium (W3C), have a similar syntax and are often discussed in the same breath. But in fact, HTML is just one markup language, while XML allows you to create your own languages. More significantly, HTML has a much looser syntax than XML, thanks in no small part to historical factors. The following is thus legal HTML: **<img src="foo.png">**.

But because every tag must be explicitly closed in XML-derived languages, this would be illegal in an XML document. Instead, we would have to say: **<img src="foo.png"/>**.

In order to bridge the gap between HTML and XML, the W3C has issued a recommendation known as XHTML, the XML implementation of HTML. While there are indeed various benefits to the use of XHTML, the biggest one is that XML tools will now work on our HTML documents.

Of course, this means that our XHTML documents will look a bit more formal than the HTML documents we might be used to writing. While HTML allows us to be sloppy, using <P> to separate paragraphs, XHTML is much stricter, forcing us to begin paragraphs with <P> and end them with </P>. Attributes must also appear in double quotes, which many people fail to do when working with straight HTML.

While XHTML might be a pain for humans, it actually reduces the load on programs by making the syntax more regular, and thus easier to read and write. But the biggest benefit is the fact that XHTML documents can now be treated as XML documents.

The DOM

XML documents are trees, which should ring a bell for those of you who studied computer science in college. Trees are remarkably easy to work with in theory, but the practice can be a bit tricky sometimes, depending on the way in which the interface is implemented.

There are two popular and cross-platform APIs for working with XML: SAX (the Simple API for XML) is designed to work with incoming streams of XML data, allowing it to be small and efficient. The DOM (document object model), by contrast, gives us access to the entire document tree at once. This allows us to traverse and modify nodes, including adding new nodes and removing old ones. However, it also means that the entire document must be loaded into memory before we can begin to work with documents using the DOM. This makes it more powerful than SAX, but also slower and more resource-intensive.

XMLC works by converting an XML file, normally written in HTML or XHTML, into a Java class that creates and manipulates a DOM tree. You can use standard DOM methods to add, modify and remove nodes on the tree, thus changing the document that will eventually be output.

But the truly clever idea in XMLC is the use of HTML “id” attributes. When the XMLC complier sees an id attribute, it creates methods that allow us to retrieve and modify the text contained within that attribute. The site designers thus work with HTML, identifying areas of dynamic text by giving them unique identifiers. When the designers have finished with their mockup of the original HTML page, they compile it (using XMLC) into a Java class. Developers then create servlets that instantiate that class, use methods to replace the mockup text with dynamically generated content and send the document to the user's browser.

The basic idea is that the designers do not work on hybrids of text and HTML, but rather on mockups of the final output. So long as the id attributes do not change, the HTML file and servlet can evolve in parallel, with neither designers nor developers waiting for their counterparts.

Installing Enhydra

As I mentioned above, XMLC is one element of the Enhydra application server. The 3.x version of Enhydra is considered to be production-ready and includes a copy of XMLC that most users will find more than adequate. Because I am particularly interested in Enhydra for working with Enterprise JavaBeans (EJB), I have been working with the beta version of 4.x, otherwise known as Enhydra Enterprise. By the time you read this, the final release of Enhydra Enterprise should be available, giving web developers an open-source, production-quality J2EE-compliant application server.

To work with XMLC, I downloaded the Enhydra Enterprise beta, a 15.7MB file named enhydra4.0.tar.gz. Open this file, and you will find a wealth of libraries, applications and documentation for the Enhydra application server. We will ignore much of this for now, concentrating on XMLC for the time being.

Almost all of Enhydra is written as Java classes invoked from shell scripts. In order for the shell scripts to find the Java classes, they must be configured for your particular installation. You can do this by entering the Enhydra directory (enhydra4.0 on my system) and running the **configure** script:

```
./configure /usr/java/jdk1.3
```

configure normally takes a single argument—the root directory of your JDK 1.3 installation. While earlier versions of Enhydra (and particularly earlier versions of Enhydra Enterprise) wouldn't work with JDK 1.3, current versions will only work with 1.3. Since JDK 1.3 has a number of other benefits, and a Linux version is supported by Sun, it is probably a good idea to install it.

If you have installed Enhydra somewhere other than /usr/local/enhydra, you should probably set the ENHYDRA environment variable to your installation directory.

Full use of XMLC depends on placing three different .jar files in your CLASSPATH. Since we will be concentrating on XMLC for the rest of this article, we should probably add them now, using bash syntax:

```
export CLASSPATH=$ENHYDRA/lib/xmlc.jar:\
$ENHYDRA/lib/enhydra.jar:\
$ENHYDRA/lib/xmlc-support.jar
```

If you're like me, you will want to have a number of items in your CLASSPATH in addition to Enhydra-related items. Here is how I set my CLASSPATH, for instance:

```
export CLASSPATH=$ENHYDRA/lib/xmlc.jar:\
$ENHYDRA/lib/enhydra.jar:\
$ENHYDRA/lib/xmlc-support.jar:\
$TOMCAT_HOME/classes:\
$TOMCAT_HOME/lib/servlet.jar:\
/usr/share/pgsql/jdbc7.1-1.2.jar:\
.
```

Notice how I placed the Enhydra .jar files before the others on my system in order to avoid potential problems with conflicts. Since Enhydra has the newest versions of some classes, such as those having to do with the DOM, they should take precedence.

Note that not all three Enhydra-provided .jar files are necessary for each stage of working with XMLC. However, I found it convenient to include all of them at all stages in order to avoid unpleasant surprises later on.

A Simple HTML File

Now that we have installed everything we need to work with XMLC, let's try it with a simple HTML file:

```
<html>
    <head><title>This is a title</title></head>
    <body>
        <h1>This is a headline.</h1>
        <p id="firstpara">This is a paragraph.</p>
        <img src="foo.gif"/>
        <p>This is a second paragraph.</p>
    </body>
</html>
```

While XMLC works just fine with straight HTML files, XHTML is a better idea because it stops us from generating files that the DOM cannot represent. For example, XML forbids overlapping tags:

```
<i><p>Wow</i>, he thought.</p>
```

The above is tolerable HTML but is illegal XML and XHTML. So while your web browser can somehow handle this HTML and make sense of it, XMLC will generate a warning indicating that it is discarding what it considers to be a useless closing tag. XMLC will often warn you when your HTML is not well formed, helping you to identify potential problems. While you might not have to consider your document's structure when you are writing simple HTML documents, the manipulations that you can perform with XMLC require that you have a clear understanding of how your document will be rendered.

The first paragraph in the previous sample statement is identified with the id attribute “firstpara”. We will soon see how we can manipulate that text from within a Java program, using the id as a lever into the document.

To turn our document into a Java class, we invoke the **xmlc** program. Assuming that our above HTML file was called foo.html, we can say:

```
$ENHYDRA/bin/xmlc -parseinfo -verbose -keep foo.html
```

This turns foo.html into a Java source file called foo.java, which is in turn compiled into foo.class. The -keep argument retains foo.java, rather than deleting it once it has been compiled into foo.class. And while they are unnecessary, I like to use -parseinfo and -verbose when working with xmlc, if only to get some visual feedback on the compilation process.

The Java source code created by XMLC is fairly long and boring, if well-commented. For those of us who want to modify foo.html, the most important parts of foo.java are the getElementFirstpara() and setTextFirstpara() methods. The former returns the text associated with the id “firstpara”, while the latter allows us to swap that text with an arbitrary string.

Listing 1 contains the source code to a small command-line Java class (PrintFoo.java) that prints the contents of the Java-ized version of foo.html. Before printing it, it uses setTextFirstpara() to modify the output:

```
   myfoo.setTextFirstpara("This has been changed");
```

Once we have made that change, we can display the document:

```
   System.out.print(myfoo.toDocument());
```

We could traverse the DOM tree ourselves, looking for nodes with a certain id and then modify it manually. However, XMLC's convenience methods make it extremely easy and straightforward to modify such text.

Listing 1. PrintFoo.java

If you have just run PrintFoo, you will notice that the output HTML is displayed without any of the original white space. The resulting document is harder for humans to read but is rendered identically by browsers. That said, I have always tried to keep my HTML documents formatted correctly for easier debugging, and it would be nice for XMLC to include a -preserve-whitespace option.

From what we have seen so far, it would seem that XMLC makes it easy to modify entire paragraphs but difficult to change a single word. However, XMLC takes advantage of the HTML “span” tag, which takes an id attribute and allows us to identify individual words, characters and images that we might want to modify. For example:

```
   <P id="para">This is a paragraph,
       <span id="phrase">and this is a phrase</span>.
   </P>
```

When we compile this HTML using XMLC, we will be able to modify the contents of the entire paragraph using the SetTextPara() method and the individual phrase using the SetTextPhrase() method.

Servlets

Now that we have seen how to work with XMLC from the command line, let's look at a servlet that accomplishes the same task. For starters, our simple PrintFooServlet servlet will receive an HTTP request and will return a copy of the document.

Listing 2 contains a copy of the servlet that displays a foo.html. Like its command-line counterpart, it creates an instance of our “foo” class, modifies some of its text and then writes a textual representation of the XML tree to an output stream. In this particular case, however, the output stream is connected to the user's browser. The user thus sees the modified template without knowing that two Java classes (and an original HTML document) were involved.

Listing 2. PrintFooServlet.java

For our servlet to work, I needed to put a copy of foo.class in a directory located under the Jakarta-Tomcat servlet engine's CLASSPATH environment variable. I chose to put it in $TOMCAT/classes, at the top level. If this were a production class, I would undoubtedly want to put it in a more intelligent place, taking advantage of Java's hierarchical namespace. However, I executed xmlc without specifying a package, meaning that foo.class must be put in the top-level namespace. In order to place foo.class in the il.co.lerner namespace, I would have had to use the -class option:

```
$ENHYDRA/bin/xmlc -class il.co.lerner.foo\
-parseinfo -verbose -keep foo.html
```

With foo.class in $TOMCAT/classes, I was able to compile PrintFooServlet.java successfully. Now the only remaining challenge was to execute this servlet and display my modified HTML page. Once again, I needed to modify the CLASSPATH, but this time the CLASSPATH in need of change was that of the Tomcat servlet engine, which executes servlets on our behalf. I modified $TOMCAT/bin/tomcat.sh such that just before it exports its CLASSPATH, we add the three Enhydra-supplied .jar files and restarted Tomcat. Moments after pointing my browser at the servlet, I was delighted to see a modified version of my original HTML file on my screen.

Databases and XMLC

It is easy to see how we could populate a page with information taken from a relational database. For example, here is a small PostgreSQL table that we can use to store a different saying for each calendar day:

```
CREATE TABLE DailySayings (
    date   TIMESTAMP NOT NULL,
    saying TEXT NOT NULL,
    UNIQUE(date)
)
```

Now let's insert a number of sayings into our system:

```
INSERT INTO DailySayings(date, saying)
VALUES (CURRENT_DATE,
     'A bird in the hand is worth two in the bush.');
INSERT INTO DailySayings(date, saying)
VALUES (CURRENT_DATE+1,
     'A penny saved is a penny earned.');
INSERT INTO DailySayings(date, saying)
VALUES (CURRENT_DATE+2,
     'The rain in Spain falls mainly in the plain.');
```

To retrieve today's saying, we merely need the following query:

```
SELECT saying
FROM DailySayings
WHERE date = CURRENT_DATE
```

In order to write a servlet that displays today's saying, we will need two classes: a template that we will create with XMLC (saying.html, which will be compiled into saying.class) and another that will load and manipulate the template (DailySaying.java). We will agree in advance of writing our XMLC document and our manipulation class that the id “saying” will link the two together.

Our XMLC document is fairly straightforward:

```
<html>
   <head><title>Today's saying</title></head>
   <body>
      <h1>Today's saying</h1>
      <p>And now, as you requested, today's saying:
        <span id="saying">Saying Goes Here</span>.</p>
   </body>
</html>
```

I compiled this HTML document into the Java class il.co.lerner.saying, keeping around the .java file just for fun:

```
$ENHYDRA/bin/xmlc -class il.co.lerner.saying\
-parseinfo -verbose -keep saying.html
```

I then copied the resulting saying.class file into $TOMCAT_HOME/classes/il/co/lerner, where I keep my servlet-related classes.

Once I installed my document, I had to write a manipulation class. This class executes the SQL query that we saw above, retrieving the results and sticking them into our compiled XMLC document. Listing 3 [see Listing 3 at ftp://ftp.linuxjournal.com/pub/lj/listings/issue88/] contains the source code for our servlet, which I compiled and put into an active servlet context on my Tomcat server. After restarting Tomcat and Apache, I was able to retrieve today's saying via my web browser, with the SQL results instantiated into the HTML document.

Is XMLC a Good Solution?

When I first began to look into XMLC, I had my serious doubts about its viability. After years of working with hybrid templates, it just seemed too weird to turn an HTML file into a Java class, only to manipulate that class using the DOM. And indeed, it takes significantly greater resources to fire up a DOM parser than to simply display a file.

As I have begun to work with XMLC, however, I am increasingly aware of its advantages over such templates. In essence, XMLC forces designers and developers to create a contract, or API specification, between their documents and programs. Once this API is in place, it cannot easily be changed, which is not necessarily a bad thing. Most importantly, the stability of the API between designers and developers allows them to work in parallel, barely interfering with each other's work.

Because a Java manipulation class can modify the HTML of a compiled document in any way it chooses, we can easily imagine a situation in which we bring in three classes at a time: a header file, the main body of the document and a footer file. Our class could then use DOM methods to attach the header to the beginning of the document and the footer to the end. In such a way, we could add global formatting to our site without having to copy boilerplate text to the top of each file.

There are, of course, a number of irksome details when working with XMLC. One is that it quickly gets boring and frustrating to write one servlet per HTML file. True, we could write a single servlet that takes the name of a file in its query string, acting almost as a document template for a variety of classes created by XMLC. Perhaps I have not yet explored Enhydra enough to have discovered the answer to this question, and perhaps Enhydra developers quickly get used to creating two Java classes for each page they wish to display. Regardless, this can quickly create an overwhelming number of classes, even on a small- to medium-sized site.

The biggest problem that I see with XMLC is the lack of a high-level API to manipulate HTML (and XML, for that matter). One of the FAQs for XMLC is “How do I add a row to an HTML table?” Such a task, which is trivial to accomplish with standard HTML, quickly becomes a burden with XMLC. You must first find the bottom of the table to which you want to add rows and then add individual nodes (and attributes) to that node. It has a very non-HTML feel to it and forces the developer to think of nodes when he or she would prefer to think in terms of HTML. Given that Enhydra includes an API to create SQL queries using Java methods, I would imagine that a similar API for HTML manipulation wouldn't be too difficult.

Conclusion

XMLC is an intriguing technology that sits at the heart of the Enhydra application server. XMLC forces developers (and designers) to consider how they will interact before they begin working and then allows them to work independently. While this mode of operation might throw experienced template users off balance, it quickly becomes second nature and feels more natural than I ever expected.

Indeed, the fact that Zope's ZPT uses a similar method for separating form from content probably points to a trend within the web development community. We can expect to see more XMLC-like systems in the near future. If we're lucky, perhaps there will even be some standardization of these templates, so that designers can move across systems without having to learn the subtle differences between them.

While XMLC is important, Enhydra has many other features that make it worth investigating. Next month we will continue to look into Enhydra, looking at ways in which it speeds up the writing of server-side database applications.

Resources

Reuven M. Lerner owns a small consulting firm specializing in web and internet technologies. He lives with his wife Shira and daughter Atara Margalit in Modi'in, Israel's “city of the future”. You can reach him at reuven@lerner.co.il, or on the ATF home page, http://www.lerner.co.il/atf/.

Launch, Not Lunch Platforms!

Marcel Gagné

Issue #88, August 2001

Marcel provides a sampling of application launchers.

You know, François, I am having trouble staying focused on this menu. My mind, it keeps wandering. Eh? Well, you see, *mon ami*, the topic of this month's issue is platforms. I keep thinking about launch pads. You know, *la navette*, the space shuttle, catapults, all sorts of launchers. Even when I try to get my head out of the clouds and try to focus on our Linux systems, I think application launchers. I think my brain, it is stuck on this whole launch pad business.

What? But François, surely you know what I mean by application launchers. The big foot in the bottom left-hand corner of that GNOME desktop at table eleven is an example, just like the big K on the KDE desktop at table six. Then, of course, there are the little buttons on KDE's kicker or GNOME's panel. Nevertheless, people have been experimenting with other ways of launching applications for a long time. For instance...ah, *mes amis*! *Bienvenue*! Welcome to *Chez Marcel*. François and I were just discussing application launchers, and I was about to give him some examples. Please sit down and be comfortable.

François, what are you doing just sitting there? *Du vin, mon ami*. *Vite*! The 1989 Alsace Riesling is drinking wonderfully. It is time to let our guests sample its grandeur, *non*? *Vite*, François.

I was showing François the program launchers in KDE and GNOME, but there are other desktops out there, and besides, we still put icons on our desktops for easy access. For those of you who, like me, enjoy a little Window Maker from time to time, let me start you off with a single-button dock application that is really quite good despite its name. Written by Alexandre Beraud and Emmanuel Sunyer, WMbad is a good little program launcher with some slick features such as theme support and scroll or fade effects between applications. Just click the arrow and watch the little pixmaps slide from one to the other.

Why would you do such a thing? That is a good question. Well, I don't know about you, *mes amis*, but my desktop is at times a little, shall we say, crowded. A small launch application that requires little room does not seem like such a bad idea. Have a look at Figure 1 for a picture of WMbad in action.

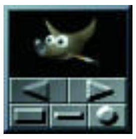


Figure 1. Not much space. Not bad for WMbad.

You will of course want to build this yourself, *non*? Start by picking up the source tarball at perso.mnet.fr/esunyer/wmbadeng.html. The installation is easy but with a few more steps than we are used to. There is a theme and pixmap directory that need to be copied into your home directory:

```
tar -xzvf wmbad-0.3.0.tar.gz
cd wmbad-0.3.0
make
make install
mkdir $HOME/.bad
cp config.bad $HOME/.bad
cp alpha.xpm $HOME/.bad
cp -R pics $HOME/.bad
cp -R themes $HOME/.bad
```

Ah, François. *Merci*. Please, pour for our guests. I promise you, *mes amis*, you will love this wine. Meanwhile, the compile, as you can see, is done. You can now start the application by typing **wmbad &**. Everything you need to customize WMbad is available from the application itself. Need a quick xterm? Just click the little bar on the left-hand side. Click on the little round button on the right-hand side and an editor session pops up allowing you to update the config file. You will find the format extremely easy to work with, see the following sample. Note the line that reads “Commands number”. If you add commands, you need to increment that. For instance, in the following example, I have seven scrolling commands defined (the default file has four):

```
[Theme pixmap]
default.xpm
[NONE/SCROLL/STORE]
SCROLL
[Commands number]
7
[Com 1]
emacs
editor.xpm
```

Now, I happen to know that some of you are running AfterStep. The next item on our menu is for you. Actually, *mes amis*, this can be for everyone since I have run it from my KDE and Window Maker desktops. The application is called **asbutton** and comes to us from Ryan Lathouwers. You can get the latest copy at home.pacbell.net/ryanlath/asbutton.html.

Next, unpack the source and build your launcher:

```
tar -xzvf asbutton-0.3.tar.gz
cd asbutton-0.3
make
make install
cp .asbuttonrc $HOME
```

The final command copies the default configuration file to your home directory. The format of the file is simple and well documented so adding additional commands should be very easy. If you are eager to see the program in action with its default batch of commands, then launch the asbutton program launcher (ahem) by typing **asbutton &**.

If you should wish to run a nine-button configuration, as opposed to the four-button default, use the -n 9 flag. In order to make the space used by this tiny launcher as productive as possible, each button can launch multiple applications. Click the left button and the **top** program starts. Use your right mouse button instead and you launch **xosview**. Do that with nine buttons and you have one-click access to 18 programs while taking up a meager 64 × 64 pixels (see Figure 2).

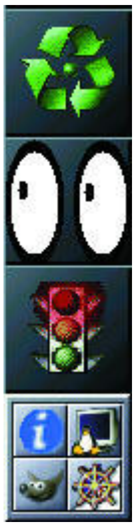


Figure 2. asbutton Program

If you do wish to have this startup attached to your AfterStep wharf, then add the following line to your wharf configuration file:

```
*Wharf asbutton - Swallow "asbutton" asbutton  &
```

If you are having trouble locating it, the wharf config file should be located at $HOME/GNUstep/Library/AfterStep/wharf.

Another such launcher I would like you to look at is attractive because of its name. Would you believe **minibar**? Upon looking at Michael Eddington's minibar, I was at first disappointed to find it completely empty of even a humble table wine. *Le truc*, in this case, was to remind myself that this is software and not a real minibar. This is a small, vertical program launcher that takes up virtually no room on your desktop. The icons are tiny, providing single-click access to whatever commands you see fit. For starters, pick up your copy at http://www.phed.org/miniProject/.

A little word of warning here, and no, I do not mean that you should be careful not to overindulge in the soufflé. Indulge all you like, *mes amis*. François, more wine! *Non*, what I am talking about is this. There is a precompiled binary included with the distribution, but I ran into runtime library problems. The simplest way to deal with the problem is to extract the software, remove the old binary, then build anew:

```
tar -xzvf minibar-0.05.tar.gz
cd minibar-0.05
make clean
make
make install
cp minibarrc $HOME/.minibarrc
```

As you can see, the last step involves copying a configuration file into your home directory. Once again, this is a simple file to work with. Each button (and command) is defined simply (see Figure 3). For instance, if I wanted to add a button that started a copy of the GIMP, I would add these lines:

```
/usr/include/X11/pixmaps/mini-palette.xpm
The GIMP
gimp
```

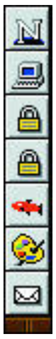


Figure 3. minibar

The first line is the path to one of your mini-icons (the installation copies a handful into your /usr/include/X11/pixmaps directory), but you can use other mini-pixmaps on your system or create your own. In the following example, I had a 48 × 48 PNG image for the GIMP that I wanted to convert to a 16 × 16 size and then save as an xpm file. A little ImageMagick and *voilà*:

```
convert -geometry 16x16! gnome-gimp.png gimp.xpm
```

The **convert** program is part of the ImageMagick suite of tools and is likely already installed on your Linux system. Other interesting commands you may wish to explore include **mogrify** and **identify**. If, perchance, you do not have ImageMagick installed, check your installation CD-ROM or visit http://www.imagemagick.org/.

The truth, *mes amis*, is that application launchers like these are simple menus with a little flash. Since the core is still simplicity, I thought I would add one final item on today's menu, one that is more closely related to the other type of launch pad. *Oui*, it is time to launch our ships into space and protect our fair planet from invading alien hordes. In keeping with our theme of simplicity, allow me to present you David Slimp's *Perl Defense Blaster*. This is an arcade-style game written entirely in Perl that runs on an ASCII screen with very simple graphics. It is also strangely addictive. Get your copy of *Perl Defense Blaster* at http://perlblaster.sourceforge.net/.

You may also need to visit CPAN (http://www.cpan.org/) to pick up a couple of modules: Curses.pm and TermReadKey.pm. You can also get these via FTP at the following addresses: ftp.cpan.org/CPAN/modules/by-module/Curses and ftp.cpan.org/CPAN/modules/by-module/Term.

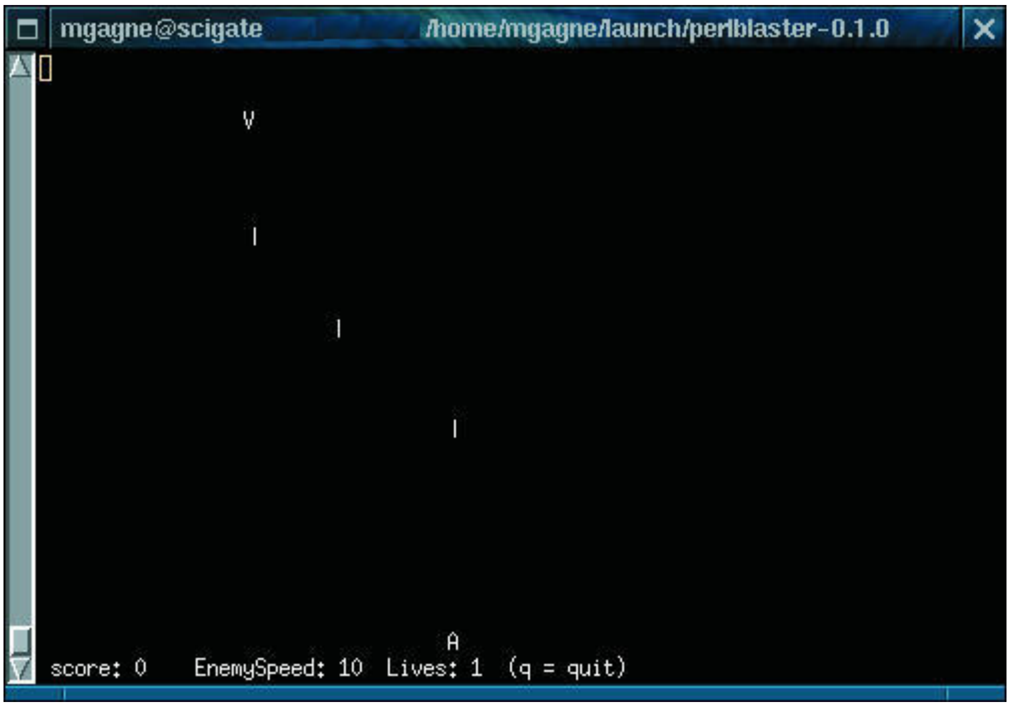
It is true that I have covered the process for installing Perl modules before, but nevertheless, here is a quick reminder. We will use the Curses.pm module as an example:

```
cd /usr/local/temp_dir
tar -xzvf Curses-1.05.tar.gz
cd Curses-1.05
perl Makefile.PL
make
make install
```

Because *Perl Defense Blaster* is just a script (and consequently another excellent opportunity for experimenting with Perl code, *non*?), all you have to do is extract the source and run it:

```
tar -xzvf perlblaster-0.1.0.tar.gz
cd perlblaster-0.1.0
./perlblaster &
```

Use the “4” and the “6” on your numeric keypad to move side to side and the “5” to fire. *Bonne chance*! The fate of our world depends on your skill and lightning reflexes, *non*? For a peek at *Perl Defense Blaster* in action, have a look at Figure 4.



```
V

I

I

I

A
score: 0    EnemySpeed: 10  Lives: 1  (q = quit)
```

Figure 4. *Perl Defense Blaster*

Et bien, mes amis, the time has come once again for us to raise our glasses one final time. François will happily refill your glasses once more before you leave us. I hope that you have enjoyed today's menu and that you will join us again next time. Thank you once again for coming. Until next time, please join us here at *Chez Marcel*. Your table will be waiting.

A votre santé! Bon appétit!

Marcel Gagné lives in Mississauga, Ontario. In real life, he is president of Salmar Consulting, Inc., a systems integration and network consulting firm. He is also a pilot, writes science fiction and fantasy and is coeditor of TransVersions, a science fiction, fantasy and horror anthology. He loves Linux and all flavors of UNIX and will even admit it in public. He is the author of *Linux System Administration: A User's Guide*, available this fall from Addison-Wesley. He can be reached via e-mail at mggagne@salmar.com. You can discover lots of other things (including great Wine links) from his web site at http://www.salmar.com/marcel/.

Archive Index Issue Table of Contents

Advanced search

swatch: Automated Log Monitoring for the Vigilant but Lazy

Mick Bauer

Issue #88, August 2001

Sifting through logs every day, looking for trouble, could leave little time for anything else—swatch helps.

Previously the Paranoid Penguin has pondered a plethora of powerful programs pursuant to protecting people's PCs from pernicious punks. [The right to excessive alliteration revocable at any time—Ed.] One important feature these tools share is logging; just as important as keeping system crackers out is knowing when they've tried to get in. But who's got the time or attention span to sift through scads of mostly innocuous log files on every system they administer, every single day?

swatch (the “Simple WATCHer”) does. **swatch**, written 100% in Perl, monitors logs as they're being written to and takes action when it finds something you've told it to look for. This simple, flexible and useful tool is a must-have for any healthily fearful system administrator.

Installing swatch

There are two ways to install swatch. First, of course, is via whatever binary package of swatch, if any, your Linux distribution of choice provides. The current version of Mandrake has an RPM package of swatch, but none of the other more popular distributions (i.e., Red Hat, SuSE, Slackware or Debian) appear to include it.

This is just as well, though, because the second way to install swatch is quite interesting. **swatch**'s source distribution, available from www.stanford.edu/~atkins/swatch, includes a sophisticated script called Makefile.PL. The script automatically checks for all necessary Perl modules and uses Perl 5's CPAN functionality to download and install any needed modules; it then generates a Makefile that can be used to build swatch.

After you've installed the required modules, either automatically from swatch's Makefile.PL script or manually (and then running perl Makefile.PL), Makefile.PL should return the following:

```
[root@barrelofun swatch-3.0.1]# perl Makefile.PL
Checking for Time::HiRes 1.12 ... ok
Checking for Date::Calc ... ok
Checking for Date::Format ... ok
Checking for File::Tail ... ok
Checking if your kit is complete...
Looks good
Writing Makefile for swatch
[root@barrelofun swatch-3.0.1]#
```

Once Makefile.PL has successfully created a Makefile for swatch, you can execute the following commands to build and install it:

```
make
make test
make install
make realclean
```

The **make test** command is optional but useful; it ensures that swatch can properly use the Perl modules we took the trouble to install.

swatch Configuration in Brief

Since the whole point of swatch is to simplify our lives, configuring swatch itself is, well, simple. **swatch** is controlled by a single file, default $HOME/.swatchrc. This file contains text patterns in the form of regular expressions you wish swatch to watch for. Each regular expression is followed by the action(s) you wish swatch to take whenever it encounters that text.

For example, suppose you've got a web server, and you want to be alerted any time someone attempts a buffer-overflow attack by requesting an extremely long filename. By trying this yourself against the web server while tailing its /var/apache/error.log, you know that Apache will log an entry that includes the string “File name too long”. Suppose further that you want to be e-mailed every time this happens. Here's what you'd need to have in your .swatchrc file:

```
watchfor /File name too long/
    mail addresses=mick\@visi.com,
    subject=BufferOverflow_attempt
```

As you can see, the entry begins with a “watchfor” statement, followed by a regular expression. If you aren't proficient in the use of regular expressions yet (you are *planning* to learn regular expressions, aren't you?), don't worry: this can be as simple as a snippet of the text you want swatch to look for, spelled out verbatim between two slashes.

swatch will perform your choice of a number of actions when it matches your regular expression. In this example we've told swatch to send e-mail to mick\@visi.com, with a subject of BufferOverflow_attempt. Note the backslash before the @ sign; without it, Perl will interpret the @ sign as a special character. Note also that if you want spaces in your subject line, each space also needs to be escaped with a backslash, e.g., **subject=Buffer\ Overflow\ attempt**. Actions besides sending e-mail include those seen in Table 1.

Table 1. Some Actions swatch Can Take

For more details on configuring these and the other actions swatch supports, see the swatch(1) man page.

Let's take our example a step further. Suppose, in addition to being e-mailed about buffer-overflow attempts, you want to know whenever someone hits a certain web page, but only if you're logged on to a console at the time. In the same .swatchrc file, you'd add something like this:

```
watchfor /wuzza.html/
      echo=red bell 2
```

The event will then cause a beep and print to the console.

It's important to note you will only see these messages and hear these beeps if you are logged on the console in the same shell session from which you launched swatch. If you log out to go get a sandwich, when you return and log back in, you will no longer see messages generated by the swatch processes launched in your old session, even though those processes will still be running.

When in doubt add either a “mail” action or some other non-console-specific action, e.g., an “exec” action that triggers a script that pages you. Unless, that is, the pattern in question isn't critical.

Alert readers have no doubt noticed that the scenario in the previous example will work only for Apache installations in which both errors and access messages are logged to the same file. We haven't associated different expressions with different watched files, nor can we do so. But what if you want swatch to watch more than one log file?

No problem. While each .swatchrc file may describe only one watched file, there's nothing to stop you from running multiple instances of swatch, each with its own .swatchrc file. In other words, .swatchrc is the default but not the required name for swatch configurations.

To split the two examples into two files, therefore, you'd put the lines in the previous simple .swatchrc entry into a file called, say, .swatchrc.hterror, and the lines in the previous watchfor entry into a file called .swatchrc.htaccess.

Advanced swatch Configuration

So far we've only considered actions we want to be triggered every time a given pattern is matched. There are several ways we can control swatch's behavior with greater granularity, however.

The first and most obvious way is to take advantage of the fact that search patterns take the form of regular expressions. Regular expressions, which really constitute a text-formatting language of their own, are incredibly powerful and responsible for a good deal of the magic that Perl, **sed**, **vi** and many other UNIX utilities can do.

It behooves you to know at least a couple of “regex” tricks, which I'll describe here. Trick number one is called alternation, and it adds a “logical or” to your regular expression in the form of a “|” sign. Consider this regular expression:

```
/reject|failed/
```

This expression will match any line containing either the word “reject” or the word “failed”. Use alternation when you want swatch to take the same action for more than one pattern.

Trick number two is the Perl-specific regular expression modifier “case-insensitive”, also known as “slash-i” since it always follows a regular expression's trailing slash. The regular expression **/reject/i** matches any line containing the word “reject”, whether it's spelled “Reject”, “REJECT”, “rEjEcT”, etc. Granted, this isn't nearly as useful as alternation, and in the interest of full disclosure, I'm compelled to mention that slash-i is one of the more CPU-intensive Perl modifiers. However, if despite your best efforts at log-tailing, self-attacking, etc., you aren't 100% sure how a worrisome attack might look in a log file, slash-i helps you make a reasonable guess.

If you wish to become a regular expression archimage, I recommend the book *Mastering Regular Expressions* by Jeffrey E. F. Friedl. See Resources for details.

Another way to control swatch to a greater degree is to specify what time of day a given action may be performed. You can do this by sticking a “when=” option after any action. For example, below I've got a .swatchrc entry for a medium-importance event I want to know about via console messages during weekdays, but I'll need e-mail messages to know about it during the weekend. To do this I set the when option:

```
  /file system full/
        echo=red
        mail addresses=mick\@visi.com,
        subject=Volume_Full,when=7-1:1-24
```

The syntax of the when= option is **when=*`range_of_days`*:*`range_of_hours`***. Thus, we see that any time the message "file system full" is logged, swatch will echo the log entry to the console in red ink. It will also send e-mail, but only if it's Saturday ("7") or Sunday ("1").

Running swatch

swatch expects .swatchrc to live in the home directory of the user who invokes swatch. **swatch** also keeps its temporary files there by default (each time it's invoked it creates and runs a script called a "watcher process", whose name ends with a dot followed by the PID of the swatch process that created it).

The -c *path_to_configfile* and --script-dir=*path* flags let you specify alternate locations for swatch's configuration and script files, respectively. Never keep either in a world-writable directory, however. In fact, only these files' owners should even be able to read them.

For example, to invoke swatch so it reads my custom configuration file in /var/log and also uses that directory for its watcher process script, I'd use this command:

```
  swatch -c /var/log/.swatchrc.access --script-dir=/var/log &
```

I also need to tell swatch which file to tail, and for that I need the -t *filename* flag. If I wanted to use the above command to have swatch monitor /var/log/apache/access_log, it would look like this:

```
  swatch -c /var/log/.swatchrc.access --script-dir=/var/log \
       -t /var/log/apache/access_log &
```

swatch generally doesn't clean up after itself very well; it tends to leave watcher-process scripts behind. Keep an eye out for and periodically delete these in your home directory or in the script directories you tend to specify with --script-dir.

Again, if you want swatch to monitor multiple files, you'll need to run swatch multiple times, with at least a different tailing-target (-t value) specified each time and probably a different configuration file for each as well.

Fine-Tuning swatch (in Both Directions!)

Once swatch is configured and running, we must turn our attention to the Goldilocks Goal: we want swatch to be running neither too hot (alerting us about routine or trivial events) nor too cold (never alerting us about anything). But what constitutes just right? There are as many different answers to this question as there are uses for UNIX.

Anyhow, you don't need me to tell you what constitutes nuisance-level reporting: if it happens you'll know it. You may even experience a scare or two in responding to events that set off alarms initially but turn out to be harmless nonetheless. Read the manual, tweak .swatch.rc and stay the course.

The other scenario, in which too little is watched for, is much harder to address, especially for the beginning system administrator. By definition, anomalous events don't happen too frequently, so how do you anticipate how they'll manifest themselves in the logs? My first bit of advice is to get in the habit of browsing your system logs often enough to get a feel for what the routine operation of your systems looks like.

Better still, **tail** the logs in real time. If you enter the command

```
tail -f /var/log/messages
```

the last 50 lines of the system log will be printed, plus all subsequent lines, as they're generated, until you kill tail with a Ctrl-C. This works for any file, even a log file that changes rapidly.

Another good thing you can do is to “beat up on” your system in one virtual console or xterm while tailing various log files in another. The tools we explored last month and the month before, **Nessus** and **nmap**, respectively, are perfect for this.

By now you may be thinking, “Hey, I thought the whole reason I installed swatch was so I wouldn't have to watch log files manually!” Nope. **swatch** minimizes, but does not eliminate, the need for us to parse log files.

Were you able to quit using your arithmetic skills after you got your first pocket calculator? No. For that matter, can you use a calculator in the first place unless you already know how to add, multiply, etc.? Definitely not. Same goes for log file parsing: you can't tell swatch to look for things you can't identify yourself, no more than you can ask for directions to a town whose name you've forgotten.

Why You Shouldn't Configure swatch Once and Forget about It

In the same vein, I urge you to not be complacent about swatch silence. If swatch's actions don't fire very often, it could be that your system isn't getting probed or misused often, but it's at least as likely that swatch isn't casting its net widely enough. Continue to scan through your logs manually from time to time to see if you're missing anything, and continue to tweak .swatchrc.

And don't forget to reconsider periodically the auditing/logging configurations of the dæmons that generate log messages in the first place. **swatch** won't catch events that aren't logged at all. Refer to the syslogd(8) man page for general instructions on managing your syslog dæmon and the man pages of the various things that log to syslog for specific instructions on changing the way they log events.

Resources

Should We Let Perl Download and Install Its Own Modules?

Mick Bauer (mick@visi.com) is a network security consultant in the Twin Cities area. He's been a Linux devotee since 1995 and an OpenBSD zealot since 1997, taking particular pleasure in getting these cutting-edge operating systems to run on obsolete junk. Mick welcomes questions, comments and greetings.

Advanced search

DreamWorks Feature Linux and Animation

Robin Rowe

Issue #88, August 2001

Robin explains the process and benefits of making animated movies with Linux.

With more than 200 Linux desktops and 400 Linux servers, DreamWorks SKG is not only a leading producer of animated motion pictures but a major user of Linux as well. DreamWorks animation utilizes three production pipelines: Aardman in Bristol, England (*Chicken Run*), PDI/DreamWorks in Palo Alto, California (*Shrek*, *Antz*) and DreamWorks traditional animation in Glendale, California (*The Road to El Dorado*, *The Prince of Egypt*). Each production unit has its own animation techniques. Aardman is recognized for claymation, PDI/DreamWorks for CG (computer graphics) and DreamWorks in Glendale for traditional techniques.

We visited the Glendale studio where the major animated motion picture, *Spirit, Stallion of the Cimarron* is in production. *Spirit*, the story of a wild mustang's adventures in the untamed American West, is due for domestic release in the summer of 2002.

Some wonder how Linux will dislodge Windows on the desktop because leading desktop applications such as Microsoft Office (Word, Excel and Access) aren't there. But, if you are a motion picture animator most of your everyday tools are already available on Linux, and the number being ported or even produced specifically for Linux is increasing at a remarkable rate.

DreamWorks has followed three paths to Linux: new development, porting and third-party-vendor porting. Head of technology Ed Leonard says, “To dramatically reduce costs was one of the big motivating factors in moving animators to Linux. But, it is our animators' productivity that really counts. Telling the story well, not the underlying technology, is what matters to us.” Using Linux saves time for the animators because Linux PC performance is so much faster than the five-year-old computers being replaced, even though those SGI IRIX workstations were awesome machines. Leonard adds, “Microsoft software continues to play a key role in our overall business, but Linux is particularly well suited to animation production pipelines."

An animator's desktop is not the same machine that an executive or secretary would have. The animator needs a high-performance workstation with a dual-head, high-performance graphics system and specialized software for motion picture production. To typical computer users, the animator's software tools may be unfamiliar. Let's walk through the DreamWorks production cycle and see how Linux is used (see Table 1).

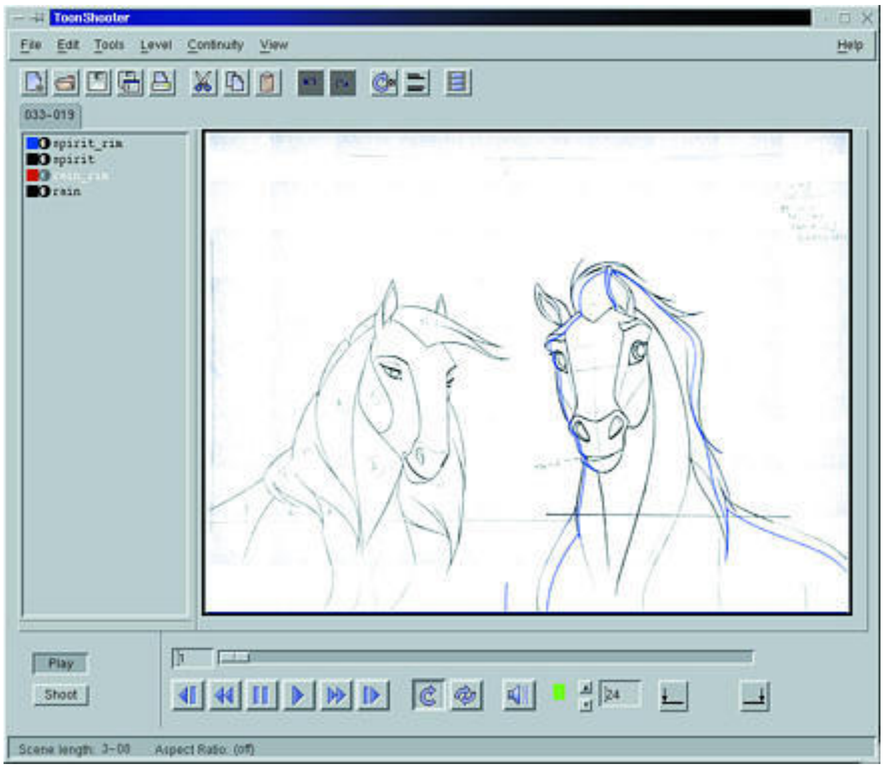
Table 1. Animation Types and Methods

An animated feature film starts with a story idea. Production takes two years. In the preproduction phase called visual development (vizdev) many different forms of art, including oil paintings, are created to capture the essence and look of the film. Some works of art go beyond the general level of detail and realism that will be in the finished film.

Similar in appearance to a comic book, a storyboard is sketched by hand on paper to help visualize key production shots. Then, an animatic movie is created by artists using custom plugins in Alias|Wavefront's 3-D animation package Maya. Although lacking the quality of finished animation, the animatic shows the context for the scene, the camera view and helps with character development. Using the animatic, the production staff can visualize parts of the film in motion before it enters production. None of this preproduction content will make it into the film. It is used as a guide for producing the real film later.

Scene planning determines the characters, backgrounds and effects to be built. Animation, backgrounds and effects are separate departments. The pieces will be brought together later using compositing software.

For character animation, a scan of a paper sketch is done using ToonShooter. Production software lead Derek Chan explains, "ToonShooter is an internal tool we wrote for Linux. It captures low resolution 640 × 480 line art that the artists use to time the film." Created more than a year ago, this Linux capture stand software is deployed in three animation departments. Chan says, "Demand was keen for this Linux software, and we delivered it ahead of schedule. DreamWorks has 60 units in production now."



ToonShooter Screenshot

ToonSketch lets artists do quick copystand-style work without paper or scanner. Senior software engineer Nhi Casey demonstrates, “We sketch on the Wacom tablet like we would on a piece of paper. It's fast and has unlimited undo.” Head of software Jim Mainard adds, “You can do ghosting to see frames before and after. That's easier for animators to work with than paper that bunches up.” Utilizing ghosting, the animator traces, or is guided by, adjacent frames.

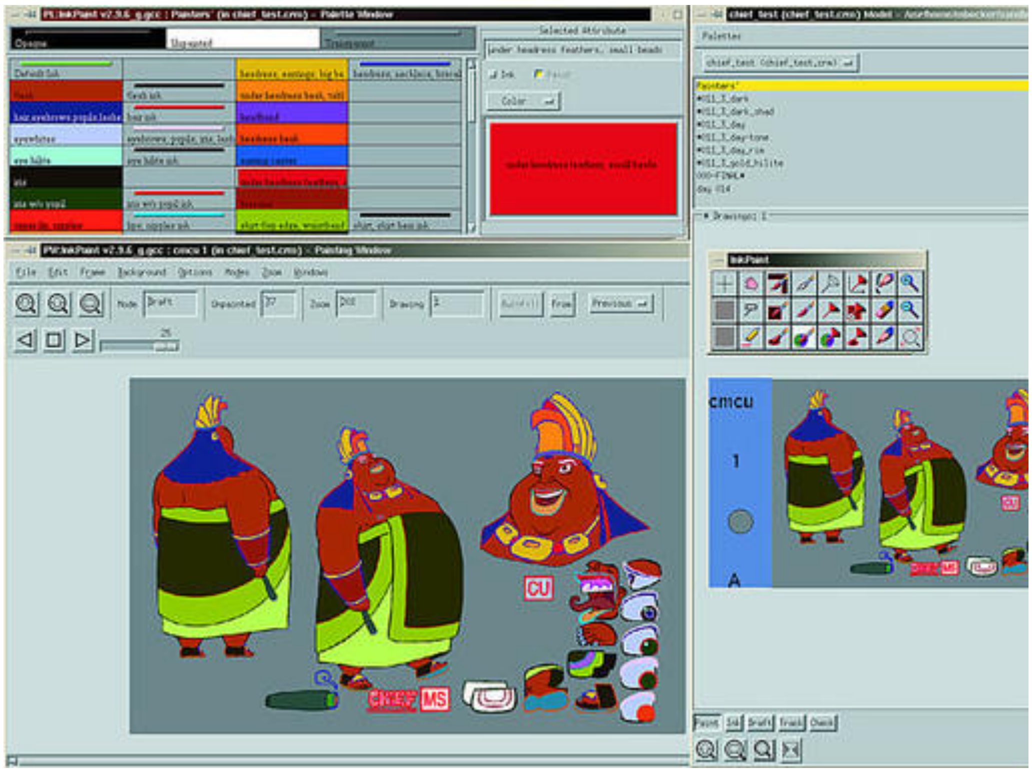
Nhi Casey, Senior Software Engineer

To maintain the look of traditional cel animation, DreamWorks will hand-draw the foreground characters. The scanning department uses large auto-feed scanners to capture each hand-drawn frame in high-resolution (2k pixels wide) format. For *Prince of Egypt* they handled more than a million pieces of paper.

ScanLevel, an IRIX-based application, saves the image into a proprietary compressed raster format. Mainard says a switch to a Linux version of this software is anticipated later this year once a driver for the scanner is finished. ProcessLevels, a suite of proprietary Linux-based tools, performs contrast enhancement to take out the color of the paper, that is, drive the background to white and darken line art. It also automatically takes out skew and optical distortion.

Digital InkAndPaint takes the scanned line drawings and adds color. Star painter Tina Staples demonstrates her skill with InkAndPaint, an internal Linux application. It works a bit like the GIMP or Photoshop but is specialized for all sorts of bucket fill image operations. Leonard jokes that it's a bit like paint by numbers on steroids. Tina is so fast at painting a frame we had to ask her to slow way down so we could follow what she was doing. Tina exclaims, “We go so fast now! Using Linux has doubled my speed, and we're meeting our quotas each week.”

Tina Staples, Star Painter


InkAndPaint Screenshot

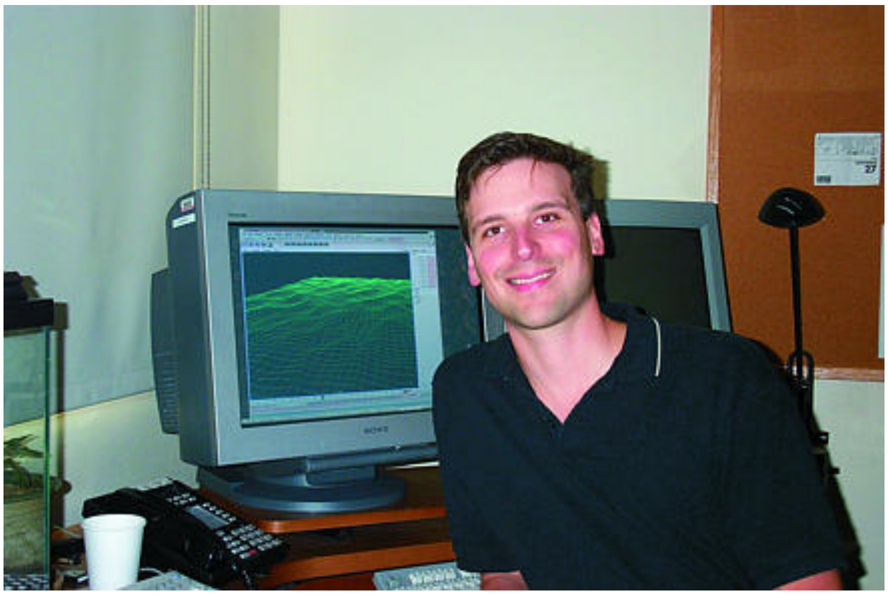
InkAndPaint does autofill by region, has touchup tools and automated tweens. “Automation breaks down as scenes progress”, says Mainard. “A good operator is needed to color areas where several lines intersect.” If that isn't done right you see a crawly effect from the color changing at the intersection points frame to frame. “Color where lines come together is hard to automate, but we hope to do more development with autofill”, says Mainard. “Tina is so fast with InkAndPaint it almost doesn't matter.”

InkAndPaint replaced dual-head Octane workstations with Linux PCs. The left monitor is used for painting, while the right monitor shows the markup guide. Leonard points out that they overcame a lot of Linux graphics issues early, such as overlay planes, to get snappy screen performance. That (and the work on the Wacom tablet drivers) has been given back to the Open Source community.

Most backgrounds are drawn 2-D by hand using either traditional tools, such as oil paint, or digital tools such as Macintosh Photoshop. Background drawings can be quite large. Mainard says, “In *Prince of Egypt* some were six to eight feet wide.” ScanBackground, an IRIX-based program, digitizes those through a 36“ Tangent scanner. Scanning algorithms stitch the large backgrounds back together.

Backgrounds that need to exhibit perspective changes are created in 3-D using Maya. Colors may be projection textures (texture maps) or paint. For 3-D paint DreamWorks was using Studio Paint, another Alias|Wavefront product. But, they are switching to **wpaint**, a proprietary Linux application developed at PDI/DreamWorks.

The special effects department creates organic effects such as dust, fire, fluids, smoke and particles. Senior software engineer for advanced R&D future films Galen Gornowicz has been developing technology for an unannounced animated motion picture for a year. “I've been working on developing an ocean scene that renders in real time. There are proprietary systems that can do this, but we're running a plugin on top of Maya”, says Gornowicz. The productivity gain in Linux is a big plus. “With the old systems we would go to lunch when rendering something like this. And, the water models were executed at the command line, not interactive like this”, he points out.


Galen Gornowicz, Senior Software Engineer for Advanced R&D Future Films

Gornowicz says, “We use Maya for effects plus a whole host of plugins such as Calypso water developed for Linux.” DreamWorks found that the water in Calypso was too photo-realistic, too much like the movie *The Perfect Storm*. Gornowicz is working to modify the water's look to match the movie's vizdev images. Developing a Maya plugin involves writing a MEL script for simple effects or using C/C++ for a more complex or high-performance plugin. When using a compiled plugin, an interpreted MEL script (sort of like JavaScript) is used to create a menu choice and hook the plugin into Maya. Any function may be called by the MEL script. A plugin has no main().

Rendering, producing an image of a computer-generated object, may be done locally on the user's workstation or on the DreamWorks renderfarm. The renderfarm accepts jobs in queue or as a priority render on three refrigerator-sized SGI Origin 2000 servers (with 8-16 CPUs).

SGI Renderfarm

Linux Renderfarm

At PDI/DreamWorks the renderfarm used for *Shrek* has a 1,000+ processors, 80% Linux and 20% IRIX. Even before acquiring PDI, DreamWorks in Glendale was rendering using Linux. They have older Origin 200 servers running IRIX but are switching to much cheaper Pentium computers. The Glendale renderfarm is half the size of the one at PDI/DreamWorks because traditional animation is less CPU-intensive than CG. VA Linux provided Glendale's first Linux render computers that proved Intel PCs capable. DreamWorks is building a new render tower made up of dual 1GHz P3 2GB RAM computers housed in a 1RU (1.75") package stacked 41 units high. This refrigerator-sized unit is replacing computers consuming 40-50 feet of data center rack space.

The compositing department puts all the pieces together: the characters, the backgrounds and the special effects. They also do matting of foreground objects, for example where a character is standing partly behind a rock. Mainard explains, "DreamWorks uses a heavily modified version of Director, which was originally developed by Cambridge Animation. This software is similar to Shake, a popular commercial compositing software package, but has been modified for use with X-sheets." An X-sheet is a frame-timing list and how animators traditionally prefer to work. All of DreamWorks tools are modified to support X-sheets. Director also has special support for rendering InkAndPaint levels at the right resolution with correct anti-aliasing.

Artists mostly work at quarter resolution with 1k wide images. For feature release the final rendering is done at 2k, or for IMAX release at 3k or 4k. The final render outputs the frames as TIFF files to the output server that passes them to Cineon Lightening film recorder serial number 1. The laser film recorder prints at a rate of about 3fps for 16-bit per channel 2K frames. Mainard says, "Once a sequence is complete we film out that with a series of Perl scripts that prepare data for the film recorder. Processed film comes back as color dailies, typically every day."

"Linux works great for games, but porting from rich graphics APIs of SGI takes time", says Leonard. "HP, IBM and SGI have responded with good Linux solutions for us." Leonard says Dell also "gets it" and will be expanding more into Linux. Asked why we saw no AMD CPUs Leonard says, "Linux on Intel provides a strong and consistent platform for the high-end workstation market across several vendors. That's why we're not pursuing things like Linux on Alpha, FreeBSD or proprietary UNIX solutions." Leonard adds, "HP has great Linux support. We were getting two patches a day from these guys." Instead of maintenance contracts, DreamWorks now buys inexpensive spare PCs. Leonard says that "disposable rendering" will enable DreamWorks to replace desktops and the renderfarm every two years instead of five.

DreamWorks uses both internally developed programs tailored to the needs of their animation production and commercially available animation software. DreamWorks has three million lines of code internally, most of which was originally developed for the SGI IRIX operating system. Linux is much more similar to IRIX than Windows, and consequently, is much easier for porting such large masses of code. Linux Maya (to be reviewed here next month) is a commercial software package that plays a large role in their production.

Each DreamWorks animation pipeline (Aardman, PDI/DreamWorks, DreamWorks traditional) has the capacity to produce a major animated motion picture every 18 to 24 months. Thanks to the economics of Linux, a fourth CG production pipeline is being constructed at DreamWorks in Glendale. Entering production early next year, the new pipeline will be based entirely on Linux using Intel IA-64 and Pentium4 processors. We can look forward to more films like *Shrek* and *Spirit* being powered by Linux, and to more amazing animation from DreamWorks SKG.

Robin Rowe 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*. His software designs include a client/server video editing system in use at a Manhattan 24-hour broadcast television news station, Time Warner New York One and associated web site http://www.ny1.com/. You can reach him at robin.rowe@movieeditor.com.

Linley on Linux

Linley Gwennap

Issue #88, August 2001

Extra, extra, read all about it.

Everyone likes their privacy, but how much are they willing to pay for it? Today, there is little privacy on the Internet, but new chips are emerging that will allow users to protect their information for little or no extra cost.

Two security standards are predominant on the Internet: SSL and IPSec. The former is built into most browsers, providing secure web transactions. IPSec creates virtual private networks (VPNs) that enable users to access remote databases securely.

Both standards use encryption to protect sensitive data from sniffers, snoopers and intruders. Until last year, the US government blocked the export of strong encryption technology, but the government has since relented. Today, the biggest barrier to the widespread use of encryption is the massive computation required to encode and decode messages. If encryption were not computationally difficult, codes could be easily broken. Fortunately, most modern PC processors have enough horsepower to encrypt messages on a broadband internet connection, or even a slow Ethernet connection. The problem is the server, which must handle messages from a large number of clients at once. These servers are typically operating at Fast Ethernet (100Mbps) speeds or higher.

To break this bottleneck, companies have turned to dedicated encryption hardware, such as VPN boxes or SSL cards. These units typically use specialized security chips that perform encryption calculations much more quickly than standard CPUs.

Today's security chips, however, are not very fast, so a high-end VPN or SSL system may combine several of these expensive devices, along with their support hardware, to achieve top speed. As a result, a high-end VPN box sells for hundreds of thousands of dollars.

But help is on the way. Greater interest in security has spurred new companies to enter the market for security chips; we now count ten companies in this market, with more on the way. Most of the chips last year came from Hifn, a spin-off of the software-compression vendor Stac. But chip giants Intel, Motorola and Philips are jumping into this market as well. Competition encourages innovation. Not surprisingly, much of this innovation is coming from small companies such as Chrysalis-ITS, SafeNet and Securealink as well as startups such as Corrent, NetOctave and BlueSteel, now the security division of Broadcom. As a result of this competition, by the middle of next year we will see security chips operating at 10Gbps, able to serve the fattest pipes in the network infrastructure.

This speed is more than 100 times faster than that of the best security chip available at the beginning of last year, a phenomenal increase. Compared with the standard pace of Moore's Law, we have compressed ten years of progress into just over two years. It will take some time for the new, faster chips to become common in systems, but by next year, we should see systems with the performance of today's high-end VPN boxes selling for just a few thousand dollars.

In fact, once encryption gets this cheap, it won't even be in separate boxes; these superfast chips can be included on every line card in every networking system. ISPs will offer a security service to their users at a minimal cost, perhaps as little as $1 per month. At this price, most people will be able to secure their daily e-mail, web surfing and other on-line activities. Some analysts predict that, by 2004, as much as half of the traffic on the Internet will be encrypted.

In many cases, this will occur with little or no impact on applications. As part of the IP standard, IPSec works at Layer 3 in the network stack, below the application layer and below even TCP. Once the operating system establishes a secure link between two sites (for example, your PC and a corporate server), all traffic between these two sites is encrypted and decrypted without any impact on the application.

SSL is a higher-layer protocol that must be directly managed by the application. But since it is already built into the browser, any services accessed through the browser can take advantage of it. The onus is on webmasters to implement more of their site on secure servers. As the cost of security falls, entire sites can be secured.

The trick to using IPSec is to make sure it is included in the operating system. Linux users can take advantage of FreeS/WAN, a well-regarded open-source implementation of IPSec. Most Linux distributions, however, do not include FreeS/WAN, although this may change as encryption becomes more popular. In contrast, IPSec is a standard feature in Windows 2000. As the cost of security chips falls, the Linux community needs to be ready. Developers, distributors and end users should make sure their systems can take advantage of inexpensive encryption hardware.

Founder and principal analyst of The Linley Group, **Linley Gwennap** (linleyg@linleygroup.com) has recently completed a new report, “A Guide to Security Processors” (http://www.linleygroup.com/npu/).

Archive Index Issue Table of Contents

Advanced search

Focus on Software

David Bandel

Issue #88, August 2001

Everywhere I look, I see (and install) more and more Linux desktops.

Well, it looks like all the big boys (Red Hat, Caldera, Mandrake, etc.) have thrown in their hats and said, “Linux is not ready for the desktop.” At least that's the message. Hey, Microsoft, you won without a fight! I don't know what the CEOs of these companies are smoking, but it must be very strong stuff. Everywhere I look, I see (and install) more and more Linux desktops. And you know what? The folks for whom I do the installs don't understand why they couldn't have a desktop before that was this robust, this good, this inexpensive. But these now publicly held companies I mentioned above are talking about 1) raising the price and 2) charging a per-CPU license. As far as they're concerned, the free ride is over. Time to pay the Linux distributors. If I didn't find Debian's GNU politics so annoying, I'd start using it. Maybe it's time to start my own distro? Or at least one for my clients? At least that would eliminate the unpleasant surprises that accompany each new release, and I could decide what's best for my clients rather than using a distribution whose creators seem increasingly out of touch with what's happening with their VARs and customers.

webCDwriter:

http://www.uni-bielefeld.de/~jhaeger/webCDwriter/

Now this is nice (and convenient). **webCDwriter** lets you surf over to your web server/CD burner and burn a CD of files on your local machine across the network. This is truly convenient, and any user can do it. In fact, its simplicity and ease of use may be its biggest drawback. You may find your CD burner is suddenly running overtime burning CDs from all over your network. No more excuses for not having a burned copy of important files and directories because the CD burner is on a remote system, and it's inconvenient transferring the files. Guess I'll have to invent yet another excuse. Requires: Java, cdrecord, mkisofs, web server, web browser w/ Java support.

Heroes:

http://heroes.sourceforge.net/

This particular game is a cross between *Snakes* and *Nibbles*, based on the old DOS *Heroes* game. The graphics are quite good, and game play is fast. The complete *Heroes* code includes a large number of soundtracks, more levels than most normal gamers can play in a night and several game modes. Requires: libm, libmikmod, libpthread, libdl, libSDL, libartsc, libX11, libXext, glibc.

Port Scan Attack Detector:

http://www.cipherdyne.com/psad/

This Perl utility takes advantage of **iptables** or **ipchains** logging and uses the logged information to determine whether the system is under attack. The parameters are highly configurable. **psad** can send an e-mail to the administrator when it sees a scan. The e-mail will include custom whois information. This is a fairly simple but effective tool (along the lines of **courtney**), but it doesn't put your Ethernet card in promiscuous mode and will watch only those ports you have logging on. Requires: Perl, Perl modules: Socket, Getopt::Long, File::Stat, and Data::Dumper.

iptrap:

http://www.jedi.claranet.fr/

If you are very paranoid or just under attack often (as my servers are), you can block offending IPs quickly and easily with this tool. I tested it on my local system that does not run mail. Telling it to block any host hitting port 25, I Telnet to another system, then Telnet back to the local system on port 25. Instantaneously, I had a rule inserted in the input chain. I had told it to REJECT rather than use the default DROP, and the resulting iptables rule showed a reject with port-unreachable. Nice. This will be put to good use. Can also run external scripts that e-mail you the output from a `dig -x <offending IP>`. Requires: glibc and iptables (or ipchains).

Password Expiration Agent:

download only: http://frida.fri.utc.sk/~behan/devel/passwd_exp/

This script, run daily, will look through your /etc/shadow file and send an e-mail to any user whose account is about to expire or be disabled. Personally, as an administrator, I like to get the list and send out notifications where appropriate myself. But if you have a lot of accounts or just don't want to bother, this is the way to do it. Requires: Perl, Perl modules provided by author (RcRecord.pm, spent.pm).

Linux Terminal Server Project:

www.ltsp.org/index.php

I don't know if any of you have had the need to set up a diskless workstation on a system served from another Linux system via TFTP. I remember the ordeal well. The HOWTO was woefully inadequate in many parts. So, I decided to repeat the experiment using **ltsp**. In just about 30 minutes, it was up and running. Okay, so I had a head start having done it once before. Drawback: unless you're running Red Hat, Debian or Caldera, be prepared to do it by hand or hack the install scripts. (That's what I had to do on my Caldera system because it didn't understand Caldera 3.1, only 2.4.) Requires: installed and running XDM, KDM or similar, DHCP server, TFTP. Workstation capable of a network boot or floppy boot.

gTaxEstimator:

http://www.gtx.seul.org/

Tax time is past, but it's coming around again fast. And while gTaxEstimator isn't yet ready for prime time, it could be by next tax season. Personally, I'm hoping for support for Schedule 2555 soon. The interface is simple and clean. This is probably the most promising piece of software I've seen in a while. Let's face it, if you live in the US (or even if you don't but are a US citizen), there's no escaping the Internal Revenue Service. Requires: libgtk, libgdk, libgmodule, libglib, libdl, libXext, libX11, libm, libz, glibc.

Make CD-ROM Recovery Utility:

http://mkcdrec.ota.be/

The **mkCDrec** utilities allow you to do several different things. They allow you to make an el-torito bootable CD for system rescue. They also allow you to back up your entire system to multiple CDs. You'll also need the mkCDrec utilities if you want to do system restores. You can backup systems that don't have burners, either by creating the iso image(s) and transferring them for later burning or using NFS to write the ISO image(s) to the system with the burner and burn later. Either way, this utility is convenient. Requires: Running system w/ mkisofs gzip, access to a system with cdrecord, mkisofs, gzip.

Until next month.

David A. Bandel (dbandel@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

Here Come the Devices!

Rick Lehrbaum

Issue #88, August 2001

As predicted, 2001 shapes up as “The Year of Linux in Devices”.

Last year, I said 2001 would be the year embedded Linux starts popping up in all kinds of “smart devices”, consumer-oriented and otherwise. It looks as though my prediction is materializing on schedule, despite the popping of the dot-com/Linux bubble and the ensuing economic downturn.

In this month's column, we take a take a brief look at three recently announced consumer products that have Linux embedded inside: an Internet/TV from Sylvania, a personal server from Memora and a Mobile Multimedia Communicator from Galleo.

The Sylvania Internet/TV

This summer, Sylvania Computer Products is expected to introduce a new 27-inch digital TV that combines the functions of a TV with those of an internet appliance. The device, which owes its internal intelligence to a single-chip PC running an embedded Linux operating system, marks a key milestone in the television industry by being one of the first consumer TVs to include a built-in internet appliance. Call it the dawn of the Internet/TV.

Sylvania's new SPC2700iHD marks the dawn of the Internet/TV.

Sylvania's new Internet/TV, designated to be Model SPC2700iHD, includes:

- A high-resolution HDTV-ready monitor with a built-in conventional TV tuner.
- An embedded internet appliance computer that supports dial-up (56K) and broadband (Ethernet) internet access.
- Linux-based operating system with an easy-to-use graphical user interface that unifies TV watching and internet viewing.
- A combination remote/keyboard/mouse to simplify system operation.
- Internet-based control and database services.

The Internet/TV receives broadcast and cable TV signals via its built-in TV tuner, but it is also usable as a high-resolution display for external video sources, including VCRs, DVD players, DSS, cable boxes and computers (up to 800 × 600 SVGA resolution).

The system's internal CPU is a 266MHz National Semiconductor Geode, equipped with 64MB of RAM and a 16MB nonvolatile Flash disk. Its long list of input/output ports includes an IR receiver for interfacing with its remote/keyboard/mouse device, an IR blaster for sending data to external devices, two USB ports, a 100Mb Ethernet port, a 56K phone line modem, composite video inputs/outputs with L/R audio, S-video input with L/R audio and high-resolution RGB video input for either HDTV or computer-generated SVGA input.

The real power of the Sylvania Internet/TV comes from a combination of internal software and on-line services created by Ch.1, Inc. (http://www.ch1.com/). Ch.1 licensed hardware and software technology for the device to Sylvania and also serves as an Internet/TV service provider. The unit's so-called Ch.1-enabled functions are used to watch TV, browse the Internet, create web and TV favorites, view a customized programming guide, access preprogrammed category portals, send e-mail, chat, shop and listen to MP3s. You can even watch a TV program while simultaneously interacting with its associated web site using the set's Picture-in-Portal technology, or you can direct your VCR (or HDR) to record through Ch.1's web-based control screen. On-screen buttons give you easy access to the Internet/TV's user manual, product or accessory upgrades, customer service and technical support.

Notably, unlike solutions based on adding an Internet access settop box to a conventional TV, the Sylvania Internet/TV lets you browse the Web with full 800 × 600 SVGA resolution. And if you want the comfort of typing with a full-size keyboard, you can easily add one as an optional accessory. So, this device may be the ticket to internet access for the roughly 50% of consumers that don't already own a PC. And best of all, they'll all be (unknowingly) Linux users!

The Servio Personal Server

New England startup Memora Corporation (http://www.memora.com/) hopes to establish a new product category, the “personal server”, with its Linux-based Servio, an easy-to-use appliance-like device that integrates a combination of services increasingly in demand in today's “well-connected” homes: gateway, firewall, wired/wireless network server, e-mail hosting and shared storage of multimedia and other files. Besides offering these capabilities within the home, the device also offers secure external access (via the Internet) to e-mail, web pages and designated files. According to Memora, the personal server functions as “a single point of presence for organizing, accessing, and sharing [users'] digital information when, where, and with whomever they choose”.

Memora's Personal Server lets users organize, access and share their digital information when, where and with whomever they choose.

The Servio Personal Server is essentially a small (and unique looking) Linux-based computer system, not too different from an ordinary desktop PC, that requires no keyboard or display to operate. Inside there's a 600MHz (or faster) Intel Celeron processor with 128MB of memory and a 30GB hard disk. External input/output connections consist of two fast Ethernet (10/100 Mb/sec) ports and two USB ports. One Ethernet port typically connects to a DSL modem, while the other goes to an in-home LAN. The USB ports let you connect the device to wireless LANs and other supported external interfaces.

The Servio's internal software consists of Linux, some other open-source programs (including the Apache web server, MySQL and the Exim mail server) and a set of Memora-written programs that control the configuration and operation of the device. The latter are browser-based applications that, among other things, integrate e-mail, database and web services.

The system comes preconfigured for plug-and-run operation by nontechnical users. You simply hook it up between a broadband connection (DSL or cable) and a local (wired or wireless) network inside your home. Next, you turn on the power and let the system's auto-configuration program discover your network's settings. Finally, using any PC browser, you enter your name and a public name for the device. If everything goes as expected, you're now ready to set up accounts for friends, associates and family members, and begin sharing photos, video clips, music, files and take advantage of the other services provided by the device.

The Galleo Mobile Multimedia Communicator

Galleo (http://www.galleo.com/) [see the July/August 2001 issue of our sister publication, *Embedded Linux Journal*, for further details on the Galleo] recently announced a Linux-based Mobile Multimedia Communicator. Like several other newly-announced handheld computers with wireless connectivity, the device combines the functions of a PDA, web appliance and cellular phone. Galleo's device comes with software that supports cellular phone communication, internet access, web browsing, PIM applications, multimedia (MP3 player, streaming video), games and personalized content, plus IPSec-compliant VPN network security.

Galleo's Mobile Multimedia Communicator combines the functions of a PDA, web appliance and cellular phone.

According to Galleo product manager Yovav Meydad, patent-pending cellular communication technology delivers an “always on, always connected” data communication capability that represents “a credible alternative to WAP” and “gives the end user the same [web browsing] experience [as on a desktop PC] while he is on the move”. “Fit-to-page” software with zoom/pan functions allows viewing standard web pages on the unit's landscape-mode quarter VGA display. Another aid to using unmodified on-line content and services is an included Java Virtual Machine.

A full-featured web browser supports HTTP 1.1, HTML 4.0, XML 1.0, JavaScript, SSL, TLS, RSA, VPN, Personal Java 1.2 JVM and display of PNG, GIF and JPEG images. MP3 player, media player and video-streaming functions are also included, as are a full PIM suite including functions for e-mail, calendar, tasks, contacts, notes and data synchronization.

The user interface supports pen and touch input using a stylus with software that provides handwriting recognition and an on-screen keyboard, plus a few hard keys. A built-in joystick is intended to support gaming and simplify web site navigation.

Inside there's a 206MHz Intel StrongARM SA-1110 system-on-chip processor equipped with 32MB system RAM and 16MB of nonvolatile Flash disk. Its display is a bright, full-color, 320 × 240 pixel TFT LCD. The unit provides input/output interfaces for RS-232 serial, USB, infrared (IrDA) and stereo audio (for headphones/MP3 connection). There is also a pair of expansion slots for additional memory and secure data cards. A built-in cellular telephone module supports dualband GSM, GPRS, TriCodec and fax/data transmission at 14.4 kbps. Galleo plans to support CDPD for the North American market but plans to introduce the device first in Europe, where GSM and GPRS are making their initial appearance.

Rick Lehrbaum (rick@linuxdevices.com) created the LinuxDevices.com “embedded Linux portal”, which is now part of the ZDNet Linux Resource Center. Rick has worked in the field of embedded systems since 1979. He cofounded Ampro Computers, founded the PC/104 Consortium and was instrumental in creating and launching the Embedded Linux Consortium.

Archive Index Issue Table of Contents

Advanced search

Copyright Questions

Lawrence Rosen

Issue #88, August 2001

Lawrence answers your questions regarding copyright law.

I'd like to create an open-source version of a proprietary program. Can I do this without getting sued by the owner of the proprietary program?

—Andre Durand, founder of Jabber.Com

For the sake of illustration only, I'm going to assume that you want to create an open-source version of a program such as Windows. Unfortunately, the owners of most proprietary programs (like the owner of Windows) do not allow you to reverse engineer their programs or to copy their code. That doesn't mean you can't exercise your own creativity to create your own winning version of a proprietary program, including your own improvements and new features. The simple answer to your question is yes, you can create an open-source version of a proprietary program, as long as you create your own version of the program without infringing the copyrights or patents of the proprietary program. The courts uniformly hold that a copyright is infringed when one intentionally, or even subconsciously, makes copies of a copyrighted work. However, if a person, without making any use of a prior copyrighted work, by his or her own independent labor produces something similar, there is no infringement. As Judge Learned Hand put it, to sustain an infringement suit, “more must appear than the mere similarity or even identity of the supposed infringement with the part in question” (Fred Fisher Inc. vs. Dillingham D.C.N.Y. 1924 298 F.145, 147). He continues:

> Two or more authors may write on the same subject, treat it similarly, and use the same common materials in like manner or for one purpose. Their productions may contain the same thoughts, sentiments, ideas; they may be identical. Such resemblance or identity is material only as showing whether there has been unlawful copying.

Copying is a matter to be proven by evidence. The law requires proof of either outright copying or, if the evidence of copying is unclear, proof that you had access to the original and that there are “substantial similarities” between your work and the original. Because of the risk that evidence can be interpreted in many ways, when you set out to implement your own independent version of a copyrighted proprietary program, you should make a clear record of precisely what you have done. Here are some simple, common-sense practices you should follow. When you examine the program you intend to re-implement, restrict yourself to an external review of the functions and processes that the other program performs. Remember that if your program turns out to be too similar to the proprietary program, the court may not believe that you didn't see and copy copyrighted code. The less you look at, the less likely it is that your code will be “substantially similar”. Don't copy the user documentation for the proprietary program, even if your new program is intended to operate identically. Such copies of documentation are infringing even if the software itself doesn't infringe. If you have had access to the source code or implementation details of the proprietary program, you may have to separate yourself from the implementation of the open-source version. One way big companies do this is to create a specification that describes what the program is to do, and give that specification to a separate group of engineers who have never seen the original proprietary program. Those engineers can implement according to the specification without copying the original work. To protect yourself, document this procedure thoroughly. One final caution: the copyright in the original proprietary program does not extend to the “ideas” implemented by the program but only to their “expression”. You have the right to re-implement the ideas of the program without fear of copyright infringement. However, the ideas may be protected by patents. Merely making, using or selling the patented software can subject you to claims of patent infringement. You can infringe a patent even without copying it, and even if you independently and innocently arrive at the exact same idea. So if some aspects of the proprietary program are protected by patent, you will have to find a way to implement those portions of the program without using any of the patented technology.

Is there any advantage to registering my copyright in a program?

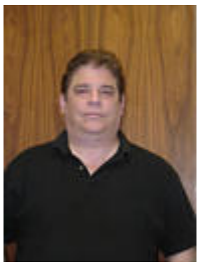
—Mark Jordon, 1707WebWay.Com

Registering a copyright is a simple and inexpensive process. The procedure is well documented in Circular 61, Copyright Registration for Computer Programs, available from the Library of Congress. For further information, go to www.loc.gov/copyright/circs/circ61.pdf. Registration is not a condition of copyright protection. However, there are several advantages to registering your work:

- Registration establishes a public record of the copyright claim.
- Before an infringement suit may be filed in court, registration is necessary for works of US origin.
- If made before or within five years of publication, registration will establish prima facie evidence in court of the validity of the copyright and of the facts stated in the certificate.
- If registration is made within three months after publication of the work or prior to an infringement of the work, statutory damages and attorney's fees will be available to the copyright owner in court actions. Otherwise, only an award of actual damages and profits is available to the copyright owner.
- Registration allows the owner of the copyright to record the registration with the US Customs Service for protection against the importation of infringing copies.
- You may register your copyright any time within the life of the copyright.

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

Lawrence Rosen is an attorney in private practice in Redwood City, California (http://www.rosenlaw.com/). He is also executive director and general counsel for Open Source Initiative, which manages and promotes the Open Source Definition (http://www.opensource.org/).



Kylix

Petr Sorfa

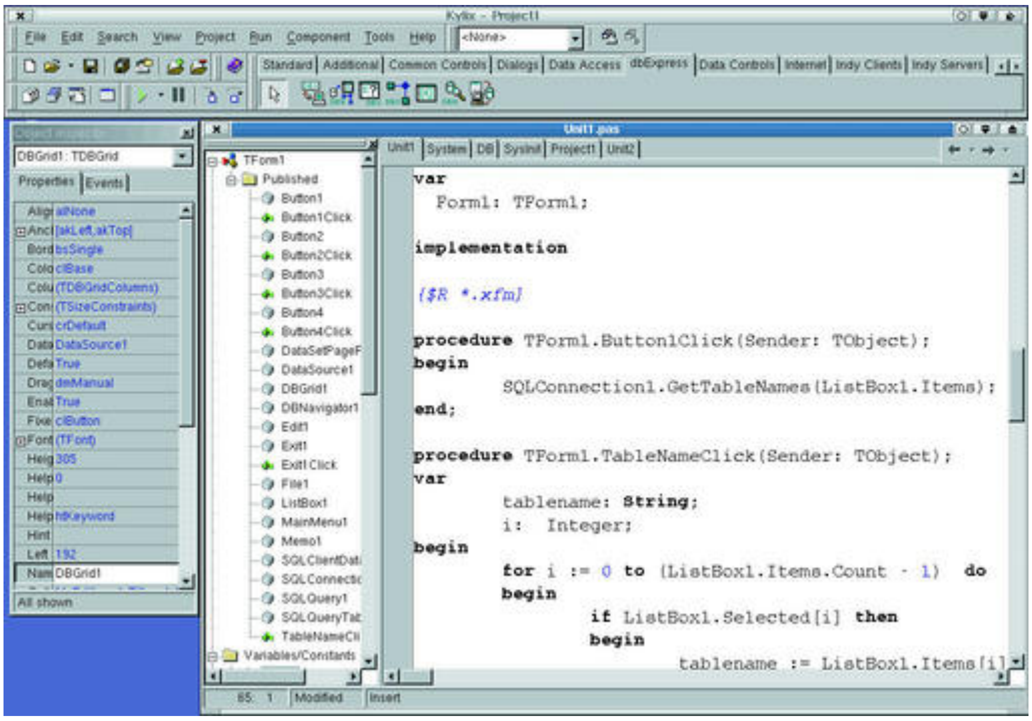
Issue #88, August 2001

Borland is creating a commercial market for Linux that has been poorly represented until now.

- Manufacturer: Borland Software Corporation
- E-mail: customer-service@borland.com
- URL: http://www.borland.com/kylix/
- Price: GPL development version—free download (or $99 US for a hard copy version), Developer—$999 US ($199 until August 23, 2001), Server—$1,999 US
- Reviewer: Petr Sorfa

Kylix represents an important and, hopefully, successful step for program development for the Linux operating system. Borland is creating a commercial market for Linux that has been poorly represented until now. Borland is embracing the Linux community by making certain Kylix components open source and releasing an open edition of Kylix at no cost (expected sometime in the summer of 2001).

Kylix is an integrated development environment (IDE) for the creation of software. More importantly, it is a rapid application development (RAD) tool that provides components for quick development of database connectivity, internet content and graphical user interfaces (GUIs, see Figure 1).



Aligi	alNone
Anci	[akLeft,akTop]
Bord	bsSingle
Colo	clBase
Colu	(TDBGridColumns)
Con	(TSizeConstraints)
Curs	crDefault
Data	DataSource1
Defa	True
Drag	dmManual
Enal	True
Fixe	clButton
Font	(TFont)
Heig	305
Help	0
Help	
Help	htKeyword
Hint	
Left	192
Nam	DBGrid1

```
var
  Form1: TForm1;

implementation

{$R *.xfm}

procedure TForm1.Button1Click(Sender: TObject);
begin
        SQLConnection1.GetTableNames(ListBox1.Items);
end;

procedure TForm1.TableNameClick(Sender: TObject);
var
        tablename: String;
        i:  Integer;
begin
        for i := 0 to (ListBox1.Items.Count - 1)  do
        begin
                if ListBox1.Selected[i] then
                begin
                        tablename := ListBox1.Items[i]
```

Figure 1. Kylix

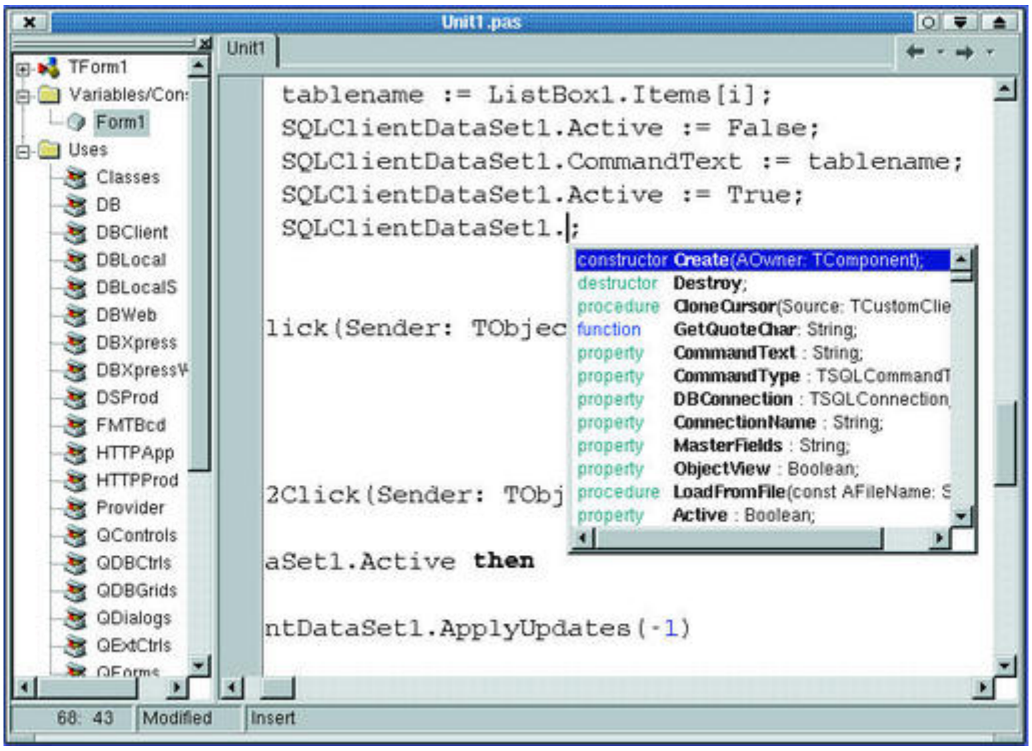
Kylix took two years to create, which in my experience is a short development time for a product so diverse and complex. I had heard of it for the first time one year into its development and have been waiting eagerly to use it ever since. Borland brings back memories of my first commercial programming language Pascal, and I was interested to see how things have progressed over the years. In addition, my copy arrived taped to the bottom of a Borland skateboard—Borland is tapping into those good old memories.

First Test—Dive Straight in

My first test was to try to create a GUI front end to my MySQL database. I was unsuccessful, even though it looked really easy at the Borland demo. Unfortunately, you must read the tutorial book before using Kylix. In a sense, the fault is mine as my experience is developing applications on UNIX platforms using products like KDevelop, whereas Kylix essentially is based on the Windows mindset. The differences are subtle but enough to force you to read the manual.

What Does It Offer?

The Kylix editor is definitely one of the most capable editors I have ever seen for Linux. Apart from color highlighting program sections while you type, it attempts to predetermine what you are typing (including programming constructs) and assists you with function parameters and class elements (see Figure 2). This feature also analyzes a class declaration and automatically creates functions with default content. Emacs fans can rejoice, as the editor can be set to handle Emacs key mappings.



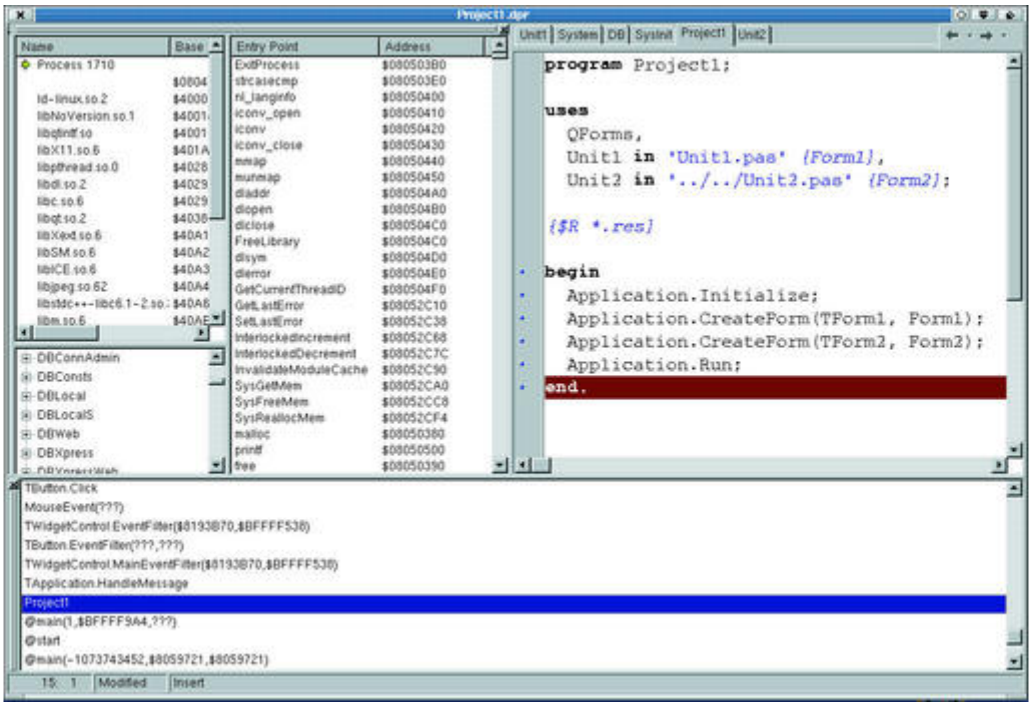
```
tablename := ListBox1.Items[i];
SQLClientDataSet1.Active := False;
SQLClientDataSet1.CommandText := tablename;
SQLClientDataSet1.Active := True;
SQLClientDataSet1.|;

lick(Sender: TObjec

2Click(Sender: TObj

aSet1.Active then

ntDataSet1.ApplyUpdates(-1)
```

```
constructor Create(AOwner: TComponent);
destructor  Destroy;
procedure   CloneCursor(Source: TCustomClie
function    GetQuoteChar: String;
property    CommandText : String;
property    CommandType : TSQLCommandT
property    DBConnection : TSQLConnection
property    ConnectionName : String;
property    MasterFields : String;
property    ObjectView : Boolean;
procedure   LoadFromFile(const AFileName: S
property    Active : Boolean;
```

Figure 2. Code Insight

Okay, there is something really sneaky about the compiler—it's just too fast. When I created my first program in Kylix, I pressed the compile button and nothing happened. So I pressed it again. I only then realized the program compiled so quickly that the compiler had finished its job while my clicking finger was still coming to a rest. The compiler is available as a command-line program, **dcc**. **dcc** also incorporates the linker. This is particularly useful for batch builds and development without the IDE for those die-hard **vi** power users or text terminal development.

Debugging is the standard useful fare. It is a good implementation but nothing extraordinary (like changing the code while debugging and carrying on execution without restarting the program). I was extremely pleased to discover that GDB supports the Kylix-generated binaries. Unfortunately, I could only use GDB to debug at machine code level as the symbol table failed to load. I either was using an incorrect version of GDB or did not have a correct compilation flag set. The Kylix debugger and GDB provide a win-win situation for beginner and advanced users alike (see Figure 3).



Name	Base
Process 1710	
	$0804
ld-linux.so.2	$4000
libNoVersion.so.1	$4001
libqtintf.so	$4001
libX11.so.6	$401A
libpthread.so.0	$4028
libdl.so.2	$4029
libc.so.6	$4029
libqt.so.2	$4036
libXext.so.6	$40A1
libSM.so.6	$40A2
libICE.so.6	$40A3
libjpeg.so.62	$40A4
libstdc++-libc6.1-2.so.	$40A6
libm.so.6	$40AE

Entry Point	Address
ExitProcess	$080503B0
strcasecmp	$080503E0
nl_langinfo	$08050400
iconv_open	$08050410
iconv	$08050420
iconv_close	$08050430
mmap	$08050440
munmap	$08050450
dladdr	$080504A0
dlopen	$080504B0
dlclose	$080504C0
FreeLibrary	$080504C0
dlsym	$080504D0
dlerror	$080504E0
GetCurrentThreadID	$080504F0
GetLastError	$08052C10
SetLastError	$08052C38
InterlockedIncrement	$08052C68
InterlockedDecrement	$08052C7C
InvalidateModuleCache	$08052C90
SysGetMem	$08052CA0
SysFreeMem	$08052CC8
SysReallocMem	$08052CF4
malloc	$08050380
printf	$08050500
free	$08050390

```
DBConnAdmin
DBConsts
DBLocal
DBLocalS
DBWeb
DBXpress
```

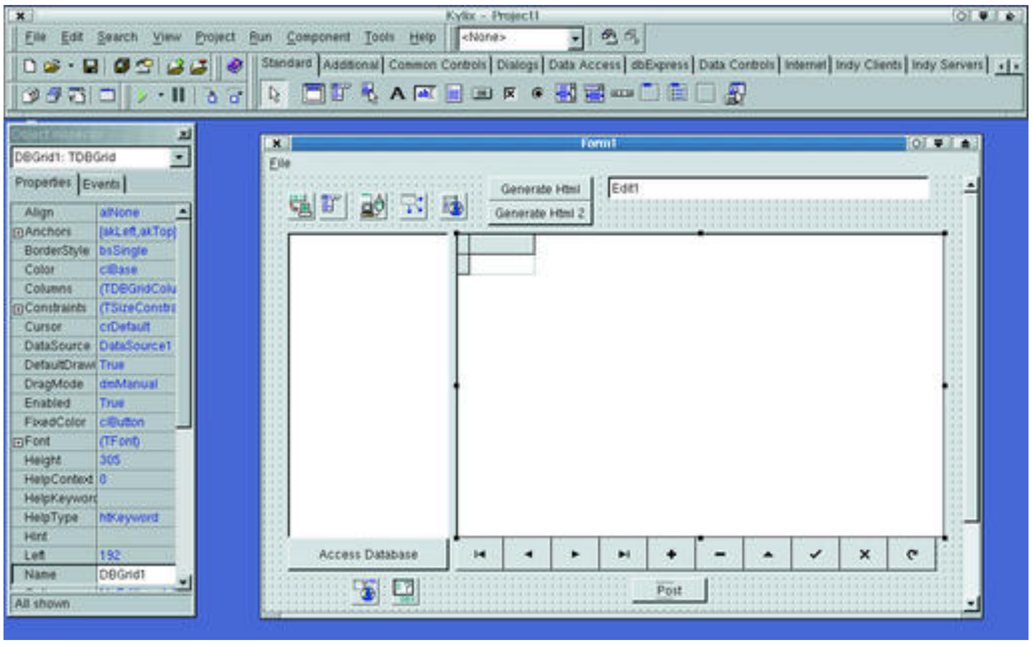
```
program Project1;

uses
  QForms,
  Unit1 in 'Unit1.pas' {Form1},
  Unit2 in '../../Unit2.pas' {Form2};

{$R *.res}

begin
  Application.Initialize;
  Application.CreateForm(TForm1, Form1);
  Application.CreateForm(TForm2, Form2);
  Application.Run;
end.
```

```
TButton.Click
MouseEvent(???)
TWidgetControl.EventFilter($8193B70,$BFFFF538)
TButton.EventFilter(???,???)
TWidgetControl.MainEventFilter($8193B70,$BFFFF538)
TApplication.HandleMessage
Project1
@main(1,$BFFFF9A4,???)
@start
@main(-1073743452,$8059721,$8059721)
```

```
15: 1    Modified    Insert
```

Figure 3. Debugger

The Kylix GUI builder is a good commercial front-end builder (see Figure 4). Kylix extends what is usually expected from GUI builders (as RAD components), such as database connectivity and web server tools (which have nothing really to do with the GUI), by actually representing them via the GUI. This use of nonvisual RAD programming components in the GUI allows for easy access to associated GUI front ends.



DBGrid1: TDBGrid

Properties	
Align	alNone
Anchors	[akLeft,akTop]
BorderStyle	bsSingle
Color	clBase
Columns	(TDBGridColu
Constraints	(TSizeConstra
Cursor	crDefault
DataSource	DataSource1
DefaultDrawi	True
DragMode	dmManual
Enabled	True
FixedColor	clButton
Font	(TFont)
Height	305
HelpContext	0
HelpKeyword	
HelpType	htKeyword
Hint	
Left	192
Name	DBGrid1

All shown

Figure 4. Form Designer

The one thing I did expect, but did not find, was a graphical way of setting up relationships between GUI components, for example, connecting a button via a line to a dialog box to establish a relationship between the button and the dialog box (such as a button click to close the dialog). This sort of graphical-relationship building is available in several other products, most notably Trolltech's Qt Designer for the Qt Widget set. As a side note, Kylix uses the Qt Widget set as a base for its cross-platform components (CLX).

Since I was using the Server version of Kylix, I had access to all the components that are available from Borland (see Figure 5). There are some excellent web server utilities for development with Apache. The database components are part of a collection called dbExpress. The dbExpress philosophy retains the live data in the computer's memory, and only when an update is initiated or needed is the data sent to the actual database. This approach apparently prevents extended locks on the database and cuts down unnecessary network traffic.

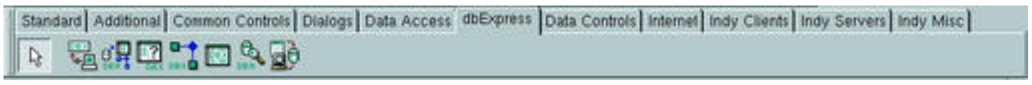


Figure 5. Components

I was disappointed that there was no OpenGL component. Granted, Kylix is aimed primarily at business users, but a little 3-D in an application is always a welcome sight.

As far as project maintenance, Kylix provides a to-do list that can be shared amongst a development team. The Code Explorer allows quick perusal of the code structure. An object repository allows project members to share sections of work. Although Borland assured me source control products, such as CVS, could be integrated with Kylix, a third-party interface product needs to be developed to support it.

Another problem is that the documentation is inadequate. For a $2,000 product, I expect several good books included in the package. All that is included is a well-written tutorial book that covers only the basics and really needs to be expanded to provide a good springboard for new Kylix developers. The program reference guide is sufficient but is obviously not a substitute for the on-line documentation. Fortunately, the on-line documentation provides good coverage and includes many examples. What is really missing is a guide for C/C++ programmers for Kylix and the Object Pascal programming language or some sort of computer-based training (CBT) introduction. Most Linux application programmers are C/C++ programmers, and a guide like this would have been a nod to the Linux platform. I strongly recommend that new Kylix developers read the whitepapers on Borland's web site, particularly when coming to grips with the database and web components.

There are three editions of Kylix: Open, which is available for free (downloadable) for noncommercial GPL development (or $99 US for a hard copy version); Developer, for commercial use with a limited number of features and components ($999); and Server, with all of the features and components ($1,999).

I was really hoping that the $99 version would be a stripped-down version that would allow you to create personal or small commercial products. The Open edition is available on the condition that any code that is produced with it using CLX must be released under the GPL. This implies that the source code of these programs is to be made publicly available. Although this has great open-source merits, it does effectively prevent a majority of commercial development. It looks like the $999 Developer version is the cheapest commercial-quality one and that there does not seem to be a cheap commercial version that matches the Delphi standard edition for Windows.

However, the Borland community is undoubtedly strong and helpful. The on-line documents and newsgroups are an excellent and necessary source of information. Most of my difficult problems were answered by reading the Borland newsgroup messages. In addition, several Borland folks have an active and helpful presence on the newsgroups (http://www.borland.com/newsgroups/).

Installation

I tried to install Kylix on several Linux distributions. Unfortunately, the supported Mandrake 7.2 did not install cleanly, and I had to use Borland's libc to circumvent the installation problem. A Linux system, which I have hacked together from various bits and pieces, had no problem installing and running Kylix. To assist potential Kylix developers, Borland has made a Kylix pre-test tool to check if your Linux installation has all the required components. You can download the Kylix pre-test tool from http://www.borland.com/devsupport/kylix/downloads/.

Programming

The initial version of Kylix supports the Object Pascal programming language, which is a merger of Pascal, Modula 2 and object-oriented concepts. The language is probably easier to learn for absolute programming beginners as it is closer to the English language, very much like a scripting language. My major gripe was getting used to the := symbols for assigning variable values. While programming, I appreciated the Kylix editor that helps you to program and use the components.

The Borland component library for cross platforms (CLX) is a strange beast, particularly for a seasoned Motif/Qt/Java programmer. Its API names and relationship logic have a Windows flavor and take some time to get used to and understand what function or class will do what you want. Borland has also released CLX as open source on the sourceforge.net site at http://freeclx.sourceforge.net/. CLX covers all the core elements needed to build a graphical application.

Regarding databases, although Kylix supports MySQL, it has a major problem at the time of this writing for it only supports the MySQL with the library libmysqlclient.so.6.0.0. Please note that any other version, later or earlier of libmysqlclient.so, will cause your application to crash. This caused me a lot of hassles, and I was not alone; a quick look at the newgroups showed others with a similar problem. However, once you get past the setup difficulties, creating database applications for Linux has never been easier. Due to the nature of the Borland's database approach, it is actually possible to have a database that resides in memory only and then dump it as a binary or XML file on your local system. Borland provides components to connect to several of the major databases, particularly Oracle. It is possible to create your own database support as well.

Although an OpenGL interface is provided on a second CD and is downloadable from www.delphi-jedi.org/DelphiGraphics/index.htm, it was fairly clear that it had not been tested extensively (also indicated by the lack of comprehensive examples). I had enormous trouble getting the GLUT toolkit for OpenGL to work and eventually abandoned the effort.

One thing the Kylix team might have overlooked is that Linux programmers may want to develop a command-line program to do some background task. For these kinds of applications, one generally uses the C/C++ argc/argv parameters to obtain the command-line arguments. Although there are several possible ways of doing it inside Kylix, none of them seemed to work. I had to search the Internet before I found a fairly obscure way of copying memory to obtain the argc/argv data.

Portability

One of the major attractions of Kylix is the ability to develop applications on Linux and then port them to Windows using Delphi. Granted there are several restrictions, but if you stick to using the Kylix API set, CLX, you are on a clean porting path.

Third-Party Extensions and Tools

To make Kylix even more attractive to developers, several third-party extensions and tools are available, ranging from UML support to code profiling. A companion CD shipped with Kylix makes many of these available.

The Linux Competition

Kylix not only faces competition from proprietary Linux IDEs, such as SNIFF+, SlickEdit and CodeWarrior, but also from the open-source projects, mainly KDevelop, Qt Designer, KDE Studio and several others. However, none of these provides the number of components and scope of Kylix. In this area Kylix is the definite winner.

The Future

Although Kylix is an exciting product, it does cater to a certain audience: existing Delphi programmers and development teams aiming for portability of an enterprise/business application to Linux and Windows. The main problem, of course, is the Object Pascal programming language itself. It might not be enough to convince C/C++ programming shops to swap over to Kylix yet.

Borland is hoping to release a Kylix version that supports C++ at the end of this year. It may be actually possible to compile the Linux kernel with this version of Kylix (or so I have heard). Now, this is really a product to be excited about—a possible GCC killer?

Another enhancement that Borland might address in the future is support of architectures other than Intel.

Conclusion

Kylix is an extensive development product and delivers more than any other commercial equivalent for Linux through its business-oriented components. Borland is serious about being part of the Linux community and has shown this by releasing the Open edition for no cost to download and $99 for physical media, as well as making the CLX components available as open source. If the Object Pascal programming language is not an issue, Kylix is the development tool for commercial Linux development. The Open edition serves the Open Source community. Although there are one or two hiccups, Borland has bravely paved the way for commercial development on Linux.

The Good/The Bad

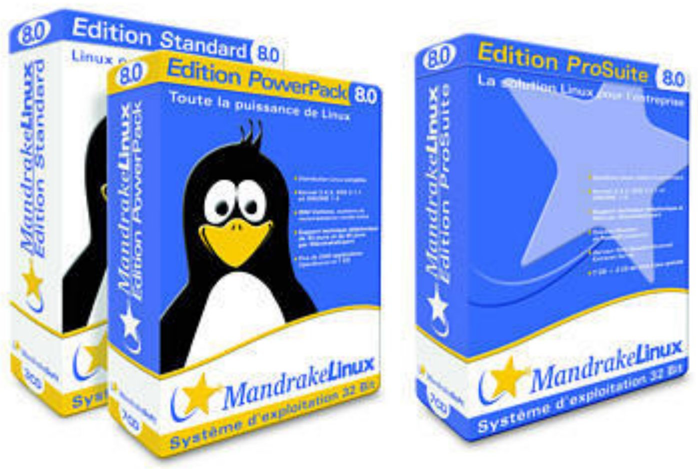
Petr Sorfa (petrs@sco.com) is a member of the Santa Cruz Operation's Development Systems Group, where he is the maintainer of the cscope and Sar3D open-source projects. He has a BSc from the University of Cape Town and a BSC Honours from Rhodes University. His interests include open-source projects, computer graphics, development systems and sequential art (comics).

Mandrake 8.0

Choong Ng

Issue #88, August 2001

Mandrake 8.0 adds kernel 2.4.3, XFree86 4.0.3, KDE 2.1.1, GCC 2.96 and miscellaneous other upgrades to 7.2.



- Manufacturer: MandrakeSoft
- E-mail: sales@mandrakesoft.com
- URL: http://www.mandrakesoft.com/
- Price: $69 US (plus $20 for shipping)
- Reviewer: Choong Ng

Linux Mandrake 8.0 is the latest release from MandrakeSoft. Released on-line on April 19, 2001, Mandrake 8.0 adds kernel 2.4.3, XFree86 4.0.3, KDE 2.1.1, GCC 2.96 and miscellaneous other upgrades to 7.2 (the previous release). Mandrake is advertised primarily as a workstation and desktop-oriented distribution, so I will be evaluating it with these applications in mind. Supremely important in a desktop distribution are ease of setup, ease of configuration, ease of operation, quality of the software bundle and ease of installing new software; I will focus primarily on these criteria. My test machine for this review is a PIII 560 with 384MB of RAM, a 6GB hard disk and a Matrox G400.

Setup and First Impressions

Mandrake 8.0 comes on two (download edition) or four (retail version) CD-ROMs; the first two of which will be necessary for most installations. As with nearly all modern distributions, loading the first CD and rebooting is all that is necessary to start the installation. Mandrake 8.0's graphical installer closely resembles the installer used in the 7.x releases and is really about as user-friendly as an OS installer should be. Once the installer loads, it prompts the user for information such as how the computer will be used and what broad categories of packages to install. Many of the decisions that users new to computers or new to Linux (e.g., converting from Windows, more about this below) won't necessarily know how to answer, such as sizing and placement of partitions, are handled by the installer's defaults. As with the 7.x releases, you can go back to any previous part of the install by clicking on the button next to the appropriate step on the left side of the screen. Even X configuration—the area where I most often run into trouble—went without a hitch. This appears to be at least in part due to the move to XFree86 4.0.3, a significant improvement despite lacking solid support for certain graphics chipsets (notably several by Chips and Technologies common in older notebooks). One option to pay special attention to is the option to log on a user at boot automatically. Once you have answered all of the configuration questions, the installer will install and configure the requested packages, meanwhile displaying annoying “hints” à la the Windows 98 installer. The install took a little less than 15 minutes from the first reboot (to start from the installation CD) to a KDE user desktop.

If you get the retail version of Mandrake 8.0 (this is discussed later), the startup process is completely graphical. The boot-loader menu is a text list with an abstract design in the background, and the Linux startup process with its status icons resembles a Mac OS boot more than anything else. While most *Linux Journal* readers likely will disable this feature in favor of actually seeing what's going on during the boot process, I can see it being highly beneficial to users who wouldn't know UNIX from eunuchs and think bash is a verb. When the desktop finally loads (startup could take a minute or two on slower hardware) the user is presented with a very Windows-esque interface. Good or bad, that's what it is (see the next section). At first examination this represents a fairly well set up desktop, with icons for documentation, connection to the Internet, etc., right on the desktop and the most useful applications and utilities already in the KDE menu.

Desktop Usability

Mandrake 8.0 is definitely a step forward in terms of desktop usability. Between the KDE 2 desktop and the pre-installed productivity applications, MandrakeSoft has managed to put together a very usable desktop environment suitable for many users' needs. The default install by no means caters to the advanced user or the server market; unlike the 7.x series of releases, Mandrake 8.0 doesn't include any alternative window managers nor does it install (by default) many popular services such as SSH and NFS. This is clearly a very desktop-oriented distro. While Mandrake doesn't really add much to the functionality of user applications—KDE 2 and bundled applications take care of most of that—it does bring a marked improvement to system configuration. The graphical configuration utilities included (Mandrake Control Center, User Manager, etc.) are a distinct improvement over the utilities included with most other distros including previous releases of Mandrake. Where users once had to guess at cryptic names not obvious without documentation, now there are obvious names and easy-to-operate utilities like Mandrake Control Center to ease the process. This is definitely a step forward.

That said, there are still significant problems. First and foremost for desktop users is the presence of GNOME and other non-KDE applications in a (by default) KDE environment. While this isn't a problem on its own, the fact that the GTK toolkit produces a significantly different look and a new set of widgets makes for a less consistent interface. For new users this shouldn't present much of a problem (you can just train them on both) nor should it present much of a problem for many Windows users (many Windows applications have bizarre nonstandard interfaces), but it may be an issue if you are a migrating Mac OS user who is used to software written by highly interface-conscious developers sporting carefully designed interfaces and consistent UI elements throughout. This too can be worked through, nonetheless it is something to be planned for if migrating users from one of these platforms.

Performance

Subjective performance is very good. I attribute this largely to the move from the XFree86 3.x series of releases to 4.0.3. The move to KDE 2.1.1 may also contribute—most of the review is conducted under KDE except where noted. Performance on SMP systems should also be markedly improved due to improved SMP support in kernel 2.4. One issue that anyone intending to install Mandrake 8.0 should be aware of is that the default configuration (XFree86 4 and KDE 2) needs quite a bit of RAM. I would strongly recommend 128 or 256MB of RAM. I would also recommend installing a graphics card with good 2-D acceleration under Linux, such as a Matrox G200 or ATI RAGE PRO (note: these are two inexpensive cards I have personally found to work well, but there are many others out there), as this will improve the subjective performance of X applications greatly. Between the video card and RAM you should be able to get Mandrake running acceptably without much trouble.

Other Notes

Mandrake 8.0 has a few technology updates and niceties that made my life significantly easier. One is fairly good support for USB. While it still doesn't allow transparent hot-plugging of USB devices, Kudzu did detect my USB mouse at boot without problems, and even the scroll wheel—extremely useful to those of us who are on the Web a lot—worked without any user configuration. While hot-plugging still isn't supported (as noted above) and USB and PS/2 mice sporadically cannot operate simultaneously, support is good enough for a desktop system. I looked specifically at support for the 2.4 series of kernels, especially with respect to issues with building and installing modules. I had no issues save one involving the menu entry for the correct USB controller not appearing in menuconfig, and I believe that this is fixed in kernel 2.4.5. Overall, the move to 2.4 appears to be a good thing.

I also appreciated the inclusion of Galeon—one of the better browsers available for Linux—along with the usual Konqueror, Netscape Communicator and Mozilla. Overall Mandrake is becoming a complete system suitable for everyday desktop use.

There are, however, a few gotchas associated with the version upgrades. Be sure that your graphics card is well supported under XFree86 4 because this is potentially the cause of all manner of apparent application instability and general glitching. At one point I had problems getting my graphics card to switch to 1024 × 768, but manually editing my X config took care of this. I expect the X configuration issue to be fixed in an update to Mandrake Control Center. Another thing to be aware of is that the downloadable version of Mandrake 8.0 is not exactly the same as the retail version. There is the obvious difference that the download edition comes with two CDs rather than the four that the retail version includes (it omits the commercial packages), but I noticed that the download version did not have the graphical overlay to the kernel boot process enabled. While I didn't notice any other concerns, it is possible that there are more hidden differences lurking in the details of some package's configuration. Overall, though, I found no problems that make Mandrake 8.0 any less usable than the previous release.

Conclusion

Mandrake 8.0 is certainly worth consideration for workstation and desktop use, especially when deploying to users who are not already trained in using Linux systems. It may be a highly useful tool for migrating users to Linux from other popular platforms; version 8.0 appears geared toward minimizing retraining costs for migrating users. The real downside to Mandrake 8.0 is its resource usage; if you are already running a number of Linux workstations on low-end hardware—perhaps 233MHz machines with 64MB of RAM, a likely configuration for an older network—you will probably need hardware upgrades to get reasonable performance out of your machine. To put it shortly, Mandrake 8.0 is best as a desktop environment and is definitely a top candidate when a user-friendly interface is high on your list of priorities.

The Good/The Bad

Choong Ng is the product reviewer at *Linux Journal* and a teriyaki connoisseur.

Archive Index Issue Table of Contents

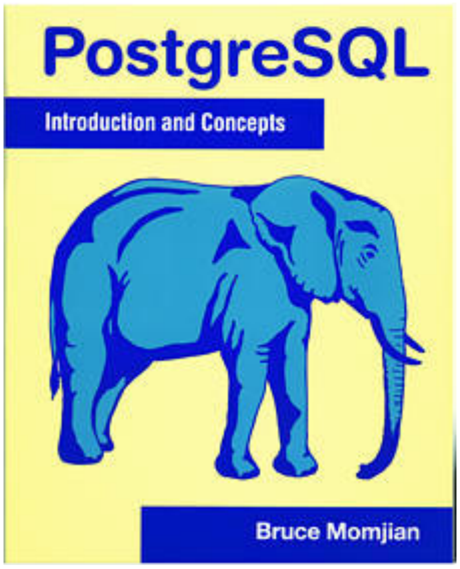
Advanced search

PostgreSQL: Introduction and Concepts

Stephanie Black

Issue #88, August 2001

PostgreSQL: Introduction and Concepts is an excellent introduction, not only to PostgreSQL in particular, but to databases in general.



- Author: Bruce Momjian
- Publisher: Addison-Wesley
- URL: www.postgresql.org/docs/awbook.html
- Price: $44.95 US
- ISBN: 0-201-70331-9
- Reviewer: Stephanie Black

It's hard to call oneself a fan of databases. Usually, one uses these kinds of applications out of sheer necessity, not because they're within an arm's reach of “exciting”. If databases themselves don't bore new users into out-of-mind experiences, the books that come with them usually do. Usually, the books assume knowledge users don't have, speak above/below the user's comprehension level or leave out all the useful pieces of information.

PostgreSQL: Introduction and Concepts (Addison-Wesley, 2000) is an excellent introduction, not only to PostgreSQL in particular, but to databases in general. It takes the novice database user through all the steps of the TEST database and does so without obfuscation or confusion. For the seasoned database designer/maintainer, it offers a wonderful reference to programming interfaces (including PHP, Embedded C and Python), **pgsql** commands and other features unique to PostgreSQL.

As of this writing, the book is in its second printing, having sold a massive 2,155 copies in the first month after its release last November. Given the increasing number of people making the transition to open-source software, and the number of people who need a good reference manual for database servers, the need for a book such as *PostgreSQL* is evident.

Momjian, a founding member of the PostgreSQL Global Development Team, has included a wide range of information on topics ranging from basic database setup to more senior administrative subjects, such as host/hostssl connections, backing up/restoring databases and internationalization. The average user will learn to run **psql** to access her/his database, enter data, view existing tables and make queries.

This is not, however, merely a glorified HOWTO: numerous features in *PostgreSQL* are useful to programmers, webmasters/mistresses, as well as DBAs. There are a fair number of programming interfaces for PostgreSQL, broadening its potential uses among computer professionals. This book certainly unlocks much of that potential in its discussion of the languages PostgreSQL can interface with, complete with sample programs illustrating the process.

The book is just as informative about what can be done with this database software as it is about how to do it. From this perspective, it is a success. It is not, however, perfect.

Some organization is present—a good thing to be sure in a book about databases. The trouble is, it's apparent that Momjian hasn't been around database newbies for a while. Were “Appendix B: Installation”, “Chapter 2: Issuing Database Commands” and the first portion of “Chapter 20: Administration” all put together, the new PostgrSQL user would have an easier time setting things up. As it is, one wanders around those three sections for a while, trying to get the hang of things. Momjian has included a fair amount of information about database setup and user creation in “Administration”, which makes sense if he assumes the user will be on a network that is administered by a seasoned DBA. It is inappropriate, however, for an individual who wants to create and maintain a database on a single-user system. Get ready for a lot of flipping back and forth if you fall into the latter category.

I believe every technical book should have three qualities: good indices, clear writing and wise use of appendices. Regrettably, Momjian has included appendices to the tune of 50% of the book, which tends to detract from his work and gives the reader a sense of the book being padded. Certainly an appendix with additional resources is moderately helpful, but a reader shouldn't need a FAQ listing for software that is on the Internet. Such information is best left on-line. The reference manual, included in Appendix D, is helpful but large enough (nearly 200 pages!) to warrant its own cover or reference (with its URL) as an additional resource in Appendix A.

That said, the remainder of the book is well thought out, and the content of chapters is in accord with their titles.

Technical Glitches

There are some commands that, according to the book, should work but don't. To be fair, this could be a discrepancy between the version used to test the book's commands and the version in current use. The most recent release is 7.0.3; testing of some of the commands was done with 6.5.3. (If you're using the 6.5.3 release, it would be best to update it; if you're involved in the release of a Linux distribution, you'll want to clamor for the inclusion of a more recent version of PostgreSQL.) When I tried CREATE USER I got:

```
sjb=> CREATE USER paa;
CREATE USER
sjb=> CREATE GROUP alteregos WITH USER sjb, paa;
ERROR: parser: parse error at or near "group"
```

Several variations of this were attempted with the same results. An associate attempted the same commands using the same release of the software, with the same results. Such situations are, alas, not addressed in the book.

NULL/NOT NULL values also represent problems with this version of the software, at least on the Red Hat 6.2 and Storm 2.0.6 systems that were used. While the author of the book isn't necessarily responsible for these errors, it would have been helpful if possible workarounds were included in the book. There is no recognition that PostgreSQL has its “moments”.

Conclusion

Aside from some relatively minor problems as mentioned above, *PostgreSQL* is a solid reference for constructing, manipulating and managing a database. As the software grows, so will its audience. As to whether it's worth the purchase price, one of the best things about this book is its availability on-line. If you are a PostgreSQL guru and need only an occasional reference, you might well forego purchasing the book and use the on-line version. Otherwise, it's a remarkably good investment.

Stephanie Black is a writer—of words and code. When not writing, she runs a Linux consultancy, Coastal Den Computing, in Vancouver, BC, Canada. In her off-hours, she's usually playing fetch with her cats or collaborating/colluding with her partner, a fabric artist and business manager.

Letters

Various

Issue #88, August 2001

Readers sound off.

Don't Change Your Hair for Me

I have been reading your magazine for about four years and have to say that I love it. I have friends that say that there isn't enough flash to it (maybe a little glitter would help), but I say that there really isn't anything that I would change. I love the inventor/hacker spirit. There is this feeling that there is always something cool to do with Linux that no one else tried on a Mac or Wintel machine. I mean from using wireless communication in Africa to running your can opener in your house from a workstation in Borneo. Okay, maybe not the latter, but I am waiting for it or the in-dash Linux MP3 player. Now that is some rockin' good stuff. I say keep up the good work.

—Matthew James

See Don Marti's interview with Hugo Fiennes and Marc Merlin in the July/August 2001 issue of our sister publication *Embedded Linux Journal* for information on the Rio Car, an in-dash Linux MP3 player.

—Editor

Crash Free

A few years ago, you sent out Linux bumper stickers to some of us subscribers, for some reason, I forget why. I thought you'd like to know that my car hasn't crashed since I put Linux on it.

—Keith Blackwell

Agree with Review (Mandrake 7.2 March 2001)

I currently use Mandrake 7.2. I upgraded from a previous Mandrake version. I noticed the May 2001 letter ("Mandrake 7.2 Review") and feel I must speak up in defense of the review.

Overall, I am pleased enough with the product (obviously, since I am still using it). However, I must admit the March 2001 *LJ* review's overall tone agreed with my opinion of the product.

I could go on about many things, but I'll just bring up a few. Their graphical installation is not anywhere close to robust. After I finally got the install done on my laptop, I responded in e-mail with a list of bugs, gotchas, etc., but did not get a reply. Many of the DrakConf programs have more than their share of bugs. For example, I have gotten into the habit of starting printer configuration (PrinterDrake) to do just one thing at a time. I exit the entire program before starting it up again to do another thing with it. I abandoned RpmDrake. It was very promising but simply didn't work.

I manually upgraded many RPMs in January from one of their mirror sites. Most of these upgrades were security related and didn't fix problems like these.

In short, I would prefer honest reviews over the watered-down "you scratch my back and I'll scratch yours" kind.

—Harry Rockefeller

See our review of Mandrake 8.0 on page 78.

—Editor

Searching for Meaning

I just received the June 2001 issue of the *Linux Journal* today. The front cover shows a small ship (looks like a cross between a steamer and a cruise ship) circling the globe. Is this a (subtle) advertisement for the Linux Lunacy Geek Cruises, or is there a deeper meaning to it?

—Vivek Bhatiavivekb@cs.ucf.edu

It's as deep as the answer to "How many surrealists does it take to screw in a lightbulb?"

—Editor

Offensively Offended

I am surprised and offended that your fine publication has seen fit to sneak in religious references (Matthew 9:17) in the June 2001 Best of Technical Support, page 96. Perhaps one of your staff members has a religious agenda that he wishes to impose on others. It has no place in a technical publication.

—Noel Moss

The reference is not religious but literary. The passage alone contains nothing religious and makes a good metaphor for the question answered.

—Editor

It Wasn't Me

As the founder and project leader of the Linux Terminal Server Project (LTSP), I'd like to tell you how excited I was to see the LTSP article in the June 2001 issue of *Linux Journal*.

It seems like the project is really becoming a mainstream feature of Linux. I'd also like to point out that while I think that Jorge did a great job in writing the article, I think it may give some people the impression that we have written our own bootroms. The bootrom images that we have had on our web site are entirely the works of Etherboot Project (Etherboot.sourceforge.net), which is led by Ken Yap. We had them available for download, just as a courtesy to people who want to get started quickly. We now refer people to the ROM-o-matic.net site. Marty Connor has done an excellent job of setting up that site, where you can select options on the screen and have a bootrom image generated within seconds, which you can then download and write to a floppy disk for testing.

I've always tried to make it clear that we, at the LTSP, didn't really invent anything new here, we just pulled together several technologies that have existed for years and made it easier for the average sysadmin to deploy.

If we've been successful at that, it is only because we are standing on the shoulders of giants (to quote Isaac Newton and Linus Torvalds).

Thank you for a wonderful magazine as always.

—Jim McQuillanLinux Terminal Server Projectjam@Ltsp.org

Archive Index Issue Table of Contents

Advanced search

UpFRONT

Various

Issue #88, August 2001

Stop the Presses, *LJ* Index and more.

Linux Bytes Other Markets—Linux Outperforms Solaris Running

Financial Risk Management Systems

Selling Linux solutions to the financial industry is a tough job. This traditionally conservative industry has been a slow adopter of the Open Source movement. Even as this is gradually changing, better, cheaper, faster Linux solutions aimed at the financial industry begin to appear. Last year Reuters developed Value at Risk, a statistical approach predicting how much your financial investments can lose given a financial horizon. The targeted development platform for the engine was Solaris 2.7. Once it was completed, we decided to give Linux a try. Most of the third-party libraries being available—a sure sign Linux is gaining weight in the financial industry—I ported the core of the engine itself (about 100 KLOC) to Linux 2.2.14 SMP (Red Hat flavor, multiprocessor).

We tested the financial engine with Solaris on a Sun 450, a four-processor machine running at 300MHz, and with Linux on a two-processor Pentium III running at 733MHz. Table 1 shows the results on a typical individual portfolio (30+ stocks).

Table 1. Performance Comparison

On Linux the optimized engine was built using GCC 2.95.2.1 with -O3 and on Solaris using SPARCworks 5.0 with -xO3. A financial engine like ours is not I/O-bound but CPU-bound (the financial properties are cached). Most of the time is spent calculating hypothetical future prices of a financial instrument and manipulating these prices (additions/multiplications, and sorting operations are involved in the financial algorithms). These tests would suggest the combination Linux/Intel is at least twice as fast than the Solaris equivalent. They are designed not to put one platform over another but as the typical request an on-line broker might send to our engine. They are real life examples, taken from common portfolio configurations.

To be fair we did not anticipate such performance gain by switching to Linux. We had in mind the deployment costs for our customers, and we were even ready to lose a bit of speed. Surprised by these tests, we ran the following test case simulating a common financial algorithm, which led to similar results (see Table 2):

```
for(int j=0; j<10000; j++)
  {
    // Build a time series
    std::vector<double> ts;
    ts.reserve(2000);
    for(int i=0; i<2000; ++i)
      ts.push_back(random());
    // Simulate the perturbation
    for(int i=0; i<2000; ++i)
      ts[i]+=5.0;
    // Sort the time series
    std::sort(ts.begin(), ts.end());
  }
```

Table 2. Performance Comparison II

We proved here that Linux is a viable alternative to Solaris for our product offering. Customers can choose their own deployment platform, Linux, Solaris or a combination of the two, in which case our engines will communicate with one another, some running on Solaris, some on Linux. We knew a Linux box was as reliable as Solaris. We knew it was cheaper. We know today, at least for our type of application, that it is faster. It is definitely our platform of choice, and that will contribute to the adoption of Linux in the financial community, which is a very good thing.

—Sebastien Marc

Hacker Insurance

On May 28th, ZDNet reported that J. S. Wurzler Underwriting Managers—a company that offers “hacker insurance”--now charges between 5 and 15% more to companies who employ Windows NT for internet operations (www.zdnet.com/intweek/stories/news/0,4164,2766045,00.html). Of course the numerous security holes for which Microsoft is famous were cited as one of the reasons for the increase, but more surprising was the effect of higher employee turnover in companies that use Microsoft NT. The article relates:

> Wurzler found that system administrators working on open source systems tend to be better trained and stay with their employers longer than those at firms using Windows software, where turnover can exceed

> 33 percent per year. That turnover contributes to another problem: System administrators are not implementing all the patches that have been issued for Windows NT, Wurzler said.

The turnover results in reduced implementation of security patches, compounding the difficulties of weak MS security.

LJ History

In the August 1996 issue of *Linux Journal* we ran an article about Mobile-IP and its potential for allowing trouble-free mobile connectivity. With the advent of inexpensive 802.11b hardware and community projects such as Seattle Wireless (http://www.seattlewireless.net/) and the Bay Area Wireless User Group (http://www.bawug.org/), Mobile-IP is today just a few short steps from reality.

LJ Index—June 2001

1. Trillions of dollars lost by investors when the dot-com bubble burst: 4.6
2. Sum in billions of dollars in cash held by Microsoft: 30
3. Cash-holdings position of Microsoft among all US corporations: 1
4. Position Bill Gates and Steve Ballmer expect subscriptions (e.g., those from HailStorm) to hold among future revenue sources for Microsoft: 1
5. Longest uptime in days, currently recorded by the Linux Counter Project: 434
6. Number of Linux machines reading mail by OCR for the US Postal Service: 900
7. Female percentage of LinuxCertified.com students: 5-8
8. Percent of federal computing installations that already used Linux in 1998: 25
9. Number of Compaq Alpha workstations in the National Oceanic and Atmospheric Administration's (NOAA) Forecast Systems Laboratory's new Linux Beowulf cluster: 277
10. Number of other Linux-based clusters at the same laboratory: 2
11. Power multiplier of NOAA's new system over previous cluster: 20
12. Sum in billions of dollars in cash held by Dell: 8
13. Percentage rate per week at which component prices are falling in May 2001, according to Dell: 1
14. Days of inventory Dell keeps on hand: 3
15. Position of SGI 1450 Server with DB2 UDB EEE v7.2 running IBM DB2 UDB EEE 7.2 on Linux 2.4.3 in the Transaction Processing Performance Council's TCP-H (a decision support) benchmark in May 2001: 1

Sources:

- 1-4: *Business Week*
- 5-6: Linux Counter Project
- 7: LinuxCertified.com
- 8-14: CNN.com
- 15: Transaction Processing Performance Council (http://www.tpc.org/)

O'Reilly Perl Conference 5

Don't forget the Annual Perl Community Gathering—The O'Reilly Perl Conference 5 held July 23-27, 2001 in San Diego, California. As their web site says: “The O'Reilly Perl Conference has been a mainstay of Perl culture since 1997, gathering the leaders and young guns of Perl in an unequaled meeting of minds. TPC, as it is known, is a place where all Perl programmers can learn and share in the fun and diversity that is Perl.” See http://conferences.oreilly.com/perl/ for more details.

Stop the Presses: the Penguin Element

Microprocessors and operating systems are companion technologies, but storytellers (folks like us) usually treat the characters separately—even given major exceptions like Wintel and PowerPC. Now, as we look toward next-generation 64-bit chips from Intel and AMD, the stories tend to be about what those companies are doing vs. each other, rather than with the communities of developers surrounding the operating systems that put CPUs to work.

As we go to press, Intel's Itanium (aka IA-64) is rolling into production, almost exactly seven years after Intel and Hewlett-Packard announced the partnership that would work on the new chip design, which reportedly began its life in HP's labs as “PA-RISC Wide-Word”. The new joint design was code-named Merced and later given its elementine brand name. Sun Microsystems came to market with its UltraSPARC III 64-bit processor last fall, but that's bound to be less interesting to the Linux community than both Itanium and AMD's Sledgehammer, currently slated for release about a year from now.

But Itanium is here. As Linley Gwennap says, “Changes in servers never happen fast. But with Itanium now a reality, Intel's dominance is only a matter of time.”

And what operating system will most help achieve this dominance? The top press release linked from Intel's Itanium index page is titled “Intel Itanium-Based Systems Poised For Production”, and says this about operating systems:

> Four operating systems will support Itanium-based systems, including the Microsoft Windows* platform

> (64-bit Edition* for workstations and 64-bit Windows Advanced Server Limited Edition 2002* for servers); Hewlett-Packard's HP-UX 11i v1.5*, IBM's AIX-5L* and Linux. Caldera International, Red Hat, SuSE Linux and Turbolinux plan to provide 64-bit versions of the Linux operating system. [The asterisks disclaim ownership of various name brands.]

Note the future tense employed by the operative verb phrase here: *will support*.

On May 29, SuSE announced SuSE Linux 7.2 for IA-64 and Red Hat announced Red Hat Linux 7.1 for the Itanium Processor. The next day Turbolinux announced Turbolinux Operating System 7 for the Itanium. All three expressed availability in the present tense. At this writing, three of the four headlines among Intel's Itanium “news and events” involve Linux. The fourth is “Microsoft Unveils Plans for 64-Bit Windows Platform”.

As for hardware, SGI announced (also on May 29) Silicon Graphics 750 system for Linux, which it called “The First Itanium Processor-Based System Using Linux”. IBM also said it was building the second of the world's fastest Linux supercomputers at NCSA by clustering Itanium-based boxes with Turbolinux. It's due to be installed this summer.

Credit should be spread in many directions, but perhaps that can be done at once by pointing to the Trillian Project (now called LinuxIA64.org at http://linuxia64.org/), which began in May 1999 and brought together Caldera, CERN, Cygnus (now Red Hat), HP, Intel, Linuxcare, NEC, SGI, Turbolinux and VA Linux. Most of those organizations now have something to show for the effort.

So, it's already hard to imagine that Itanium's success won't be due largely to its adoption as a penguin element.

—Doc Searls

They Said It

Only the insane take themselves quite seriously.

—Sir Max Beerbohm

Before a mad scientist goes mad, there's probably a time when he's only partially mad. And this is the time when he's going to throw his best parties.

—Jack Handey

A quick change of transportation metaphors is now called for, from railroads to shipping, because however cleverly Michael Eisner, Rupert Murdoch, Steve Case and the rest of these broadband tycoons rearrange the deck chairs on their respective Titanics, an even more titanic iceberg with their names carved into it has already calved off some remote Arctic ice shelf and is inexorably drifting their way. That iceberg, of course, is the Internet.

—Christopher Locke

The network is a stochastic synchronicity generator.

—Christopher Locke

To be sexy, hackers need to learn how to emit fitness-to-reproduce signals.

—Eric S. Raymond

The average American has one breast and one testicle.

—Wiredog

Well, I think if you say you're going to do something and don't do it, that's trustworthiness.

—George W. Bush

Regular consumption of beer eliminates the weaker brain cells, making the brain a faster and more efficient machine. That's why you always feel smarter after a few beers.

—Cliff Claven, the character on *Cheers*

Don't bother just to be better than your contemporaries or predecessors. Try to be better than yourself.

—William Faulkner

You know, it's at times like this when I'm trapped in a Vogon airlock with a man from Betelgeuse and about to die of asphyxiation in deep space that I really wish I'd listened to what my mother told me when I was young!

Why, what did she tell you?

I don't know, I didn't listen!

—Douglas Adams

I love deadlines. I love the whooshing sound they make as they fly by.

—Douglas Adams

Science has lost a friend, literature has lost a luminary, the mountain gorilla and the black rhino have lost a gallant defender (he once climbed Kilimanjaro in a rhino suit to raise money to fight the cretinous trade in rhino horn), Apple Computer has lost its most eloquent apologist. And I have lost an irreplaceable intellectual companion and one of the kindest and funniest men I ever met. I officially received a happy piece of news yesterday, which would have delighted him. I wasn't allowed to tell anyone during the weeks I have secretly known about it, and now that I am allowed to it is too late.

—Richard Dawkins on Douglas Adams

We used the term open source not to piss off the fsf folks, but to claim a semantic space where we could talk about issues without scaring away the people whose beliefs we wanted to change.

—Eric S. Raymond

Great satire doesn't proceed from nihilism but from moral indignation. Compare Douglas Adams with Jonathan Swift.

—Eric S. Raymond

“Media relations” need to be outlawed in every country in the world. In the best case, media relations are incestuous relationships between dimwitted first cousins; on the Web it is one of the worst examples of inbreeding by the unholy alliance of ad agencies and pixel mechanics of dubious talent.

—The Head Lemur

It's a Diaper! No, It's a Sectional Bookcase!

Question: What's...

a ballpoint pena sectional bookcasea fire extinguishera vaginal fungicidea chemical for controlling eyespot or rynchosporium in barleya TV antennaa body appliancean eyeglass framea fauceta diapera hair salon

Answer: UNIX

Thanks for that information goes to none other than the operating system's most-credited creator, Dennis Ritchie. The full story is at his Bell Labs web site (cm.bell-labs.com/cm/cs/who/dmr/otherunix.html).

Archive Index Issue Table of Contents

Advanced search

Best of Technical Support

Various

Issue #88, August 2001

Our experts answer your technical questions.

Copying Old Tapes

Before I came to my current job, the archive system was sporadic at best, using multiple media formats from DAT to Exabyte. I have been given the task of copying all these formats to a single standard (in this case AIT). I am trying to copy one tape to another and cannot find a proper command. We are using the GNU version of **tar** and would like to keep this standard. I have tried

```
dd if=/dev/<device file> of=/dev/<device file2>
```

and cannot even get the command to read the tape. —Charles Long, charlesl@wildbrain.com

The command you are using looks correct, the only missing item may be a block size (bs) in case the format of the tapes was written with a specific block size factor (that you have to find out). Then the command will be

```
dd if=/dev/XXXX of=/dev/YYYY bs=<your_block_size_number>
```

—Felipe E. Barousse Boué, fbarousse@piensa.com

dd should read a tape, regardless of the program used to write the tape (e.g., **tar**, **dump**, NetBackup). The problem could be one of the following: 1) tape is written on a different/incompatible drive, 2) tape is faulty, 3) heads on drive are dirty or 4) the header on the tape. So skip before reading (use **mt fsf** and no-rewind device) and try reading the tape(s) with the original program in read-only mode to make sure the tape(s) and drive(s) are working correctly. If you cannot read the tapes in any way, then the backups are useless. —Keith Trollope, keith@wishing-well.demon.co.uk

Protecting a Second Hard Drive from autoinstall

I want to install Red Hat 7.1 on only my master disk. I have a slave drive with Win98 installed at present. The master disk is empty, so I'm not trying to dual boot my PC in that sense. Short of physically disconnecting it, is there a way to autopartition the one disk? How would you recommend I divide up my 40GB disk? —Paul Henman, linuxj@henman.ca

Red Hat definitely lets you partition the drives as you wish. I usually recommend the following: / 100MB /safe 100MB /usr 3-4G in your case, more if you want /var everything else

You need to symlink /tmp to /var/tmp/tmp and /home to /var/home. The advantage of this scheme is that your root partition is critical, so if you keep it small, you reduce the changes of corruption, and you can keep an on-line backup copy of it in /safe. —Marc Merlin, marc_bts@valinux.com

Starting X from a Bootdisk

I had installed SuSE 6.3 on an extreme partition of my HDD (so no LILO). I later created a bootdisk from the CD through the **rawrite** utility. I am able to access the command-line interface by booting from the floppy and specifying the installed partition. Can I initiate the GUI interface in any way? —Manoj Ramakrishnan, mark2gp@sify.com

There are different ways to initiate the GUI interface. You can directly start X from the command line with **startx**. You can start **xdm**, **kdm** or **gdm** as root; they often have an **init** script in /etc/init.d/, or they can be started from the command line. The last and probably easiest option is to set the right runlevel, 3, in YaST2. —Marc Merlin, marc_bts@valinux.com

Setting the Bytes per Inode

I would like to know how to set the inode size for my entire hard drive, each partition. Is there a way I can set the inode size during Red Hat 7.0's installation, or do I have to use **debugfs** to change inode sizes? —Matt Walters, matt.walters@syberos.com

You cannot change the bytes per inode after the filesystem has been created. If you want to set it when you install Red Hat, you can drop to the shell (F2 from the text-mode installer) and create the filesystems yourself. After that, you can return to the installer, which has the option of not formatting partitions. A sample command would be:

```
mke2fs -s 1 -b 4096 -i 8192 -m 1 /dev/sda1
```

where -s 1 parses superblocks (Linux 2.2 or better), -b 4096 sets the disk block size to 4K, -i 8192 sets bytes per inode to 8K and -m 1 sets reserved block size to 1% of your partition size (appropriate for today's big disks and partitions). —Marc Merlin, marc_bts@valinux.com

I Have No /dev/cdrom and I Must Mount

I erased my /dev/cdrom by mistake and I can't do **mkdir** to replace it because it must be a block mode file. Do you know how I can remount my CD-ROM? —Yves, cloude7@hotmail.com

/dev/cdrom is usually a symbolic link to one of the disk drives detected at boot. Find the device for the CD-ROM with **dmesg | grep CDROM** (e.g., mine says **hdc: YAMAHA CRW8424E, ATAPI CDROM drive**). Then create the link with **cd /dev ln -s hdc cdrom**. However, if you are using an IDE CD-W, then it will be something like **cd /dev ln -s scd0 cdrom** rather than **hdc**. —Keith Trollope, keith@wishing-well.demon.co.uk

Archive Index Issue Table of Contents

Advanced search

New Products

Heather Mead

Issue #88, August 2001

SysOrb 2.0, Quick Restore Data Protection, iXsystems 1U Server and more.

SysOrb 2.0

Solit Solutions ApS has released SysOrb 2.0, a cross-platform network and systems monitoring system that supports Red Hat, Debian, FreeBSD, Windows NT/2000 and Solaris. A web interface provides access to monitoring configuration, real-time and historical data from the database backend. Alerts can be sent via SMS, e-mail and pager when problems are discovered. A free download and license are available at http://www.sysorb.com/.

Contact: Solit Solutions ApS, Maglebjergvej 5D, DK - 2800 Lyngby, +45-45-880-888, sysorb@sysorb.com, http://www.sysorb.com/, http://www.solit.dk/.

Quick Restore Data Protection

Quick Restore version 2.7.4 is data protection software that provides data backup and recovery for heterogeneous networks. Based on the Network Data Management Protocol (NDMP), Quick Restore allows local or remote backup of NAS systems to tape drives and libraries. New features include broadened platform support for easier operation, extended firewall support, improved DLT tape format control and support for newly available tape libraries.

Contact: Workstation Solutions, Five Overlook Drive, Amherst, New Hampshire 03031, 800-487-0080 (toll-free), info@worksta.com, http://www.worksta.com/.

iXsystems 1U Server

The iXtreme 1400 1U rack-optimized internet server is now available from iXsystems (formerly BSDi). The iXtreme 1400 is 15 inches deep and has a footprint less than two-thirds that of the typical 1U server. Eight fans are used to cool the CPUs, disk drives and power supplies, allowing the server to operate in high-temperature areas. The 1400 can be configured with a maximum memory of 4GB, dual Ethernet ports, two EIDE or two SCSI high-speed disks and dual-processor speeds up to 933MHz.

Contact: iXsystems, Inc., 1550 The Alameda, Suite 100, San Jose, California 95126, 866-449-7978 (toll-free), info@ixsystems.net, http://www.ixsystems.net/.

Xmcd 3.0

Xmcd, a CD player application for the X Window System that runs on most UNIX variants, can now be obtained in version 3.0. The 3.0 package includes **xmcd**, an X11/Motif-based CD audio player utility, and cda, a command-line, nongraphical CD audio player. Xmcd supports a wide array of CD-ROM, CD-R, CD-RW and DVD devices, including multidisc changers and some older SCSI-1 drives. It supports CD recognition via Gracenote CDDB and can connect to the CDDB servers on the Internet to get the artist, disc title, song titles and other information about the CD being played. **xmcd** is released under the GPL and can be downloaded from the web site.

Contact: Xmcd, http://www.amb.org.xmcd/.

INTERNETpro Enterprise Stack

The INTERNETpro Enterprise Stack, available from Internet Appliance, Inc., is a managed hosting and security solution that integrates optimized web services software and hardware components into an appliance-linked infrastructure. The entire stack can be centrally managed, administered and provisioned, while delivering availability, scalability and security for service providers and enterprise environments. Solutions include a data center, a site-to-site VPN and a remote access VPN.

Contact: Internet Appliance, Inc., 40515 Encyclopedia Circle, Fremont, California 94538, 877-986-6366 (toll-free), sales@internetappliance.com, http://www.internetappliance.com/.

MindRover: The Europa Project

Loki Software, Inc. released its Linux port of CogniToy's popular *MindRover: The Europa Project* game, a 3-D strategy/programming game that allows players to create autonomous robotic vehicles and compete with them in races, battles and sports. Starting with an empty vehicle, gamers add components such as sensors, weapons and engines. Components are wired together and properties set using a visual programming system. Gamers can share their custom robots over e-mail or through web site competitions.

Contact: Loki Software, Inc., 250 El Camino Real, Suite #100, Tustin, California 92780, info@lokigames.com, http://www.lokigames.com/.

Opera 5.0

Opera 5.0 is the official release version of Opera Software's full-featured web browser built on the Qt development framework. The Opera browser provides fast rendering of pages, instant toggling of image and document settings, extensive drag-and-drop features, multiple windows support and zoom capabilities up to 1,000%. Opera's download size is 1.5MB with Qt dynamically linked or 3MB with Qt statically linked. Support is provided for the following: 128-bit encryption, TLS 1.0, SSL 2 and 3, CSS1 and CSS2, XML, HTML 4.01 and JavaScript 1.3

Contact: Opera Software A/S, Waldemar Thranesgt. 98, 0175 Oslo, Norway, commercial@opera.com, http://www.opera.com/.