Advanced search

Linux Journal Issue #115/November 2003

Open-Source Desktop Publishing: Scribus
LINUX JOURNAL
READERS' CHOICE 2003
GENOME SCIENTISTS REPORT:
How we sequenced the SARS virus in five days
Virtual security zones
Highly available cluster management with OSCAR
Torture test your 1U servers
512-processor NUMA systems
Secure IMAP mail servers
Disaster plans for your web site
Design C++ programs for security
3-D visualization for submarine commanders

HA-OSCAR: the Birth of Highly Available OSCAR

Ibrahim Haddad

Chokchai Leangsuksun

Stephen L. Scott

Issue #115, November 2003

As clusters reach thousands of nodes, eliminating single points of failure becomes critical. The Open Cluster Group's HA-OSCAR is a solution.

The seeds for the Open Cluster Group (OCG) and the resulting Open Source Cluster Application Resources (OSCAR) Project were planted when a number of like-minded individuals had the good fortune to sit together for dinner at a meeting sponsored by the Department of Energy on February 17, 2000. Over dinner, this group discussed the effort involved in deploying the software necessary to build a Beowulf high-performance computing cluster. Although this group agreed that it was simple to wire together commodity computers to build the cluster, they went on to agree that the amount of effort required to install and configure the requisite software stack on a Beowulf cluster was inordinately high. Challenging to some and tedious to all was the consensus description. Thus, the idea to simplify the process was born.

Further fertilization of this idea took place in Oak Ridge, Tennessee, in April that year at a meeting that included industry, academia and research labs. It was at this first formal meeting that the OCG was formed and work was begun on OSCAR, specifically the Beowulf cluster software stack and associated installation process. At this meeting the group agreed on three core principles:

1. The adoption of clusters for mainstream, high-performance computing is inhibited by a lack of well-accepted software stacks that are robust and easy to use by the general user.
2. The OCG embraces the open-source model. As a result, the OSCAR distribution must contain only freely redistributable codes, with a

preference for the inclusion of source code under a Berkeley-style open-source license.
3. The OCG can accomplish its goals through the use of best-practices codes currently available.

Further details about the beginning of OCG and OSCAR can be found in an article by Richard Ferri in the June 2002 issue of *Linux Journal* titled “The OSCAR Revolution”.

Throughout OSCAR's brief history, the group has managed to adhere to these three principles while allowing OSCAR to encompass other forms of software. For example, although the OSCAR distribution itself contains only freely redistributable codes, others are deploying an OSCAR package that, while not a part of the formal OSCAR distribution, may be installed or dropped into an existing OSCAR distribution for installation.

The OSCAR Project continues to contain a mixture of industry and academic/ research members. The overall project is directed by a steering committee elected every two years from the current core organizations. This core list is composed of those actively contributing to project development. The 2003 core organizations include: Bald Guy Software (BGS), Dell, IBM, Intel, MSC.Software, Indiana University, the National Center for Supercomputing Applications (NCSA), Oak Ridge National Laboratory (ORNL) and Université de Sherbrooke (UdS).

In the past year, additional OCG working groups have been created to address other cluster environments. These new groups are working to leverage the technology provided by OSCAR when producing their cluster distributions. The two groups working today are Thin-OSCAR and HA-OSCAR. Thin-OSCAR is headed by the Université de Sherbrooke in Canada and is dedicated to delivering a diskless variant of OSCAR. The HA-OSCAR group is led by the authors of this article and is focused on providing a high-availability version of OSCAR.

HA-OSCAR: Mission, Goals and People

In a July 2001 meeting at Ericsson Research Canada, Ibrahim Haddad made the case for high availability in cluster computing. Initially, the discussion centered on the necessity of high-availability computing for the telecom industry. As the discussions progressed, it became clear that with the anticipated tens of thousands of nodes in high-performance computing (HPC) clusters, high-availability techniques can provide some level of the fault tolerance desired by the HPC community.

Ibrahim's group at Ericsson Research worked primarily alone on the high-availability effort until the recent addition of Dr Chokchai Leangsuksun and his team at Louisiana Tech University and the continued interest in HA-OSCAR by Stephen Scott at ORNL. In 2002, the HA-OSCAR effort was recognized officially by OCG as another working group. The primary goal of the group is to leverage existing OSCAR technology and provide for new high-availability capabilities in OSCAR clusters. The anticipated customers of this technology include the telecom industry and HPC sites.

HA-OSCAR Improvements over OSCAR

HA-OSCAR introduces several enhancements and new features to OSCAR, mainly in areas of availability, scalability and security. Most of these features can be mapped to ITU (International Telecommunication Union), TMN (Telecommunication Management Network) and FCAPS (Fault-management, Configuration, Accounting, Performance and Security). These concepts are widely adopted in the telecom industry to manage its network elements.

Dual Master Nodes and Redundancy

A typical cluster computing architecture consists of several nodes that can provide some degree of availability. However, it normally has a single-head node that is a simplex architecture and prone to single points of failure. The current release of OSCAR falls into this architectural category, which is unsuitable for mission-critical systems as it contains several individual system elements that have no redundancy for a backup or failover. In order to support HA requirements, clustered systems must provide ways to eliminate single points of failure.

Hardware duplication and network redundancy are common techniques utilized for improving the reliability and availability of computer systems. To build an HA-OSCAR cluster system, we first must provide a duplication of the cluster head node. Such an architecture can be implemented in different ways, including active-active, active-warm standby and active-cold standby.

The active-active model enables both performance and availability, because both head nodes simultaneously can provide services. However, its implementation is quite complicated and leads to data inconsistency when failures occur. Active-standby options mostly are adopted solutions. The standby server watches the primary server health and can take over control when it detects an outage. Currently, the active-warm standby configuration is the initial model of choice.

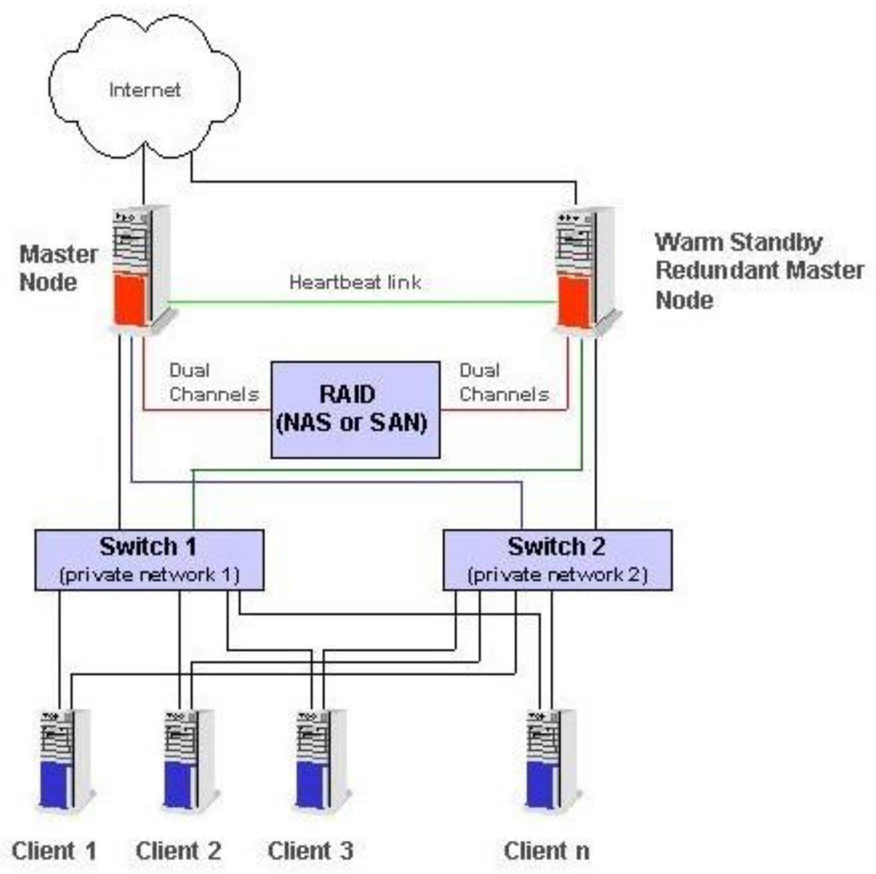


Figure 1. HA-OSCAR Cluster System Architecture

Figure 1 shows the HA-OSCAR cluster system architecture. We experimented with and planned to incorporate Linux Virtual Server and Heartbeat mechanisms into our initial active-hot standby HA-OSCAR distribution. Now, we plan to extend our initial architecture to support active-active HA after we release the hot-standby distribution. The active-active architecture can better utilize resources, because both head nodes can be simultaneously active to provide services. The dual master nodes then can run redundant DHCP, NTP, TFTP, NFS and SNMP servers. In the event of a head node outage, all functions provided by that node failover to the second redundant head node and are served at a reduced performance rate (in theory, 50% at the peak or busy hours).

Another HA functionality to support in HA-OSCAR is providing a high-availability network using redundant Ethernet ports on every machine. In addition, duplicate switching fabrics (network switches, cables, etc.) are used for the entire network configuration. This enables every node in the cluster to be present on two or more data paths within its networks. Backed with this Ethernet redundancy, the cluster achieves higher network availability. Furthermore, when both networks are up, improved communication performance may be achieved by using techniques such as channel bonding of messages across the redundant communication paths.

HA-OSCAR aims to reuse features from other implementations and existing projects, including the High-Availability Linux, Kimberlite and Linux Virtual Server projects. We then plan to contribute the added enhancements and functionalities back to the community.

Support for Dual IP Stack (IPv4 and IPv6)

IPv6 is the next-generation protocol designed by IETF to replace the current version of the Internet Protocol, IPv4. Most of today's Internet uses IPv4, which has been remarkably resilient in spite of its age, but it is beginning to have problems. Most importantly, there is a growing shortage of IPv4 addresses, which are needed by all new devices connecting to the Internet. As a result, IETF defined IPv6 to fix the problems in IPv4 and to add many improvements for the future Internet. These improvements come in different areas, such as routing, autoconfiguration, security, QoS and mobility.

HA-OSCAR has support for IPv6 activated by default. Most of the ISPs and telecom companies already are experimenting with co-existence schemes for IPv4 and IPv6. All cluster nodes installed with HA-OSCAR provide support for IPv6 and basic IPv6 capabilities compiled directly in the network utilities and binaries.

Recovering from Corrupted Disks

OSCAR assumes the client node disks on which it is installing are faultless. But, this is not always the case; some nodes may have corrupted disks. HA-OSCAR considers this issue and does not assume that all disks on all nodes are a good installation base. To this end, we support special scripts in our installs and software RAID in the kernel, in parallel with developing the necessary set of scripts needed to synchronize disk contents. As such, if a disk fails, data is not lost. In addition, our installation wizard first tries to fix the corrupted disk. HA-OSCAR also supports synchronous operation, disk removal and disk insertion. In addition, HA-OSCAR supports software RAID by default. By enabling software RAID, clusters powered by HA-OSCAR have increased data redundancy and better performance.

Linux Virtual Server (LVS) and Heartbeat Integration

We already described the HA-OSCAR hardware architecture. A key enhancement is the addition of dual head nodes that provide a backup head node for a failover in case of a primary head node outage. However, a hardware redundancy solution alone is not sufficient to archive HA unless detection and recovery mechanisms are incorporated.

Few existing solutions provide outage detection and failover. We have evaluated and selected a failover LVS. The solution includes LVS, Linux Director Dæmon (ldirectord), Heartbeat and Coda. Linux Virtual Server is a software tool that directs network connections to multiple servers that share a workload. ldirectord is a standalone dæmon to monitor services. Heartbeat provides a primary node outage detection and failover mechanism through serial line and UDP connectivity. Coda is a fault-tolerant distributed filesystem. This solution not only provides HA capability, but load balancing as well. However, additional LVS services must be enhanced in HA-OSCAR, including SIP, PBS and Web services. An external Heartbeat (eHB) mechanism to a fault management system also has been added. eHB is a precaution in case of a total outage (for example, double head node failures) from which the fault management detects, raises an alarm and sends a page to a system administrator.

Cluster-Wide Security

OSCAR currently is installed on clusters deployed mostly on private networks, where security is not a major concern. That is because these clusters are not connected to any networks outside the lab boundaries. However, when HA-OSCAR is deployed on clusters connected to the Internet, security is vital. Security is a major concern for both OSCAR and HA-OSCAR not only because a hacker might access the cluster and the data sitting on it, but also because a malicious hacker also might disrupt the normal workings of the system and its availability.

Many security solutions exist, ranging from external solutions (firewalls) to internal solutions (integrity-checking software). Unfortunately, all of them are based on a single node approach and lack a homogeneous view of the cluster. Most of the time, administrators end up installing, patching, integrating and managing several security solutions. The increased management difficulty soon leads to decreased security, as interoperability issues increase with updates of the heterogeneous pieces.

Consequently, the Distributed Security Infrastructure (DSI) was initiated as an open-source project to provide an adequate security solution for carrier-grade clustered servers. DSI is a security framework that provides applications running on clustered systems with distributed mechanisms for access control, authentication, confidentiality and integrity of communications. It also provides auditing services with process-level granularity.

Therefore, HA-OSCAR can be more successful with telecom and other mission-critical sectors if it supports advanced security features. For this reason, HA-OSCAR adopted DSI from Ericsson.

Dynamic Addition and Removal of Nodes

HA-OSCAR supports a mechanism that allows users to add and remove nodes from the cluster dynamically, in a transparent fashion, without affecting either the end-user experience or the running applications. Two open-source projects provide similar functionalities: Eddie, an Ericsson open-source initiative, and LVS. We currently are investigating the best mechanism and will implement it in HA-OSCAR. Our goal is to ensure that adding nodes to accommodate higher traffic or removing nodes for service purposes is a seamless operation and does not affect service availability.

Linux Kernel

The subject of which kernel to adopt came in addition to the decision about whether to patch the HA-OSCAR kernel ourselves or try to have our patches accepted by the mainstream kernel tree. We decided to use the latest stable 2.4 kernel and submit the patches we create to the kernel mailing list. We are trying to provide a simplified kernel building tool for these HA-OSCAR users. Users can recompile based on their local configurations.

Support for Network Filesystems

A network/distributed filesystem is an essential component for building clusters. A number of open-source projects aim to provide network filesystems for Linux clusters. Based on our previous research and lab testing, we ascertained that a different networked filesystem may be required, depending on the type of applications being run on the cluster. For instance, using the parallel virtual file system (PVFS) offers the advantage of high I/O performance for large files on a streaming video and audio server. On the other hand, sharing configuration files among cluster nodes can be achieved using the NFS without the need for high I/O. If it is desirable to maintain high availability and support storage area networks (SANs), OpenGFS, with its journaling capability, can handle such a task. Therefore, HA-OSCAR is working to support all the possible network filesystems that can be used in target environments.

Fast Cluster Setup

One important factor to consider is the time it takes to build, boot and have the cluster ready to service requests. This is not a major issue for small clusters, but as we move to large installations of 256 nodes and higher, having the capabilities of installing and booting all cluster nodes in an automated and timely manner becomes an asset. HA-OSCAR is considering implementing hierarchical clustering by dividing the cluster into multiple zones. This type of experimentation also can be helpful in identifying the slow processes in the system installation procedure, which allows us to bring it up to speed.

LinuxBIOS, for instance, can be included in place of the normal BIOS—with a little bit of hardware initialization and a compressed Linux kernel that can be booted from a cold start—to achieve faster startup times. The upcoming OSCAR release uses multicast technology, which was tested on about 500 nodes, to speed up install times and return impressive numbers. HA-OSCAR plans to adopt this method as the base install mechanism and improve on it.

Selective Installs

Similar to the base OSCAR installation, users of HA-OSCAR have the freedom of deciding which application packages to install. By default, HA-OSCAR automatically installs the essential parts to build a cluster and then prompts the user to select the applications they want.

The installation procedure takes into consideration any existing configuration and the packages already installed on the node. Some packages are sensitive to certain system libraries, such as glibc. Users should be aware that installing HA-OSCAR may require them to upgrade their systems based on such dependencies. In the same manner, a de-installation procedure is provided to clean up every HA-OSCAR-specific addition without disturbing the system integrity. This option is important for users who want to test only HA-OSCAR.

It is also worth mentioning that package install and uninstall options are available in the base OSCAR release since v2.0, and a newly enhanced version is coming out soon.

Network Upgrades

HA-OSCAR plans to investigate the possibility of providing mechanisms for selective network software upgrades without bringing down the system. Network upgrades are an interesting way of patching an operating system and its applications. As an example, most Linux distributions now come with an automatic network upgrade that eases this tedious administrative task. In the case of administrating a large cluster, HA-OSCAR users can use such a feature to upgrade their application version seamlessly, without service interruption. Network upgrade simplifies cluster administration and promotes better software management across all computing nodes.

In addition, HA-OSCAR provides a tool that allows users to change the configuration of the cluster at runtime by using a tool somewhat similar to LinuxConf. This is still a basic idea that will be investigated further in the near future.

Backups, Restores and Disaster Recovery

Generally, one cannot trust a computing system if there is no backup or recovery mechanism. For mission-critical applications, including telecom applications, it is important to be able to recover from any software or hardware failure. Thus, providing efficient backup and recovery mechanisms is an essential part of any HA system.

In case a disaster occurs, recovery ability and speed are critical. Every time HA-OSCAR is completely re-installed or the kernel updated, ghost images of before and after are saved in a designated location on a backup server and tape. Ghost for Unix takes a snapshot of an old and new kernel, gzips it and sends the image to the secondary head node as well as to a predefined disaster recovery site. Important data as well as application and configuration files also can be included in the ghost image. Normally, tape backup schedules include nightly snapshots for incremental images and weekly snapshots for full images. For faster recovery and highly reliable backups, ghost imaging, file journaling and data replication are implemented.

Supporting Web Clusters

One goal of HA-OSCAR is to be deployed optionally as a Web server cluster providing highly available Web services to a large number of clients. One step toward this goal is to set up a Web server, such as Apache, on every node; Apache can be one of the packages copied to the nodes. Then, a single IP interface is provided for the cluster, possibly using LVS Direct Routing, because it has proven to be the scalable implementation.

Support for Asynchronous Process Execution

Telecom applications must be built to face extreme or unplanned conditions of execution. Even in typical real-life situations, subscribers are putting a lot of pressure on carriers because of their high expectations regarding system performance and availability. Customers do not expect these applications to fail or their phone requests to be delayed beyond a typical threshold. This is increasingly true as telecom applications are providing additional services, some requiring real-time characteristics.

Carrier-grade applications must be designed with these subscribers' constraints in mind, taking into account the cost of software maintenance and upgrades, service availability and scalability. Complex distributed software demands a specific programming paradigm. It has been proven over the years that complex system interfaces tend to increase the time to debug and the probability of application failure.

AEM (asynchronous event mechanism) provides an event-driven methodology of development in order to provide robust applications with a mechanism that allows reacting quickly to system events by means of user-space callbacks. In the AEM implementation, the kernel plays a major role in handling events and increases the reliability of applications. For this reason, AEM provides a flexible solution for application designers, supplying an extensible framework that allows new functionalities to be added at runtime, without rebooting the system or restarting applications. In order to reach carrier-grade requirements, HA-OSCAR plans to supply efficient support for asynchronous events.

Resources

AEM: www.linux.ericsson.ca/aem

CODA: www.coda.cs.cmu.edu

DSI: www.linux.ericsson.ca/dsi

Eddie: eddie.sourceforge.net

FCAPS: www.iec.org/online/tutorials/ems/topic03.html

g4u: www.feyrer.de/g4u

GFS: www.globalfilesystem.org

GPL: www.gnu.org/copyleft/gpl.html

Linux HA: linux-ha.org

LVS: www.linuxvirtualserver.org

OCG: www.OpenClusterGroup.org

Open System Lab: www.linux.ericsson.ca

OSCAR: www.OpenClusterGroup.org/OSCAR

The OSCAR Revolution: /article/5559

PVFS: parlweb.parl.clemson.edu/pvfs

Ibrahim Haddad (Ibrahim.Haddad@Ericsson.com) is a researcher at the Open System Lab, Ericsson Research Corporate Unit. He is coauthor, along with

Richard Peterson, of the *Red Hat Linux Pocket Administrator* from McGraw-Hill, to be published in September 2003.

Chokchai Leangsuksun (box@latech.edu) is an associate professor of computer science at the Center for Entrepreneurship and Information Technology (CEnIT) at Louisiana Tech University. Prior to his academic career, he spent seven years in R&D with Lucent Technologies in system reliability and high-availability computing and telecommunication systems.

Stephen L. Scott (scottsl@ornl.gov) is a senior research scientist in the Computer Science and Mathematics Division of Oak Ridge National Laboratory, US. He is a founding member of OCG and presently is version 2 release manager. Previously he was the working group chair of the OSCAR Project.

Archive Index Issue Table of Contents

Advanced search

Cluster Hardware Torture Tests

John Goebel

Issue #115, November 2003

Designing a thorough hardware test plan now can save you time, money and machine room wiring later.

Without stable hardware, any program will fail. The frustration and expense of supporting bad hardware can drain an organization, delay progress and frustrate everyone involved. At Stanford Linear Accelerator Center (SLAC), we have created a testing method that helps our group, SLAC Computer Services (SCS), weed out potentially bad hardware and purchase the best hardware at the best possible cost. Commodity hardware changes often, so new evaluations happen each time we purchase systems. We do minor re-evaluations for revised systems for our clusters about twice a year. This general framework helps SCS perform accurate, efficient evaluations.

This article outlines our computer testing methods and system acceptance criteria. We expanded our basic ideas to other evaluations, such as storage. The methods outlined here help us choose hardware that is much more stable and supportable than our previous purchases. We have found that commodity hardware ranges in quality, so systematic methods and tools for hardware evaluation are necessary. This article is based on one instance of a hardware purchase, but the guidelines apply to the general problem of purchasing commodity computer systems for production computational work.

Defining System Requirements

Maintaining system homogeneity in a growing cluster environment is difficult, as the hardware available to build systems changes often. This has the negative effect of adding complexity in management, software support for new hardware and system stability. Furthermore, introducing new hardware can introduce new hardware bugs. To constrain change and efficiently manage our systems, SCS developed a number of tools and requirements to enable an easy fit for new hardware into our management and computing framework. We reduced the features to the minimum that would fit our management infrastructure and still produce valid results with our code. This is our list of requirements:

- One rack unit (1U) case with mounting rails for a 19" rack.
- At least two Intel Pentium III CPUs at 1GHz or greater.
- At least 1GB of ECC memory for every two CPUs.
- 100MB Ethernet interface with PXE support on the network card and in the BIOS.
- Serial console support with BIOS-level access support.
- One 9GB or larger system disk, 7,200 RPM or greater.
- All systems must be FCC- and UL-compliant.

Developing a requirements list was one of the first steps of our hardware evaluation project. Listing only must-haves as opposed to nice-to-haves grounded the group. It also slowed feature creep, useless additions to hardware and vendor-specific methods for doing a task. This simple requirement culled the field of possible vendors and reduced the tendency to add complexity where none was needed. Through this simple list, we chose 11 vendors to participate in our test/bid process. A few vendors proposed more than one model, so a total of 13 models were evaluated.

Starting Our System Testing

The 11 vendors we chose ranged from large system builders to small screwdriver shops. The two criteria for participating in the evaluation were to meet the list of basic requirements and send three systems for testing. We wanted the test systems for 90 days. In many cases, we did not need the systems that long, but it's good to have the time to investigate the hardware thoroughly.

For each system evaluation, two of the three systems were racked, and the third was placed on a table for visual inspection and testing. The systems on the tables had their lids removed and were photographed digitally. Later, the tabled systems were used for the power and cooling tests and the visual inspection. The other two systems were integrated into a rack in the same manner as all our clustered systems, but they did not join the pool of production systems. Some systems had unique physical sizing and racking restrictions that prevented us from using them.

Each model of system had a score sheet. The score sheets were posted on our working group's Web page. Each problem was noted on the Web site, and we tried to contact the vendor to resolve any issues. In this way we tested both the system and the vendor's willingness to work with us and fix problems.

We had a variety of experiences with all the systems evaluated. Some vendors simply shipped us another model, and some worked through the problem with us. Others responded that it was not a problem, and one or two ignored us. This quickly narrowed the systems that we considered manageable.

Throughout the period of testing, if a system was not completing a specific task, it was running hardware testing scripts or run-in scripts. Each system did run-in for at least 30 days. No vendor does run-in for more than 72 hours, and this allowed us to see failures over the long term. Other labs reported they also saw problems over long testing cycles.

In general, we wanted to evaluate a number of aspects of all the systems: the quality of physical engineering, operation, stability and system performance. Finally, we evaluated each vendor's contract, support and responsiveness.

Physical Inspection

The systems placed on the table were evaluated based on several criteria: quality of construction, physical design, accessibility, quality of power supply and cooling design. To start, the systems varied greatly in quality of construction. We found bent-over, jammed ribbon cables, blocked airflow, flexible cases and cheap, multiscrew access points that were unbelievably bad for a professional product. We found poor design decisions, including a power switch offset in the back of a system that was nearly inaccessible once the system was racked. On the positive side, we came across a few well-engineered systems.

Our evaluation included quality of airflow and cooling, rackability, size/weight and system layout. Features such as drive bays at the front also would be noted. Airflow is a big problem with hot x86 CPUs, especially in restricted spaces such as a 1U-rack system. Some systems had blocked airflow or little to no circulation. Heat can cause instability in systems and reduce operational lifetimes, so good airflow is critical.

Rigidity of the case, no sharp edges, how the system fits together and cabling also belong in this category. These might seem small, uninteresting factors until you get cut by a system case or have a large percentage of “dead on arrivals”, because the systems were mishandled by the shipper and the cases were too weak to take the abuse. We have to use these systems for a number of years; a simple yet glaring problem is a pain and potentially expensive to maintain.

Tool-less access should be a standard on all clustered systems. When you have thousands of systems, you are always servicing some of them. To keep the cost of that service low, parts should be quickly and easily replaceable. Unscrewing and screwing six to eight tiny machine screws slows down access to the hardware. Parts that fit so one part does not have to come out to get to another part and that provide easy access to drives are pluses. Some features we did not ask for, like keyboard and monitor connections on the front of the case, are fine but not really necessary.

We tested the quality of the power supply using a Dranetz-BMI Power Quality Analyzer (see Sidebar). Power correction often is noted in the literature for a system, but we have seen radically different measurements relative to the published number. For example, one power supply, with a published power factor correction of .96, actually had a .49 correction. This can have terrible consequences when multiplied by 512 systems. We tested the systems at idle and under heavy load. The range of quality was dramatic and an important factor in choosing a manageable system.

The physical inspection, features, cooling and power-supply quality tests weeded out a number of systems early in the process. Eliminating these right away reduced the number of systems that needed extensive testing, thereby reducing the amount of time spent on testing overall. System engineering, design and quality of parts ranged broadly.

Measuring Power Supply Quality

Power supplies often come with inaccurate quality claims. We have experienced a number of problems due to poor-quality power supplies, so we test every system's power supply. For an accurate measurement, a Power Quality Analyzer is used to measure systems at idle and under heavy load.

Prior to employing our test methods, SCS built a cluster with a poor power supply and experienced a range of problems. One of the most expensive problems was current being mismanaged by the power supply. Three phase distribution power systems often are designed with the assumption of nicely balanced loads across the three phases, which results in the neutral current approaching zero. The resulting designs usually used the same gauge wiring on the neutral as on the supply.

Unfortunately, low-quality power supplies generate large third harmonic currents, which are additive in the the neutral line. The potential result of this is neutral current loads in excess of the rated capacity of the wiring, to say nothing of the transformers that were not rated for such loading. And, the neutral cannot be fused by code, so it was possible to exceed the neutral wiring capacity without tripping a breaker on the supply lines. This required a derating on all parts of the infrastructure to remain within spec. Derating is expensive, time consuming, and the cluster cannot be used during that time.

Thanks to Gary Buhrmaster for help on this Sidebar.

Testing with Software

Run-in (often called burn-in) is the process manufacturers use to stress test systems to find faulty hardware before they put them in the field. A number of open-source run-in programs are available. One common program is the Cerberus Test Control System sourceforge.net/projects/va-ctcs. It is a series of tests and configurable wrapper scripts originally designed for VA Linux Systems' manufacturing. Cerberus is ideal for run-in tests, but we also developed specific tests based on our knowledge of system faults. We were successful in crashing systems with our scripts more often than when using a more general tool. Testing by using programs developed from system work experience can be more effective than using Cerberus alone, so consider creating a repository of testing tools.

Read the instructions carefully, and understand that run-in programs can damage a system; you assume the risk by running Cerberus. Also, there are a number of software knobs to turn, so consider what you are doing before you launch the program. But if you are going to build a cluster, you need to test system stability, and run-in scripts are designed to test exactly that quality.

At the time that we were testing systems, two members of our group wrote their own run-in scripts, based on some of the problems we have seen in our production systems. Whereas benchmarks try to measure system performance and often have sophisticated methods, run-in scripts are simple processes. A system is put under load and either passes or fails. A failure crashes the system or reports an error; a pass often does not report information. We also ran production code, which uncovered more problems. Production code always should be run whenever possible. For instance, one of the systems that passed the initial design inspection tests with flying colors failed under heavy load.

Performance

A plethora of benchmark programs is available. The best benchmark is to run the code that will be used in production, just as it is good to run production code during run-in. This is not always possible, so a standard set of benchmarks is a decent alternative. Also, standard benchmarks establish a relative performance value between systems, which is good information. We do not expect a dramatic performance difference in commodity chipsets and CPUs. Performance differences exist, however, when different chipsets and motherboard combinations are involved, which was the case in this testing trail.

We also wrote a wrapper to a number of standard benchmarking tools and packaged it into a tool called HEPIX-Comp (High Energy Physics—Compute). It is a convenience tool, not a benchmark program itself. It allows a simple `make server` or `make network` to measure different aspects of a system. For example, HEPIX-Comp is a wrapper for the following tools (among others): Bonnie++, IOZone, Netpipe, Linpack, NFS Connectathon package and streams.

Understanding the character of the code that runs on the system is paramount to evaluating with standard benchmarking. For example, if you are network-constrained, a fast front-side bus is less important than network bandwidth or latency. These are good benchmarks that measure different aspects of a system. Streams, for example, measure the I/O memory subsystem throughput, which is an important measure for systems with hierarchical memory architectures. Bonnie++ measures different types of read/write combinations for I/O performance.

Many vendors report performance that gives the best possible picture. For example, sequential writes as an I/O performance measure is pretty rosy compared to random, small writes, which are closer to reality for us. Having a standardized test suite run under the Linux installation that is used in production establishes a baseline measurement. If the system is tuned for one benchmark, it might perform the benchmark well at the expense of another system performance factor. For example, systems tuned for large block sequential writes hurt small random writes. A baseline benchmark suite at least shows an apples-to-apples comparison, although not the potentially best performance. So, this is by no means a perfect system, but rather one more data point in an evaluation that characterizes system performance.

All the data was collected and placed on internal Web pages created for the evaluation and shared among the group. We met once a week and reported on the progress of the testing. After our engineering tests were complete, we chose a system.

Non-Engineer Work

Non-engineering factors (contractual agreements, warranties and terms) are critical to the success of bringing in new systems for production work. The warranty terms and length affects the long-term cost of system support. Another consideration is the financial health of the vendor company. A warranty does little good if the vendor is not around to honor it.

Also crucial is the acceptance criteria, although seldom talked about until too late. These criteria determine the point in the deployment when the vendor is finished and the organization is willing to accept the systems. This point should be made in writing in your purchase order. If the vendor drops the system off at the curb and later, during the rollout period, some hardware-related problem surfaces, you need to be within your rights to ask the vendor to fix the system problem or remove the system. On the vendor side, a clear separation between what constitutes a hardware or software problem needs to be made. Often a vendor has to work with the client to determine the nature of the problem, so that costs need to be built in to the price of the system.

The Result

The success of the method outlined in this article is apparent in how much easier, and therefore cheaper, it is to run the systems we chose after doing this extensive evaluation. We have other systems that we purchased without doing the qualification outlined here. We have had fewer problems after the better evaluation, and we are able to get more work done in other areas, such as tool writing and infrastructure development. And, we are less frustrated, as are our researchers, with good hardware in production.

John Goebel works at the Stanford Linear Accelerator Center (SLAC) in Menlo Park, California. He is part of the SLAC Computing Services (High-Performance Group), supporting a high-energy physics project for a worldwide research community.

Archive Index Issue Table of Contents

Advanced search

Sequencing the SARS Virus

Martin Krzywinski

Yaron Butterfield

Issue #115, November 2003

At 1AM on April 7, 2003, an isolate of the SARS virus arrived at the Michael Smith Genome Sciences Centre. Five days later, the lab published the virus sequence for the first time.

In April 2003, we at the Genome Sciences Centre (GSC) publicly released the first complete sequence assembly of the coronavirus now believed to be the cause of Severe Acute Respiratory Syndrome (SARS). The GSC has been using Linux for all of its analysis, storage and network infrastructure since its inception in 1999. The sequence data from the SARS Project was stored, processed and publicly distributed from a number of Linux servers, from the capable and slim IBM x330 to the behemoth eight-way Xeon x440. Linux has provided a flexible infrastructure allowing us to automate nearly every process in our sequencing pipeline. With the support of the Linux community, by way of newsgroups, Web articles and HOWTOs, we have been able to leverage commodity and mid-range hardware in an incredibly cost-efficient manner.

Since the first documented incidence of SARS on November 16, 2002, the virus has been responsible for a total of 8,458 cases reported in China (92%), Canada (3%), Singapore (2%) and the United States (1%), as well as in more than 25 other countries. SARS mortality rate is roughly 5–10% and as high as 50% in people older than 60. As of June 24, 2003, SARS has claimed 807 lives and has had a profoundly negative impact on the economies of the affected regions—China alone stands to lose billions of dollars in revenue from tourism and taxation.

On March 27, 2003, the director of our Centre, Marco Marra, and our project leader, Caroline Astell, decided to sequence the SARS coronavirus. At 1AM on April 7, 2003, approximately 50ng of genetic material from the Tor2 isolate of the pathogen, derived from a patient in Toronto, Canada, arrived from the

Level 4 National Microbiology Lab in Winnipeg, Canada. Five days later, on April 12, 2003, our 29,751 base assembly of the sequence of the Tor2 isolate (Tor2/SARS) of the coronavirus was posted to a Zope/Plone page on our Apache server for public access. A few days later, the sequence of the Urbani isolate was posted by the (Centers for Disease Control) CDC in Atlanta, Georgia.

Biology Goes Boom

Before the 1990s, technology to collect large amounts of sequence information rapidly did not exist. The Human Genome Project (HGP) began in 1991, and by 1999 only 15% of the sequence had been collected. However, thanks to new methods, which were developed during the 1990s, the HGP turned sharply toward completion. By mid-2000, 90% of the human sequence was available, and currently the genome sequence essentially is complete. Data from sequencing projects like HGP is stored and publicly accessible through NCBI's Genbank.

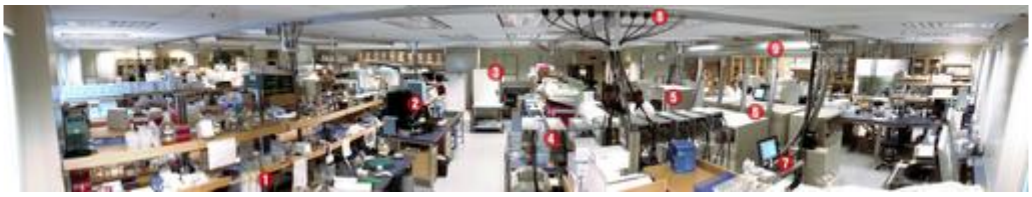


Figure 1. A panorama of our sequencing lab: 1) barcodes representing procedures, 2) the Tango liquid handling platform, 3) –112°F freezers, 4) power supplies for thermocyclers, 5) ABI 3730XL sequencers, 6) ABI 3700 sequencers, 7) x330 sequencer control cluster, 8) network/power connections and 9) vent ducts for sequencers.

During its first ten years of operation (1982–1992), Genbank collected just over 100MB of sequence in 80,000 records. During the next decade (1992–2002) Genbank's rate of growth skyrocketed, and the database grew to 29GB—ten times the size of the human genome—in 22 million records. Genbank receives on the order of 10,000 sequence records each day from sequencing labs across the world. One of these labs is the GSC, which on April 13, 2003, deposited the sequence of Tor2/SARS to Genbank. To see how Linux was involved in the process leading to the submission of sequence gi:29826276, we need to go back to the beginning.

0–18TB in Three Years

In June 1999, the lab consisted of six beige-box computers and just as many people. The central file server (2xP3-400, 512MB of RAM, Red Hat 5.2 and 2.0.36 kernel) was serving out three RAID-0 18GB SCSI disks using a DPT IV card. Another 50GB of software RAID was exported by a second machine (P3-400). With three other Linux clients and a Microsoft Windows NT station, these machines were on the BC Cancer Agency (BCCA) network.

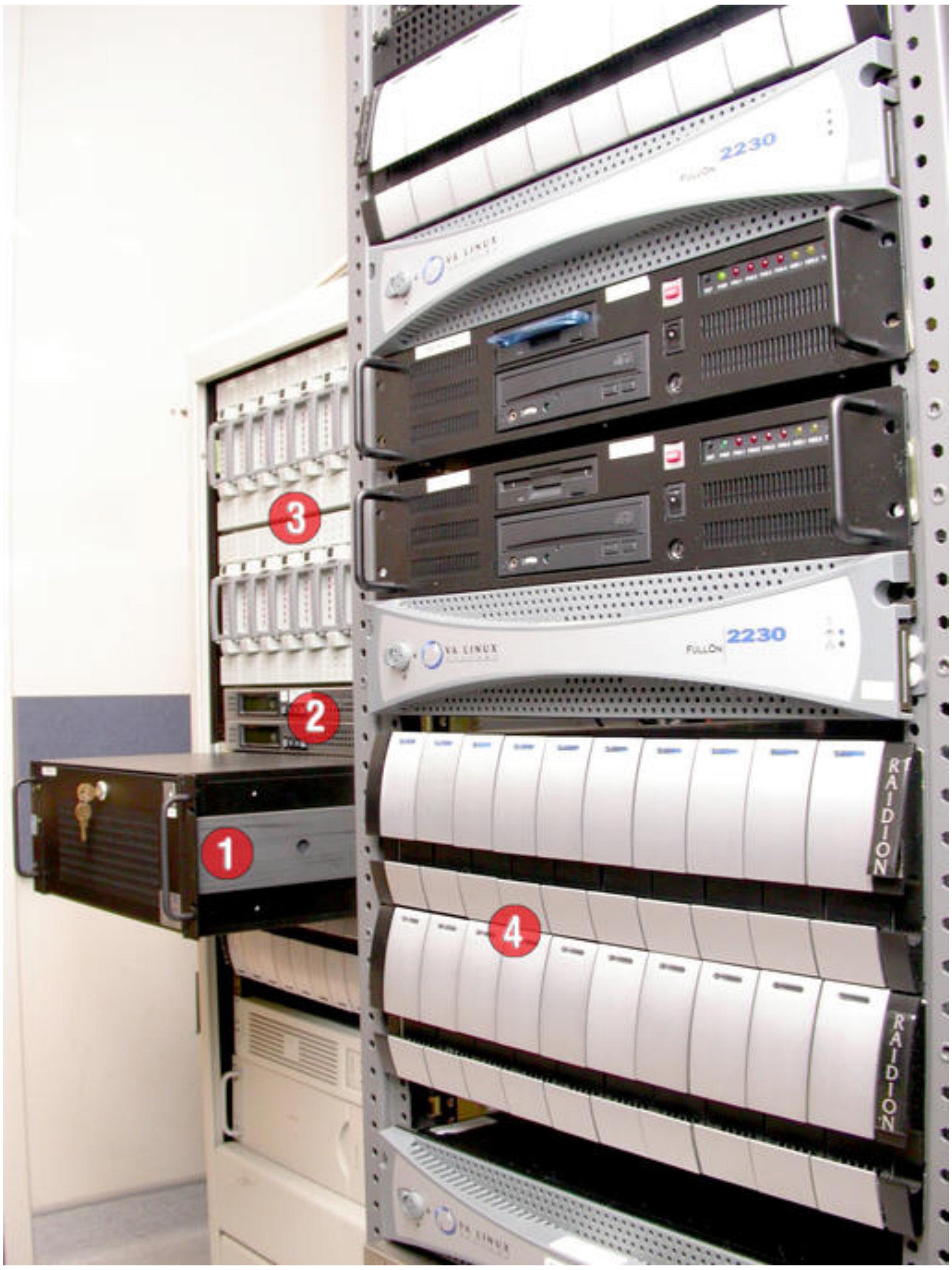


Figure 2. First-generation server hardware: 1) VA Linux VAR900 2xXeon-500 exporting 1TB, 2) Raidion.u2w RAID controllers, 3) 2x8x36GB SCSI disks and 4) VA Linux 2230s and 3x10x72GB SCSI disks.

The timing of our beginnings worked to our advantage. Like all research labs, we needed to share disks, distribute processes, compile software and store and munge data. In other words, all the things at which UNIX excels. Had we started 2–3 years earlier, adopting the fledgling Linux would have been difficult. It's likely that, instead of now relegating inexpensive old PCs to office or less-intensive network tasks, we would be trying to maximize return on our substantial investment of aging Sun servers. Fortunately, it turned out that it was possible to buy the relatively inexpensive PCs, install Linux and have a robust, flexible and incredibly cost-effective UNIX environment. Thanks to

Linux, it was no longer necessary to spend an entire salary on a UNIX workstation.

It was a good time to choose Linux. The 2.0 kernel was rock solid; the NFS server was stabilizing, and a choice of full-featured desktop environments was available. We were able to download or compile the essential toolbox for bioinformatics analysis, such as the open-source workhorses of the HGP: BLAST (sequence comparison), Phred (base calling of traces produced by sequencers), Phrap (sequence assembly) and Consed (visualization of sequence assemblies), as well as various sequence and protein databases. Of course, Perl filled in any cracks. Our cost of entry into getting computational work done was low, and we could spend grant funds more efficiently to expand the lab (Figure 1).

Linux Catches SARS

In the fall of 1999, we received our first DNA sequencer, the MegaBACE 1000 (Figure 6). A sequencer determines the specific base sequence of a DNA sample, though technology currently is limited to determining only 500–800 bases accurately at a time. This read length is much shorter than the size of even the smallest genomes (Tor2/SARS is 30,000 bases in size). Consequently, sequencers simultaneously process 96 samples at a time, and some can be loaded with multiple 96- or 384-well plates.

The MegaBACE is a SCSI device, and the Applied Biosystems (ABI) 3700 and 3730XL sequencers (Figure 6) are controlled through a serial interface and send their data across an Ethernet connection. Although these sequencers acquire large amounts of data in an automated fashion, their software is a point-and-click Windows application. The ABI machines stream their data to a bundled local Oracle database. A UNIX-based control application would revolutionize the deployment of these machines, particularly in large labs. We already have reduced the maintenance complexity of the 3700s by deploying the IBM x330s to replace the original PCs that shipped with the sequencers (Figure 6). Integrating the Windows sequencing platform into a Linux network was the perfect job for smbmount, rsync, Perl and Apache. At the end of each sequence run, the operator triggers a Web-controlled data mirroring process to copy any new data onto the network disks.

After mirroring, the files are first converted from their proprietary format, which encodes the raw signal trace, to the actual bases and their associated quality measure and then are stored in a MySQL database (3.23.55max). Thus far we have collected about 2 million sequencing reads, or about 1TB of raw sequence data.

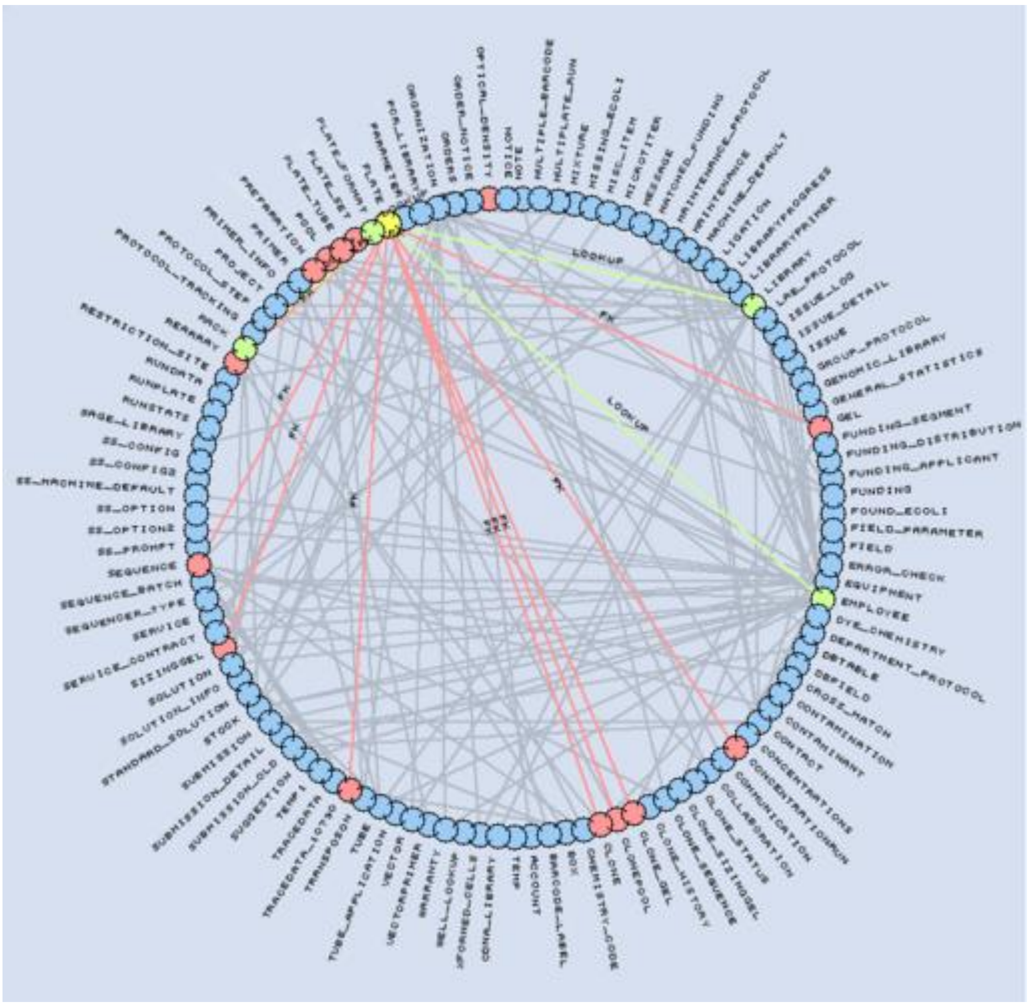
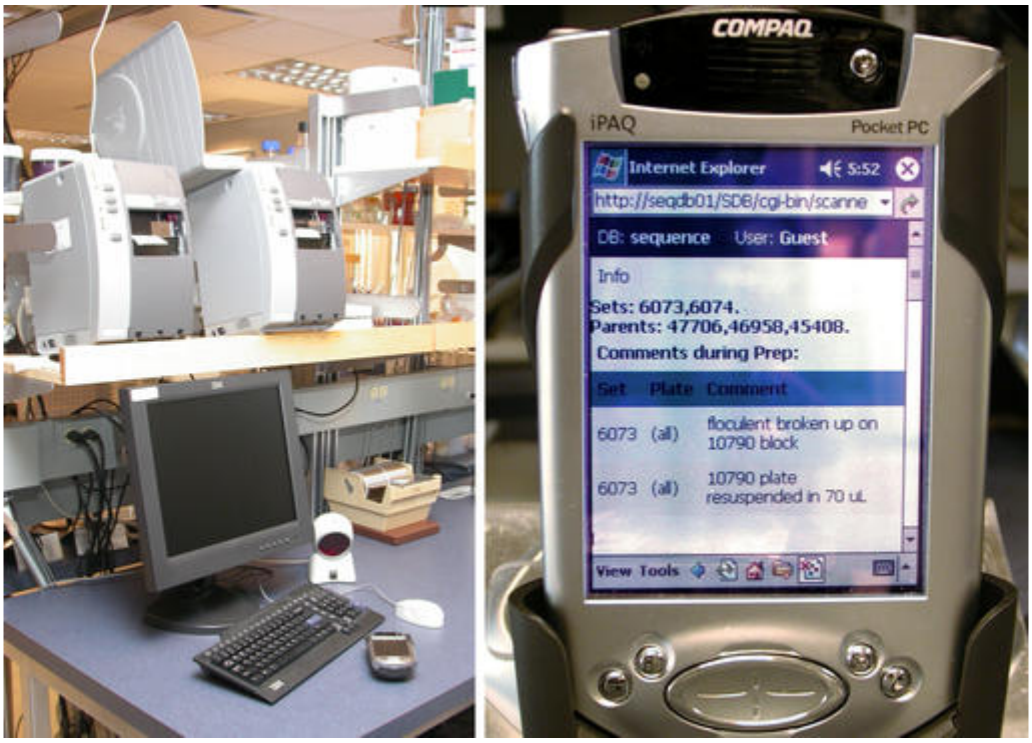


Figure 3. Representation of our LIMS schema. The Plate table (yellow) references four tables (green) and is referenced by 14 tables (red).

The MySQL Laboratory Information Management System (LIMS) database is central to our sequencing process. Its schema contains 115 tables, 1,171 fields and 195 foreign keys. The database tracks all reagents, equipment, processes and reactions performed in the lab. We circumvent MySQL's lack for native foreign key support by using application logic and a specific field naming convention. Foreign keys are named FKTYPE_TABLE__FIELD, indicating that they point to TABLE_FIELD in the table TABLE. The optional TYPE part of the foreign key name is used to support multiple keys to the same TABLE_FIELD.



http://seqdb01/SDB/cgi-bin/scanne

DB: sequence User: Guest

Info

Sets: 6073,6074.
Parents: 47706,46958,45408.
Comments during Prep:

Set	Plate	Comment
6073	(all)	floculent broken up on 10790 block
6073	(all)	10790 plate resuspended in 70 uL

Figure 4. Barcodes are printed using networked Zebra printers (left). iPAQs provide a mobile interface into the LIMS (right).

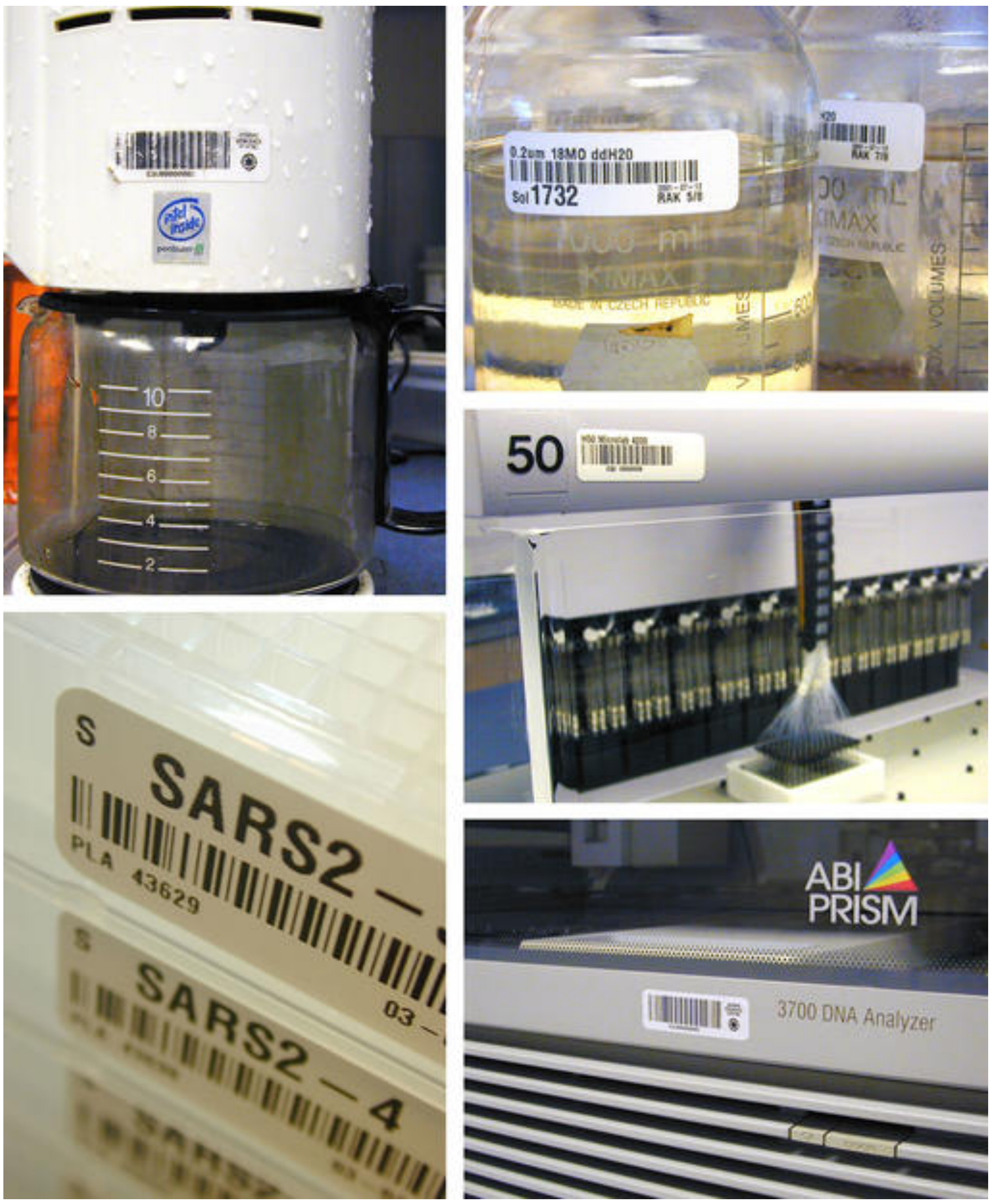


Figure 5. Nearly everything in the lab is barcoded.

Lab technologists interact with the LIMS database using Wi-Fi Compaq iPAQs outfitted with barcode scanners (Figure 4). The iPAQs connect to our internal Apache Web server powering a suite of mod_perl scripts. Objects such as solutions, plates and equipment are barcoded (Figure 5). Barcodes are printed on the networked Zebra S600/96XiIII barcode printers (Figure 4) fed with high-tack labels, which maintain adherence in our –112°F freezers. The barcoding software is written in Perl, uses the ZPL printer language to format the labels and distributes printing using lpr.

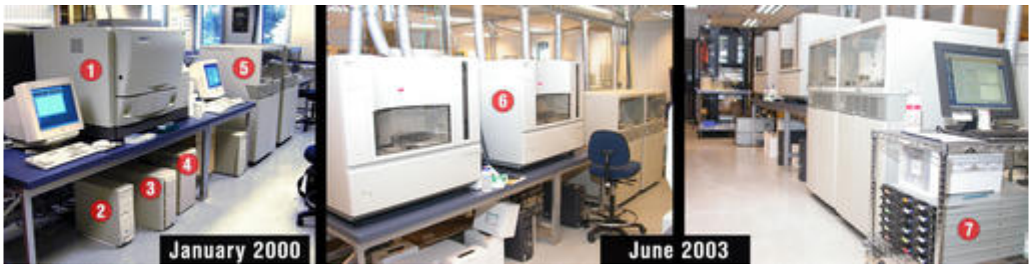


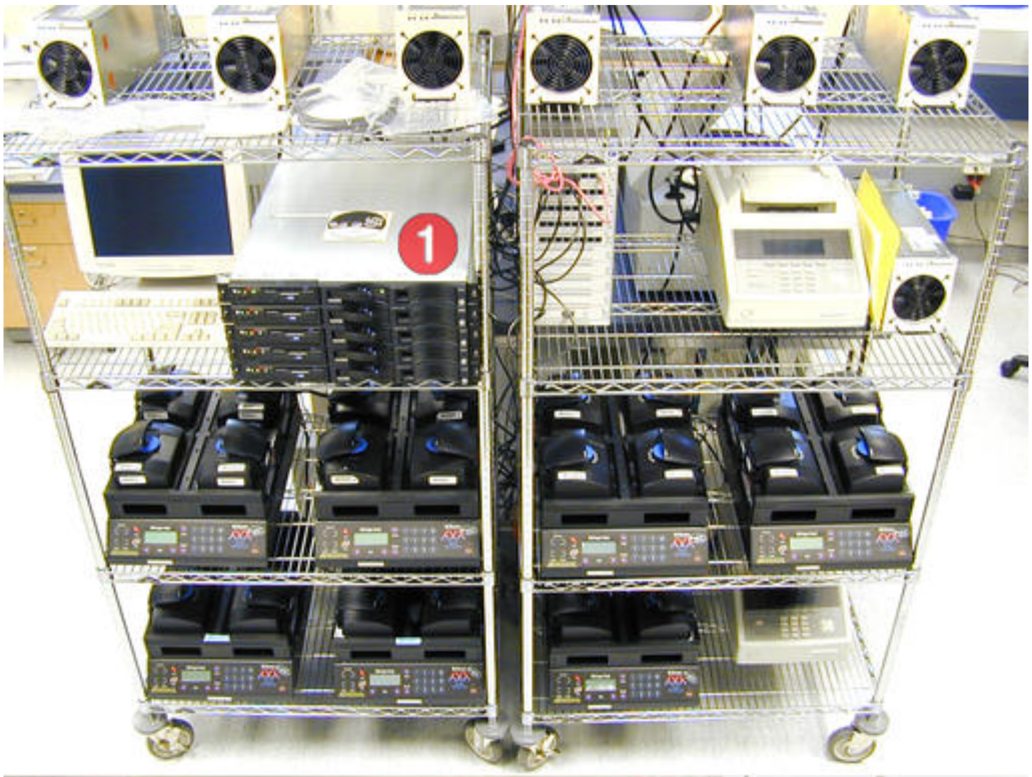
Figure 6. Sequencers: 1) MegaBACE 1000, 2) sequencer PC, 3) UPS, 4) sequencer power supply, 5) ABI 3700s, 6) ABI 3730XL and 7) x330 sequencer cluster.

Three generations of sequencers have passed through our lab since the MegaBACE 1000, and we currently operate six ABI 3700s and three ABI 3730XLs (Figure 6). The latest, the ABI 3730XL, is capable of accepting multiple 384-well plates and sequencing 1,152 DNA samples in 24 hours. With each sample yielding up to 700–800 high-quality bases, a single 3730XL produces about 800,000 bases per day.

The Tor2/SARS genome was sequenced using a whole-genome shotgun (WGS) method. In this approach, random sections of the genome are sequenced in a redundant fashion and then assembled together to recover the entire genomic sequence. Given that the size of the pathogen was anticipated to be approximately 30,000 bases, it would take a minimum of 40 reads to span the genome. However, because the reads originate from random regions, more than the minimum number of reads required in order to have enough overlap for a complete assembly. Redundancy also allows for more confidence in determination of the base at each given position in the genome.

Beige Turns Black

By the time we bought our first set of IBM x330 servers, now part of a 168-CPU cluster (Figure 7), the 1U platform was on the verge of entering the commercial off-the-shelf (COTS) category and starting to enjoy COTS prices. Beige boxes are no longer used for distributed computing. Heavily loaded production subsystems, like Apache and MySQL, are housed on IBM's 4U x440s, which are eight-way hyperthreading Xeon nodes with 8GB of RAM. These boxes are running SuSE 8.1—one of the few distributions that supports IBM's Summit chipset. The x440 is a NUMA machine with 32MB of L4 cache per four-CPU module, and without IBM's Summit patches only presents two CPUs to the kernel. SuSE's 2.4.19 derived kernel with bigmem+Summit support makes it possible to use all eight CPUs and 8GB of memory. Even without the advanced NUMA scheduler code now in the 2.5 series kernels, these x440s have been real workhorses allowing us to run eight BLAST processes concurrently with enough RAM to cache the entire human genome in shared memory. Anyone who claims Linux isn't ready for Big Iron is in for a surprise.


1

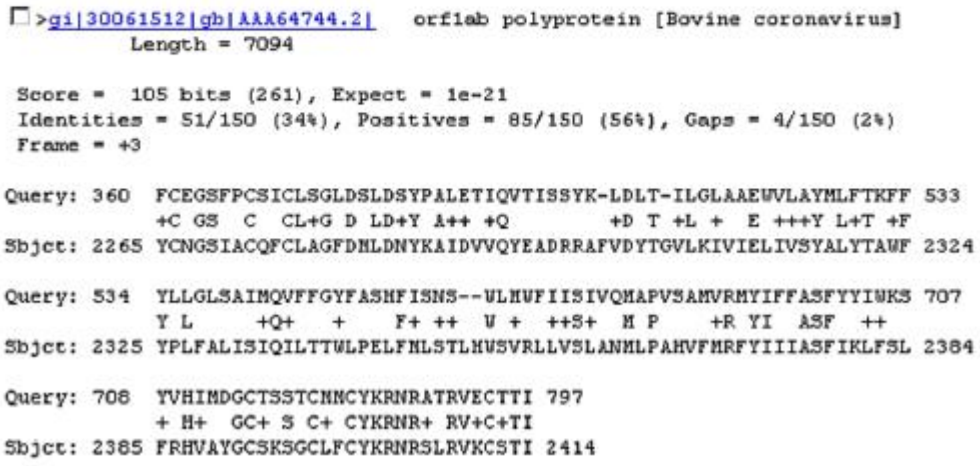
2

3
4

Figure 7. Computing and storage infrastructure: 1) initial deployment of x330 in January 2001, 2) 84 x330 nodes and 3) NetApp FAS960 filer and two IBM 3583 LTO libraries with two drives and 72 slots each controlled by Veritas running on an x342.

As we rapidly grew, the NFS subsystem was beginning to show problems. In particular, machines were crashing with some combinations of NFS server/ client versions. Although in our experience NFS clients are robust, room for improvement exists with current Linux NFS services. Our fastest NFS server, an IBM x342 (2xP3-1.24, 2GB RAM) cannot handle more than 4,000–6,000 NFS ops/ second, especially during a large number of parallel read/writes from our cluster. To address the performance limits, we acquired a NetApp FAS960 Filer (Figure 7). With 10TB of raw storage (5x14x144GB), the filer has reached 30,000 NFS ops/second. Despite NFS issues, our original VAR900 production file server (Figure 2) was the poster child of stability and reached an uptime of 394 days in February 2002 before it had to be rebooted for upgrades.

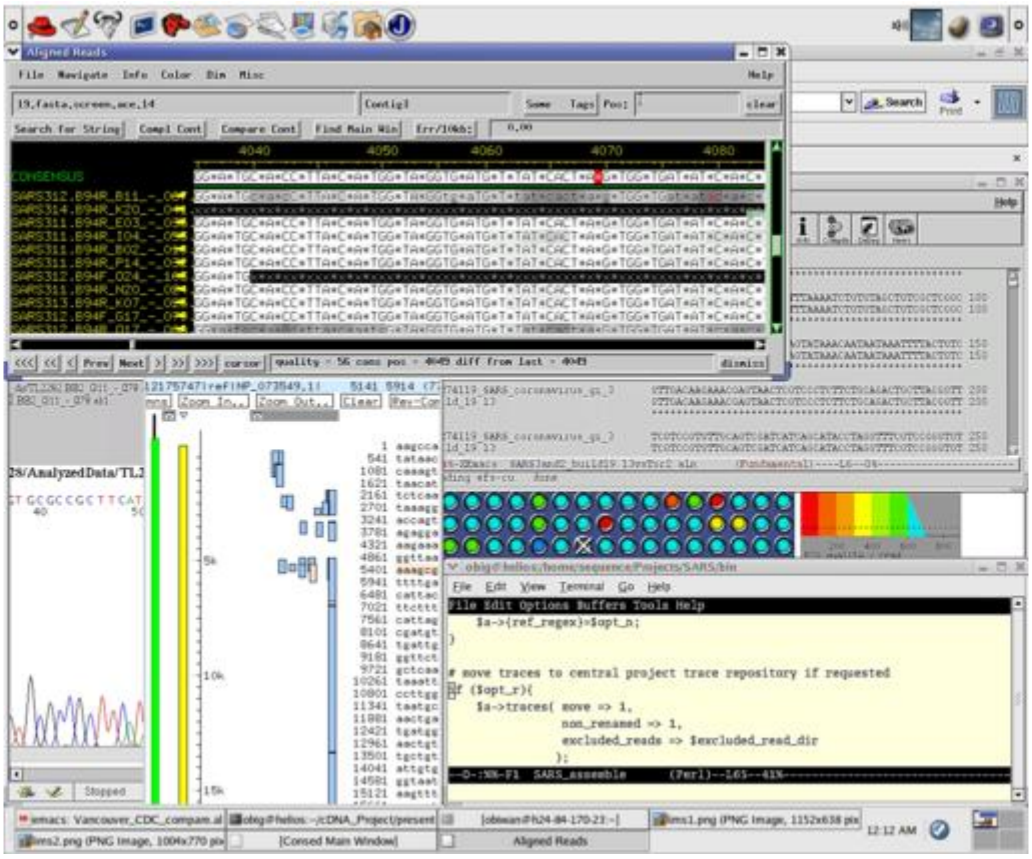
The first set of Tor2/SARS sequence data was available for our informatics group to analyze on Friday evening, April 11, 2003. To verify our sequence reactions, we checked it for contamination. A BLAST search allowed us to determine the closest match in the public proteomic and genomic databases. To our relief, the best match was to bovine coronavirus (Figure 8), indicating that we were sequencing something related to coronaviruses. The sequences of these viruses end in a string of As, and when we saw sequence reads ending in a poly-A tail we were confident that this was one end of the genome.



```
>gi|30061512|gb|AAA64744.2|    orf1ab polyprotein [Bovine coronavirus]
          Length = 7094

 Score =  105 bits (261), Expect = 1e-21
 Identities = 51/150 (34%), Positives = 85/150 (56%), Gaps = 4/150 (2%)
 Frame = +3

Query: 360  FCEGSFPCSICLSGLDSLDSYPALETIQVTISSYK-LDLT-ILGLAAEWVLAYMLFTKFF 533
            +C GS  C  CL+G D LD+Y A++ +Q        +D T +L +  E +++Y L+T +F
Sbjct: 2265 YCNGSIACQFCLAGFDMLDNYKAIDVVQYEADRRAFVDYTGVLKIVIELIVSYALYTAWF 2324

Query: 534  YLLGLSAIMQVFFGYFASHFISNS--WLHWFIISIVQMAPVSAMVRMYIFFASFYYIWKS 707
            Y L     +Q+   +    F+ ++  W +  ++S+  M P    +R YI  ASF  ++
Sbjct: 2325 YPLFALISIQILTTWLPELFNLSTLHWSVRLLVSLANMLPAHVFMRFYIIIASFIKLFSL 2384

Query: 708  YVHIMDGCTSSTCNNCYKRNRATRVECTTI 797
            + H+  GC+ S C+ CYKRNR+ RV+C+TI
Sbjct: 2385 FRHVAYGCSKSGCLFCYKRNRSLRVKCSTI 2414
```

Figure 8. Output of the Top Hit from a BLAST Query

The x330s and an x440 were used to analyze and assemble the SARS data. The genome is not very large, and the assembly took less than 15 minutes on a single CPU. In comparison, the first public assembly of the human genome, 300,000 times the size of Tor2/SARS, was done at UCSC and took four days on a 100-CPU Linux cluster.



```
File Edit Options Buffers Tools Help
    $a->{ref_regex}=$opt_n;
}

# move traces to central project trace repository if requested
if ($opt_r){
    $a->traces( move => 1,
                non_renamed => 1,
                excluded_reads => $excluded_read_dir
              );
--D-:%%-F1  SARS_assemble      (Perl)--L65--41%-----------
```

Figure 9. Sequence Analysis on the Linux Desktop

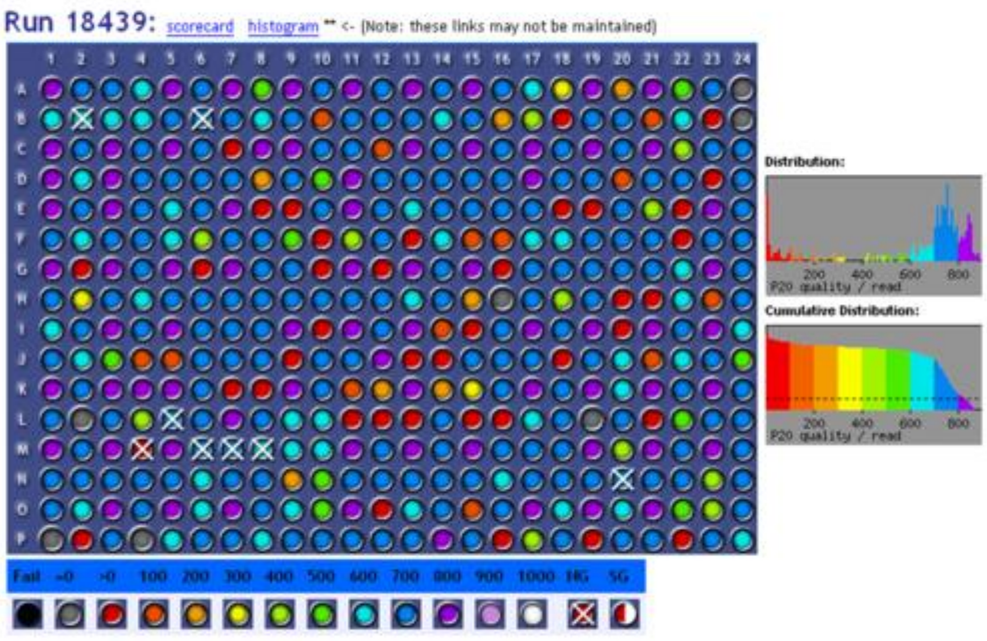


Figure 10. Sequence Read Quality for One of the SARS Plates

By Saturday, April 12, 2003, at 2:25AM, we completed our seventh build of Tor2/ SARS, and this assembly was frozen as the first draft. It was imported into AceDB to visualize alignments to other known protein sets for validation (Figure 9). We spent Saturday validating the assembly, which was posted later that day to our x440 public Web server using a custom CMS system running under Zope/Plone.

Conclusion

The sequence of Tor2/SARS has identified a fourth novel group of coronaviruses and provides the necessary information to develop diagnostic tests and, possibly, therapies including a vaccine. Linux has made it possible to get our work done without spending a fortune on hardware or software. Using commodity hardware has minimized depreciation loss due to long implementation times. We'll be watching for new bugs to catch, and in the meanwhile, our MySQL database is open for sequencing.

Acknowledgements

The authors would like to thank Marco Marra, Steven Jones, Caroline Astell, Rob Holt, Angela Brooks-Wilson, Jas Khattra, Jennifer Asano, Sarah Barber, Susanna Chan, Allison Cloutier, Sean Coughlin, Doug Freeman, Noreen Girn, Obi Griffith, Steve Leach, Mike Mayo, Helen McDonald, Steven Montgomery, Pawan Pandoh, Anca Petrescu, Gord Robertson, Jacquie Schein, Asim Siddiqui, Duane Smailus, Jeff Stott and George Yang for scientific expertise, lab and bioinformatics efforts. We also would like to thank Kirk Schoeffel, Mark Mayo and Bernard Li for their system administration advice.

Command-Line Bioinformatics

Let's do some bioinformatics using bash and a few binaries out of /bin and /usr/bin. We will calculate the GC ratio of the Tor2/SARS genome—the fraction of base pairs that are either a G or a C. Let's avoid using awk to make things interesting. First, download the sequence with wget, using -q to silence its verbose output:

```
> wget -q http://mkweb.bcgsc.ca/sars/AY274119.fa
> head AY274119.fa
gi|30248028|gb|AY274119.3| SARS coronavirus TOR2
ATATTAGGTTTTTACCTACCCAGGA...
```

The sequence file is in FASTA format consisting of a header line and the sequence, split into fixed-width lines. The following counts the number of Gs and Cs in the sequence and presents the total as a fraction of the total number of bases:

```
> grep -v "^>" AY274119.fa | fold -w 1 |
tr "ATGC" "..xx" | sort | uniq -c |
sed 's/[^0-9]//g' | t -s "\012" " " |
sed 's/\([0-9]*\) \([0-9]*\)/scale = 3;
↪\2 \/ (\1+\2)/' |
bc -i
scale = 3; 12127 / (17624+12127)
.407
```

Out of the 29,751 bases in our sequence, 12,127 are either G or C, giving a GC content of 41%.

GSC MySQL LIMS

We collected 3,250 sequencing reads containing 2.1 million quality base pairs contributing toward the initial draft assembly. This represented roughly 70X redundant coverage of the genome. WGS is usually done to no more than 10X, but for us, time was of the essence, and we wanted to avoid delays associated with finishing regions that were not fully covered by the first round of sequencing.

```
SELECT
SUM(Sequence_Length) AS bp_tot,
AVG(Quality_Length) AS bpq_avg,
SUM(Quality_Length) AS bp_qual_tot,
COUNT(Well) AS reads,
Sequence_DateTime AS date,
Equipment_Name AS equip
FROM
Equipment, Clone_Sequence, Sequence_Batch, Sequence,
Plate, Library, Project
WHERE FK_Sequence_Batch__ID=Sequence_Batch_ID AND
FK_Plate__ID=Plate_ID AND
FK_Library__Name=Library_Name AND
FK_Equipment__ID=Equipment_ID AND
FK_Project__ID=Project_ID AND
FK_Sequence__ID=Sequence_ID AND
Sequence_Subdirectory like "SARS2%" AND
Quality_Length > 100 AND
Sequence_DateTime < "20030413"
GROUP BY Sequence_ID ORDER BY Sequence_DateTime;

bp_tot bpq_avg  bp_tot reads  date              equip
437256 612.6399 205847 336 2003-04-11 21:07:06 SARS212.B21 D3730-3
412366 752.1074 245187 326 2003-04-11 22:15:34 SARS213.B21 D3730-1
269456 639.1926 225635 353 2003-04-11 22:22:34 SARS215.B21 D3700-6
130525 715.5060 118774 166 2003-04-11 22:25:44 SARS216.B21 D3700-5
282490 682.6311 249843 366 2003-04-11 22:27:14 SARS215.BR D3700-4
310119 612.7601 212015 346 2003-04-11 22:31:56 SARS213.BR D3700-1
182573 681.4975 136981 201 2003-04-11 22:36:40 SARS216.BR D3700-3
301471 642.2273 226064 352 2003-04-12 01:58:16 SARS212.BR D3700-2
401595 690.5204 220276 319 2003-04-12 05:13:26 SARS211.BR D3730-3
460100 642.0468 219580 342 2003-04-12 06:20:52 SARS214.BR D3730-2
182360 471.7832  67465 143 2003-04-12 07:14:44 SARS214.B21 D3730-1
```

Resources

Growth of Genbank: www.ncbi.nlm.nih.gov/Genbank/genbankstats.html

How Perl Saved the Human Genome Project: bioperl.org/GetStarted/tpj_ls_bio.html

Image of a Coronavirus: www3.btwebworld.com/vdg/gallery/Coronavirus.jpg

SARS Issue of *Science*: www.sciencemag.org/feature/data/sars

SARS Statistics and Information: www.cdc.gov/ncidod/sars and lassesen.com/sars

Timeline of SARS History: www.worldhistory.com/sars.htm

UCSC Assembly of Human Genome: www.cse.ucsc.edu/~learithe/browser/goldenPath/algo.html

Martin Krzywinski (martink@bcgsc.ca) is a bioinformatics research scientist at Canada's Michael Smith Genome Sciences Centre. He spends his time applying Perl to problems in physical mapping and data-processing automation. In his spare time he can be found encouraging his cat to stick to her diet.

Yaron Butterfield (ybutterf@bcgsc.ca) leads the sequencing bioinformatics team at Canada's Michael Smith Genome Sciences Centre. He and his group develop DNA sequence analysis and visualization software and pipelines for various genome and cancer-based research projects.

Archive Index Issue Table of Contents

Advanced search

TALOSS: Three-Dimensional Advanced Localization Observation Submarine Software

Douglas B. Maxwell

Richard Shell

Issue #115, November 2003

This new project from the Naval Undersea Warfare Center will test whether an integrated 3-D display can give the commanding officers of US submarines the ability to make better decisions faster.

The US military has adopted the concept of network-centric (net-centric) operations as a means to develop speed of command. Under this methodology, speed of command has three parts: 1) the force achieves information superiority, having a dramatically better awareness or understanding of the battlespace rather than simply more raw data; 2) the forces acting with speed, precision and reach achieve the massing of effects, not solely the massing of forces; and 3) the results that follow are the rapid foreclosure of enemy courses of action and the shock of closely coupled events.

A significant issue in this concept pertains to the effectiveness with which distributed military combat systems can be integrated operationally to yield maximum battlespace awareness for the battle force. Net-centric warfare requires a common operational/tactical picture; that is, there must be a reliable and consistent common tactical picture (CTP) across platforms to provide dominant battlespace awareness in the multiwarfare environment.

The challenge in achieving speed of command is developing rapid, accurate awareness and understanding of the entire battlespace. Traditionally and currently, decision-makers develop a mental model of the battlespace by assimilating data from multiple 2-D displays and paper plots. A net-centric, or distributed, environment demands a new approach for presentation of the battlespace—both in the level of detail and in the manner in which it is presented.

The TALOSS system is designed to enable the rapid and/or accurate assimilation of complex information in the undersea battlespace. TALOSS is capable of generating a common undersea tactical picture, which includes threat danger zone estimation, sensor contact tracking and one's own ship's position and orientation presented in combination with navigational/ topographic/bathymetric information. It is hypothesized that this integrated undersea picture can enable faster and more accurate decision-making, as well as provide improved planning and decision aids.

General Architecture

TALOSS is compatible with and has been tested under Red Hat 7.0–9.0 as well as Slackware 9. Linux was chosen as the operating system for several reasons: 1) it is compatible with current and future submarine combat systems; 2) it is a generic UNIX operating system, which means software and script files developed under Linux are transferred readily to other UNIX operation systems, such as HP-UX and IRIX; and 3) it is an open-source operating system with a large user community that can be tapped easily for system optimization and maintenance. TALOSS is comprised of three main modules: the Feeder, the Bezel and the 3-D Display. Figure 1 is an architecture diagram of the baseline software.

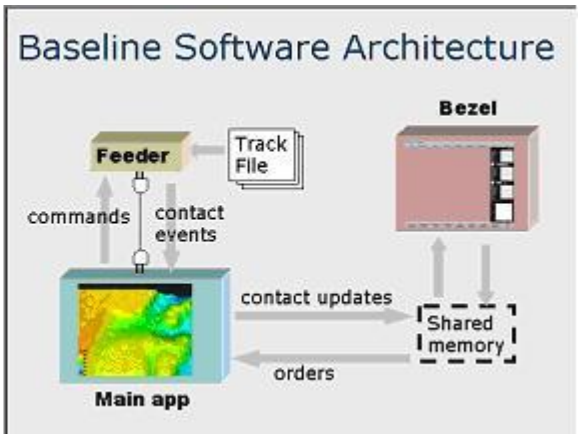


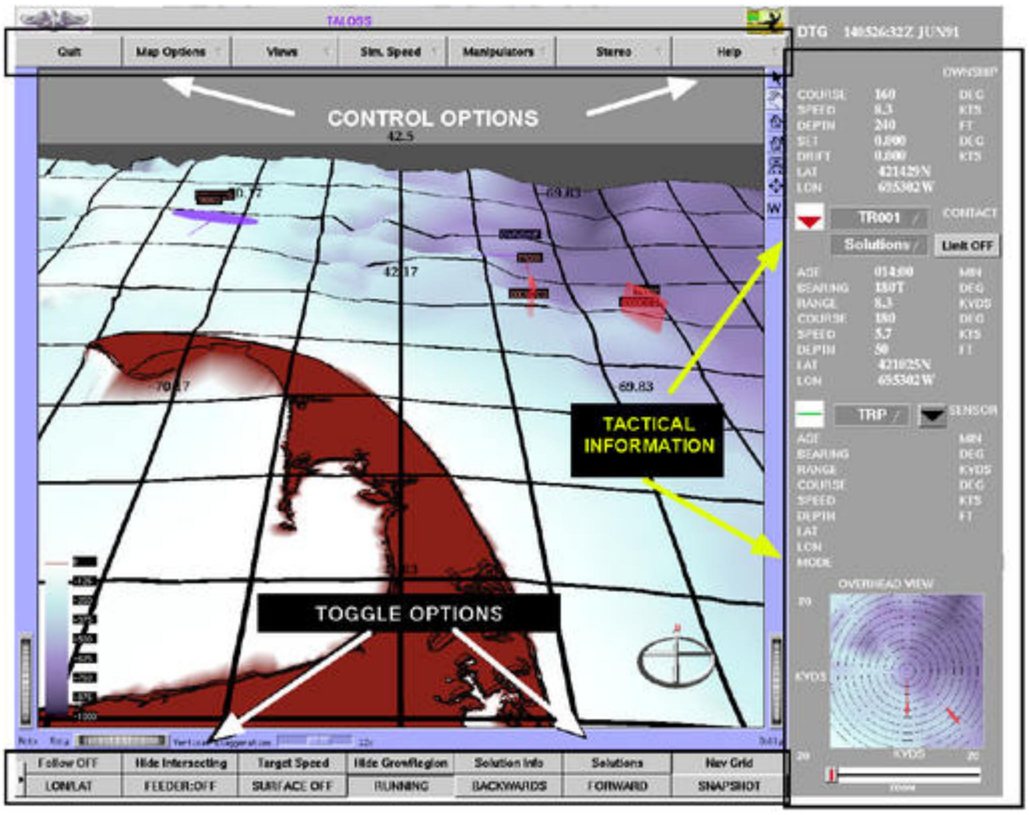
Figure 1. Baseline TALOSS Architecture

Tactical Data Input

Feeder is a program that reads in and sends submarine Combat Control System (CCS) data to the main display through TCP/IP sockets. The program can be linked directly to the combat database, or it can run reconstruction and demonstration data through ASCII input files. Feeder is a flexible module that can be reconfigured easily to import multiple databases. Because of this flexibility, the TALOSS system is compatible with non-submarine-related applications, such as oceanographic/topographic 3-D maps, 3-D velocity fields and 3-D radar/sonar maps, as well other applications involving objects moving in a 3-D environment.

Simulation Control

The Bezel is an information/control GUI written using the Fast Light Tool Kit (FLTK). It controls the 3-D main application (Main App) and, to a limited extent, the Feeder program. In addition to its role as a system controller, the Bezel also serves as a system status indicator. One way it displays system status is through an Open Inventor 3-D window showing the top-down view of the battlespace centered about one's own ship, labeled ownship. It is essentially a 2-D view of the 3-D scene giving the user an orientation point into the 3-D scene from which to locate ownship's position and orientation rapidly. The Bezel and the Main App communicate to each other through Shared Memory. Figure 2 shows a complete TALOSS display featuring the FLTK control Bezel linked with the Open Inventor 3-D display.



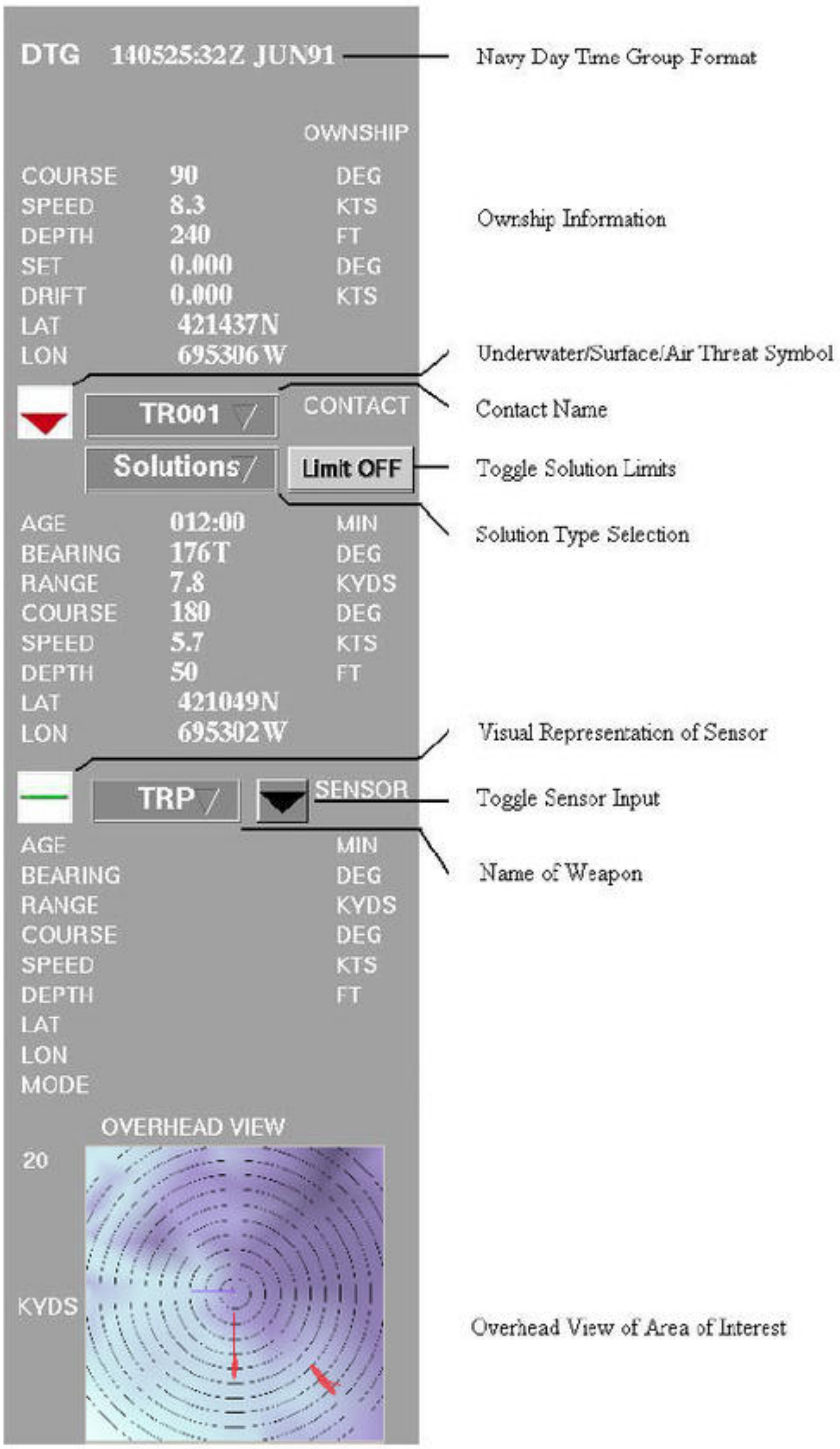
DTG 140526:32Z JUN91

COURSE	160	DEG
SPEED	8.3	KTS
DEPTH	240	FT
SET	0.000	DEG
DRIFT	0.000	KTS
LAT	421429N	
LON	695302W	
AGE	014:00	MIN
BEARING	180T	DEG
RANGE	8.3	KYDS
COURSE	180	DEG
SPEED	5.7	KTS
DEPTH	50	FT
LAT	421025N	
LON	695302W	

Figure 2. Complete TALOSS Display

In Figure 2, observe the pull-down menus at the top of the Bezel. They allow the user access to basic TALOSS functions, such as exiting the program, changing the view, changing the map color, changing the depth regime limits and so forth. Similarly, located on the bottom of the Bezel are two rows of toggle buttons that allow for the control of the tactical information presented on the 3-D display of the undersea battlespace.

The right side of the Bezel is distinct from the top and bottom in that its purpose is to provide pertinent tactical information, such as: 1) ownship, target and weapon information; 2) target and sensor selection status; and 3) numeric target containment region information. In addition, because the entire system is synchronized to a common operational clock, whose time is given in the Navy daytime group (DTG) format, that too is displayed on the top-right side of the Bezel. Figure 3 shows a representative illustration of the right side of the Bezel with corresponding feature description.



COURSE	90	DEG
SPEED	8.3	KTS
DEPTH	240	FT
SET	0.000	DEG
DRIFT	0.000	KTS
LAT	421437N	
LON	695306W	

AGE	012:00	MIN
BEARING	176T	DEG
RANGE	7.8	KYDS
COURSE	180	DEG
SPEED	5.7	KTS
DEPTH	50	FT
LAT	421049N	
LON	695302W	

AGE		MIN
BEARING		DEG
RANGE		KYDS
COURSE		DEG
SPEED		KTS
DEPTH		FT
LAT		
LON		
MODE		

Figure 3. Right-Side Bezel Features

3-D Scene Manipulation

The 3-D undersea battlespace display receives its tactical information from the Feeder program. It receives mapping and navigational information from the

Digital Nautical Chart database that is loaded at startup and updated based on the evolution of the tactical situation. All navigational information is pre-rendered as Open Inventor binary files, called Navigation Tiles. The Main App combines all tactical and navigation information into a comprehensive 3-D picture, rendering it using the computer platform-independent scenegraph, Open Inventor.

The lower-left corner of the 3-D display contains three scene interaction controls. The Rotx dolly rotates the scene about an imaginary x-axis running horizontally across the screen. The Roty dolly rotates the scene about an imaginary y-axis running vertically across the screen. The vertical exaggeration slider bar changes the depth ratio of the scene. This serves to exaggerate 3-D depth within the scene. To use any of these dolly wheels, simply place the mouse cursor over the dolly, depress and hold the left button while dragging the cursor in the desired direction. The lower-right corner of the 3-D display contains a zoom dolly. Zoom-in is limited to collision with the bottom, at which point zoom ceases.

In addition to scene manipulation devices, the 3-D display contains seven interaction buttons. The buttons perform functions such as selecting a contact, scene manipulation, resetting a home view and toggling a wire-frame view.

Some other 3-D display features of note are a free-floating 3-D compass that simultaneously indicates both direction and 3-D scene orientation and a colorbar that indicates depth shading. Depth shading is a combination of color map and depth regime selected. Figure 4 illustrates these features.

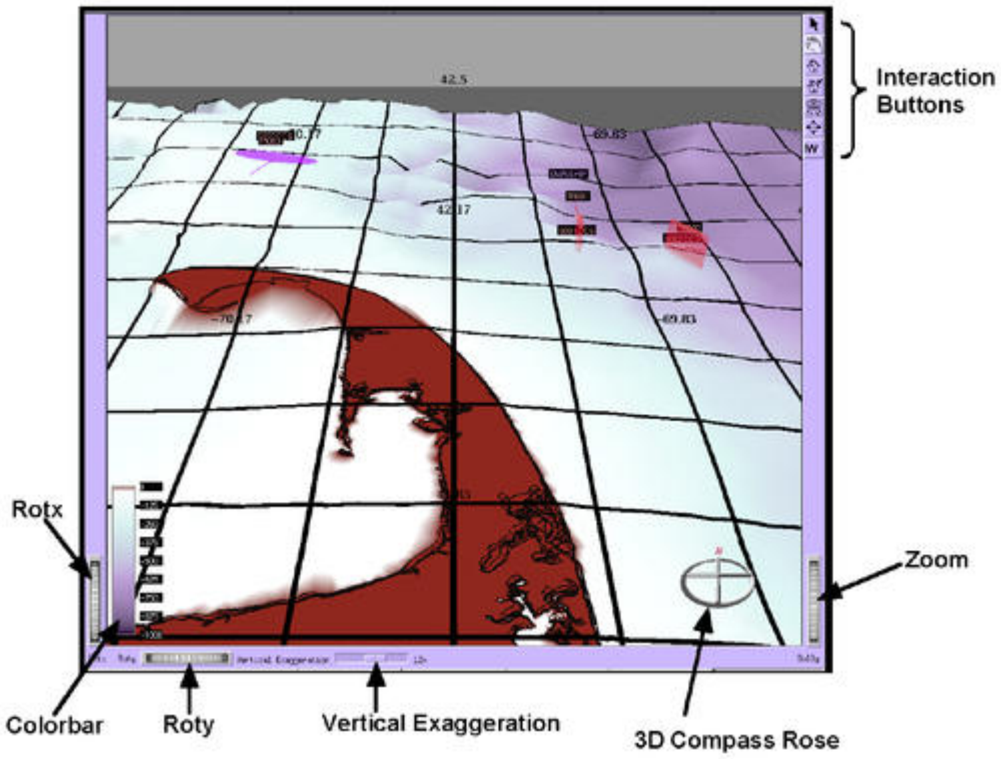


Figure 4. 3-D Display Features

Operation

The purpose of a tactical display is to give an operator a sense of the location of everything relative to the operator's own position. This is accomplished in TALOSS by a series of concentric range rings centered about ownship. The range rings represent set intervals that allow an operator to determine instantly, visually, how far any particular object is from ownship. This information is particularly important in collision avoidance and threat assessment. These rings are shown both in the 3-D display and in the top-down view shown in the Bezel.

An essential component of any tactical or situational awareness display is precise knowledge of location on the earth. TALOSS supports navigational information using two methods. The first is a system of navigational grid markers that are ten degrees of latitude and longitude. Because distance is equal for all latitudes, this amounts to ten nautical miles for latitude. Because the distance between longitude lines varies with latitude, the number of nautical miles between longitude lines is dependent on latitude. For temperate latitudes, it is approximately ten nautical miles. Second, in addition to the lines of constant latitude/longitude, an alphanumeric value appears on the 3-D map indicating latitude and longitude. These values and the navigational grid lines are extracted from the Digital Nautical Chart database. The navigation grid toggles with the range ring display; however, the alphanumeric markers are always displayed. Figure 5 shows a view of the 3-D display with all navigational information enabled.

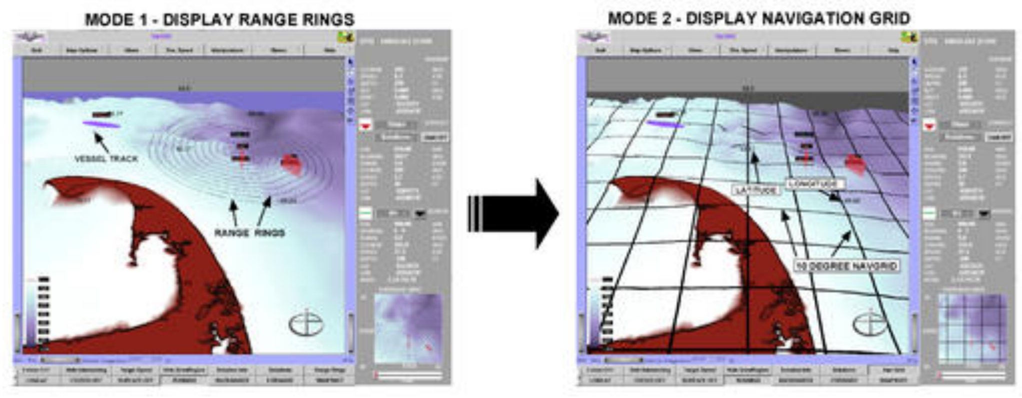

MODE 1 - DISPLAY RANGE RINGS
VESSEL TRACK
RANGE RINGS
MODE 2 - DISPLAY NAVIGATION GRID
LATITUDE
LONGITUDE
10 DEGREE NAVGRID

Figure 5. Navigation and Tracking Information

The most important component of a situational awareness display is the ability to track and to integrate all moving objects visually. In TALOSS, solid lines indicate the estimated tracks of all known objects. Associated with the tracks is a color scheme indicating intent, which can be hostile, friendly or neutral. The respective colors for these classifications are red, blue and yellow. Because the current version of TALOSS is designed for submarine applications, it also includes tracks for weapons, both ownship's and hostile. The coloring scheme is green for ownship's weapons and orange for hostile weapons. This color scheme, however, is easily modified for other applications. Figure 5 illustrates tracks on several contacts as well as ownship. All contacts have their tracks labeled as defined in the contact window of the Bezel.

In addition to tracking information, TALOSS does bookkeeping for potential danger zones as 3-D containment regions. The same color schemes used for tracks apply to these containment regions: red, blue and yellow. Figure 6 indicates a typical 3-D containment region. Containment regions are computed and displayed as complex volumetric shapes. Containment region selection is controlled by the Bezel. The Bezel allows for both containment region highlighting and selection of the contacts contributing to a particular containment region.

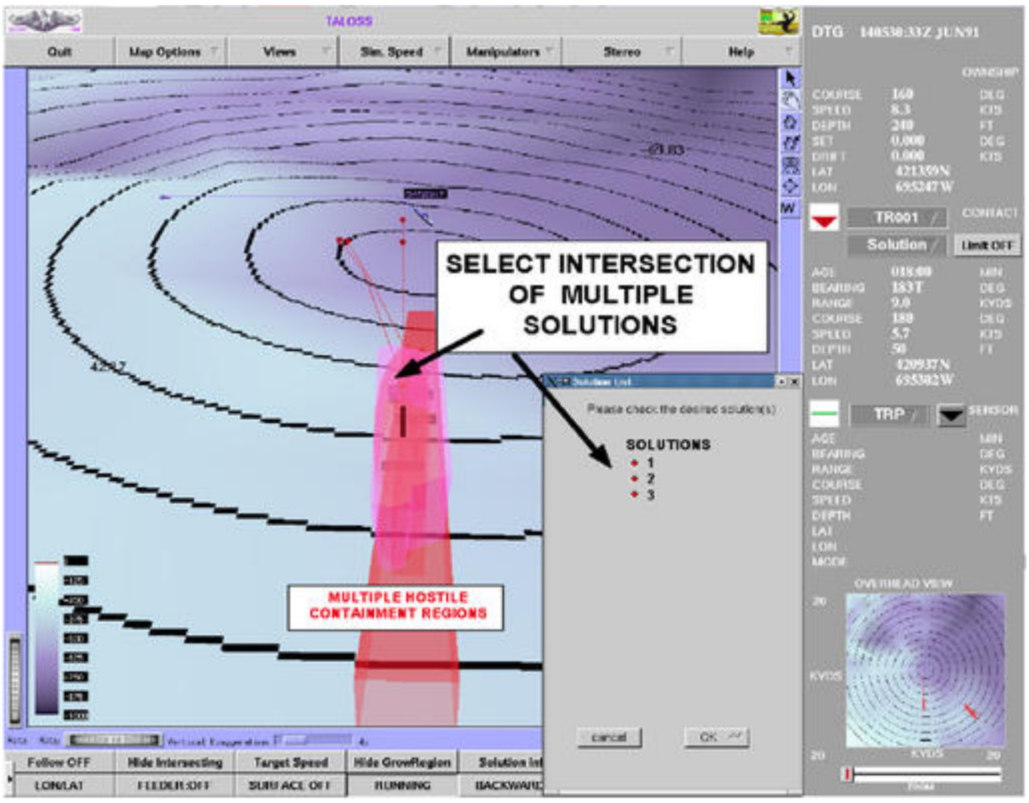


Figure 6. 3-D Containment Region

Containment regions can be grown and intersected to mimic a vessel's movement over time. Growth of the regions represents all possible locations the vessel could occupy within the containment volume when precise sensor contact on that vessel is lost. When a sensor update on the vessel is obtained, that updated information, in the form of new 3-D containment volume, can be intersected with the previously grown 3-D volume to extract a much-reduced common volume (see Figure 7). This common volume has the highest probability of containing the vessel of interest. The goal of a military combat system is to “localize” the threat containment volume rapidly, so an appropriate tactical response can be precipitated expeditiously. In other words, fight or flight.

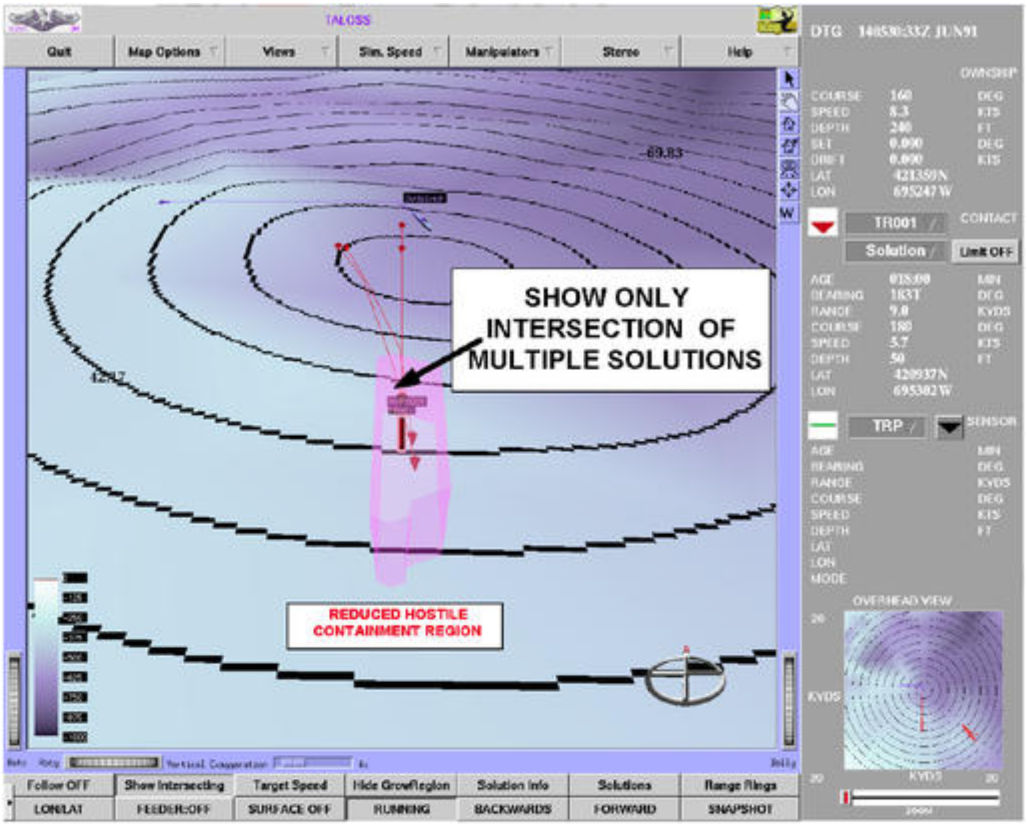


Figure 7. 3-D Intersection Region

Pioneering software in the intersection of two or more noncontiguous volumes has been developed for TALOSS by Arizona State University. One of the main advantages of using the Linux operating system for software development is that research into specific areas of interest can be done inexpensively at universities or other open-source sites. Using Linux means that the participating research partners do not need to buy expensive development platforms, such as Hewlett-Packard TAC or Silicon Graphics workstations. Instead, they can develop code easily integrated into a combat system on inexpensive Linux-based PC systems. The US government is encouraging the use of commercial-off-the-shelf (COTS) products for development and deployment in the US military, both as a cost-saving measure and as a means for ensuring widespread long-term support of the product. Using Linux as both a development and operational environment for military systems is a perfect example of COTS in action.

Conclusions

A flexible, modularized, 3-D data fusion visualization system has widespread applicability for both military and civilian applications. Under the sponsorship of the Office of Naval Research (ONR), the Naval Undersea Warfare Center, located in Newport, Rhode Island, has created such a system for the visualization and integration of the submarine undersea battlespace. This system, TALOSS, has been designed to integrate a wide variety of the databases, both civilian and military. Because the software is modularized and written to run under Linux, it has the potential for transition as an open-source rendering and data fusion engine.

The ultimate aim of the project, funding permitting, is to develop a completely modularized TALOSS toolkit, unclassified portions of which will be made available for civilian use as open-source software. There are two primary motivations for this release: 1) the software was developed with public funds and should be made available to the general public if national security is not compromised, and 2) by making the unclassified portion of the software available, it is hoped that improvements in its operation by the Open Source community can be incorporated into the classified portion of the software enhancing its operation.

Resources

"Naval Transformation Roadmap, Sea Strike, Sea Shield, Sea Basing", Department of the Navy, Washington, DC, 2002 (UNCLASSIFIED). Available on-line at spica.gl.nps.navy.mil/ORarchives/SEA-TRIAL/NavalTransformRdMap.pdf.

K. Lima, "Visualization for Multiwarfare Planning and Execution", ONR Command and Control & Combat Systems Gathering 2002, Arlington, Virginia, April 23–25, 2002 (UNCLASSIFIED).

G. M. Nielson and G. L. Graf, "Application of Volume Modeling Techniques to Dynamic Containment Regions for Naval Applications", Interim Progress, ONR Grant N0014-02-1-0287, Arizona State University, Phoenix, Arizona, May 15, 2002 (UNCLASSIFIED).

R. Shell, L. Mathews, K. Lima, R. King and F. Das Neves, "Undersea Command and Control Visualization", July 22–26, 2001, Proceedings of the Seventh Annual Joint Aerospace Weapon Systems Support, Sensors, and Simulation Symposium & Exhibition (JAWS S3), San Diego, California, 2001, p. 8.

Douglas B. Maxwell is currently a member of the Weapons and Countermeasures branch (Code 2213) of the Naval Undersea Warfare Center. He received his MS in Mechanical Engineering (2001) from Louisiana Tech University.

Richard Shell is an electrical/computer engineer at the Naval Undersea Warfare Center Division, Newport, Rhode Island. Richard Shell's current work in 3-D visualization has resulted in an invited paper and presentation at the Joint Aerospace Weapons System Support, Sensors and Symposium (JAWS S3) in 2002 as well as a paper/poster presentation at the European Undersea Defense Technology Conference (UDT), La Spezia, Italy, in 2002.

My Other Computer Is a Supercomputer

Steve Jones

Issue #115, November 2003

Steve Jones started taking notes on PBS, MPI and MOSIX in November 2002, and by June 2003 he was manager of a cluster on the TOP500 list. Here's how the Rocks distribution can help cluster managers get systems up and running.

In November 2002, I was called by Mitch Davis (Executive Director of Academic Technology, ITSS, Stanford University) and Carnet Williams (Director of Academic Technology, ITSS, Stanford University) regarding an aggressive, high-profile project. While I was Director of Network Operations at the Stanford Law School, I had the pleasure of working with both Mitch and Carnet during their respective terms as Associate Dean/CIO of Stanford Law School. They told me Dr Vijay Pande, the principal investigator behind the Folding@home Project, wanted to purchase a large commodity cluster, and they sent my name to Vijay as someone who could manage projects effectively to completion. Instinctively, I agreed. We discussed more details of the project, and right before I hung up, I asked, “How big will it be?” They responded, “300 dual-processor nodes.” I thought, “600 CPUs...that should do some damage.”

While Mitch, Carnet and Vijay worked with Dell and Intel to negotiate the purchase of the cluster, I sent an e-mail to Vijay Pande, stating that I could assist him with the network and hardware side of the project and that I hoped to learn more about the software side during the process. The last line in my message said, “I want to be part of something great.” Vijay responded promptly and welcomed my assistance. We set up our first meeting to discuss the scope of the project.

At that initial meeting, it seemed most things were up in the air. Everyone knew equipment was coming, but no real plans were in place. Vijay said he knew that authentication and filesystem choices had to be made, and of course the opportunity to use existing Stanford services was considered.

Vijay also mentioned running PBS, MPI and MOSIX. I knew very little about any of these, but took notes and, back at my desk, did a Google search for those names along with the words “beowulf” and “cluster”. I came across a presentation about building a cluster using an open-source distribution named Rocks from an organization called NPACI (www.rocksclusters.org). The presentation was excellent. It answered so many of my questions, such as, how would we put together such a cluster, how would we manage software on nodes, how would we configure the master node and how would we monitor nodes. Basically, the presentation was a framework for how we would build our cluster. I printed copies of the presentation and brought them to our next meeting. The idea of using a packaged solution was well received.

Bringing Up Iceberg

During the time these two meetings were taking place, the cluster was being racked and stacked by Dell in Stanford's Forsythe Data Center, which took seven days. I was able to download a copy of Rocks version 2.3 and run through the installation process on what is defined as the front-end node in Rocks nomenclature. This task was simple, and I was quite impressed at this point. At our third meeting, my role in the project had expanded from being involved only with hardware and the network, to handling software also, as I already had brought up the front end with Rocks successfully. I felt confident that I could handle the rest of it as well, but at this point I didn't realize the true scope of the project. I was embarking on building the largest-known Rocks cluster.

The first issue I ran into was trying to install compute nodes. A Rocks utility called insert-ethers is used to discover compute nodes' Ethernet MAC addresses, assign them an IP address and hostname and then insert this information into a database, during a negotiated process using PXE and DHCP. Following the node insertion, the node is built and configured as defined in a Red Hat Kickstart file, completing the PXE boot process. Unfortunately, I had problems with the network interface cards in the Dell PowerEdge 2650, as the Broadcom Ethernet controllers did not appear to be supported in Rocks. I sent my issue to the Rocks discussion list, and I also called Dell for support and opened a ticket for service under our Gold support contract. The Rocks developers quickly provided an experimental version of their cluster distribution that contained updated drivers, which solved the problem, and soon I saw my suggestions and observations incorporated into the maintenance release of Rocks version 2.3.1.

Figure 1. Iceberg in the Forsythe Data Center

The final issue, which was discovered at scale, was the inability to have more than 511 active jobs. My users were screaming about the 100 idle processors, because many of the jobs run on Iceberg are short-lived, one- to two-processor jobs. While working with the Rocks Development team, we looked for a defined constant in the Maui scheduler code. I eventually found it, and under the guidance of the Rocks team, recompiled and restarted Maui. The front end now can schedule as many active jobs as there are processors.

TOP500 Run

By late December 2002, with the last of the hardware and software issues resolved, we turned our sites toward putting Iceberg on the TOP500 Supercomputer list (www.top500.org). The TOP500 is a biyearly competition that ranks 500 entries by sustained performance for a linear equation solver, Linpack. On the November 2002 list, there were 97 commodity clusters, so we felt confident we could put Iceberg on the list.

Taking a run at the TOP500 list proved to be more work than we anticipated. Rocks comes with a prebuilt Linpack executable that can attain good performance on a Pentium 4 cluster, but I wanted more. My account representative put me in contact with the Scalable Systems Group at Dell. We collaborated on tuning Linpack on the cluster and did work such as linking Linpack against the Goto BLAS (basic linear algebra subroutines) library written by Kazushige Goto (www.cs.utexas.edu/users/flame/goto). Additionally, Dell suggested an improved interconnect topology. Prior to the TOP500 run, all 300 nodes were distributed over 16 100Mbit Ethernet switches (Dell PowerConnect 3024). We found that Linpack, like many highly parallel applications, benefits from an improved network interconnect (in other words, one with lower latency and/or higher bandwidth). Dell loaned us a Gigabit nonblocking switch to replace some of our 100Mbit switches.

The above enhancements improved our performance, and we submitted the results for the June 2003 TOP500 list. Iceberg sits at #319 and, in my estimation, could be higher with a faster interconnect.

Figure 2. Running Linpack to Qualify for TOP500

Iceberg Floats to Its New Home

From the beginning, Iceberg's permanent home was to be in the James H. Clark Center, which is named for the man who started Silicon Graphics and Netscape and who is the predominant funding source for the Bio-X Project. By August 2003, it was time to move. The move from the Forsythe Data Center brought many great things, one of which was a clean installation of Rocks. I really pushed for this because having a solid and stable infrastructure is key in maintaining a cluster of this size while maintaining a low total cost of ownership.

Figure 3. Iceberg in its new home. From left to right: Vijay Pande, principal investigator for Folding@home; Steve Jones, Iceberg architect; Erik Lindahl, postdoc; and Young Min Rhee, research scientist.

Through the lifetime of Iceberg at the Forsythe Data Center, we found that both choices on the configuration of software and hardware could be improved. The downtime incurred during the move allowed us to make modifications to the physical design. We decided to go with a front-end node and move the home directories to another node with attached storage. Once again, Rocks came through. It was as simple as using insert-ethers and selecting NAS appliance as the type of node being inserted. We chose to use link aggregation to exploit the dual Gigabit Ethernet network cards in the NAS appliance fully. After a few modifications on the front-end node to connect users to the new appliance and moving the backed-up data to the new appliance, we were operational once again.

Folding@home on Iceberg

We use Iceberg as a debug platform for Folding@home, which is a distributed computing project to study protein folding, misfolding, aggregation and related diseases. Volunteers contribute spare processing time to the project, and currently about 80,000 CPUs are active.

For the Folding@home research study, I exclusively use Iceberg to simulate small projects, where a project is a set of simulations of one protein coupled with a specific method. Writing a script that mimics what Folding@home does with clients was the key to this work. For a run, typically I use 10–20 CPUs at a time.

For other large projects, I used Iceberg for the starting portion only. We usually calculate 10–50ns of simulations in chunks of 1ns. I could use Iceberg for the first 1ns and then move it to Folding@home and continue there. We can rapidly iterate upon new methods in the controlled, stable environment that Iceberg presents. As we develop new projects, we use Iceberg to validate the results, and once we are confident with the new methods, we unleash the new project onto the 80,000-CPU distributed computer.

—Young Min Rhee of the Folding@home Project, folding.stanford.edu

Computational Biology on Iceberg

The Iceberg Dell Supercluster has been a complete revolution to our research. We have been using supercomputing centers for decades, but we always felt limited by relatively restrictive queues, too-small runtime quotas and difficulties in adopting the system to our needs. The last couple of years we have been building our own Linux clusters with 50–100 CPUs, but off-the-shelf hardware leads to higher support and administration costs, not to mention the fact that the cluster is sometimes idling, such as when we are busy writing papers. The new Dell cluster at Stanford is an extremely cost-effective solution to this; by having a shared resource, there are hundreds of nodes available when we want to test a new model overnight, but our cost is only the average number of nodes we are using. We are not only in complete charge of the hardware, but the resources also are an order of magnitude more powerful than anything we've used before or after the installation; we spend only about an hour a week on administration.

Computational biology usually involves extremely heavy calculations on sequences or protein structures. Some of the most common applications involve finding matching patterns between families of sequences or structures and simulating the motions of atoms in biomolecules. Pattern matching it is no big deal for a handful of sequences, but our current project, together with the Department of Energy, relies on large-scale prediction of accurate models for the proteins in the Shewanella bacterium (famous for its ability to eat radioactivity and toxic waste). This involves hundreds of thousands of sequence profiles of which we need to make all-vs.-all comparisons. It used to take weeks of carefully planned runs on our local cluster, but now we literally are able to test an idea overnight and submit a new version of the experiment the next day.

The molecular dynamics simulations of atomic motions are also as impressive. The simple idea is to calculate the forces atoms exert on each other and then use Newton's equation of motion to determine new positions a very short time later (a step is usually 2 femtoseconds, that is 2×10^{-15} of a second). A single iteration in the simulation is fast, but billions of steps are required to study biological reactions. For this reason, we have optimized our code manually to use Intel's Streaming SIMD Extensions instructions, available on the Pentium 4 Xeon CPUs to speed up our programs **Gromacs** and **Encad** by a factor of 2–4, which made it possible to simulate proteins like the Villin headpiece (Figure I) for more than a microsecond, using only two weeks of time on ten of Iceberg's nodes. Actually, with the optimized code, even the individual Dell/Intel Xeon processors are faster than the top-of-the-line IBM Power4 or Alpha CPUs (chips that are found in other supercomputers), at less than one-tenth of the cost. This is by far the best computer investment we ever made, and we will not hesitate to expand it in the future.

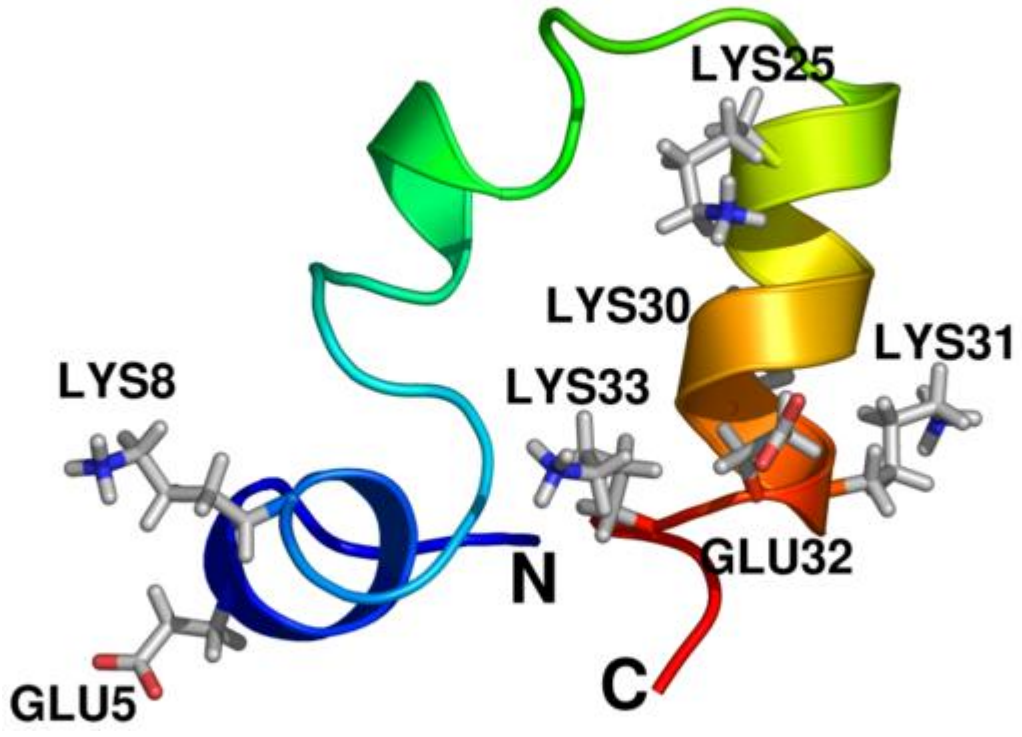


Figure I. The Villin Headpiece

The Villin headpiece is a very small protein consisting of about 600 atoms. However, the cell is always surrounded by water (red/white rods), which brings the atom count up to about 10,000. Every single atom interacts with the closest 100–200 neighbors; the interactions have to be calculated every single step, and then we repeat this for a half-billion steps to generate a microsecond of simulation data. The data for Figure I was generated from a two-week run on ten of Iceberg's nodes.

—Michael Levitt and Erik Lindahl of the Department of Structural Biology, Stanford University School of Medicine

Personal Supercomputer Philosophy

The goal of any high-performance computing cluster should be to bring it to a stable and working state as quickly as possible, and then run it until the hardware dies. This allows the users of the system performing research to have a stable system to use on demand. Making changes to the system for the sake of making changes is not how to handle a computing system of this scale. In my mind, this is what is provided by using the cluster distribution that we've chosen on which to standardize. The only administration work should be monitoring queues, filesystem sizes, reviewing logs for errors and/or activity and monitoring hardware to determine whether replacement is needed. With the combination of Rocks and a support plan from Dell, using Silver support on the compute nodes and Gold on the front end, we will enjoy three years of worry-free performance and parts replacement. Our goal is to generate funds by billing for compute time that will allow for the replacement of the cluster within the three-year period. And when the new cluster arrives, we'll install Rocks, and we will be off and running for another three-year cycle.

The current plan is to charge for usage on Iceberg with the idea that the revenue generated will cover the purchase of a replacement cluster in three years. This replacement most likely will be Iceberg-II, with the same node count, but a 1U form factor rather than 2U in order to maximize our floor space. Iceberg will stay on-line as separate cluster and decrease in size as hardware fails.

Additionally, we are in the planning stages of acquiring a new 600-node cluster that we plan to bring up in the next 6–12 months. This cluster will be housed at an off-site location. We are in negotiations with a research institute that has expressed interest in housing the cluster and providing specialized services (generated power, UPS, power, HVAC and so on) free of charge, in exchange for processing time on the cluster. Other options are being considered as well. Outside the scope of Iceberg, I'm looking into building yet another cluster for commercial purposes. I'd like to think this would be the largest Rocks cluster when completed, in addition to claiming a top-20 seat on the TOP500.

Final Observations

The right mix of an infrastructure person and researchers who actually run jobs on the cluster managing the system, along with empowering interested users in different labs that utilize the system, has been the key to keeping costs at a minimum for maintaining Iceberg. We have delegated tasks now so that firewall management, scheduler configuration, user administration, hardware maintenance and general cluster administration tasks are consuming only a small amount of any person's time. The average time spent administering is an hour a week. This speaks to the design of the system and the choices made in keeping the total cost of ownership of a 302-node supercomputer at the same price as ten nodes.

Steve Jones (stevejones@stanford.edu) left his position at the Stanford Law School in order to pursue a position as a security strategist for an Internet service provider, while consulting for a security startup. He's in the process of moving to Maine where he'll be working as the strategist for another school and starting yet another company. In his spare time, he'll also be managing a 302-node cluster dubbed Iceberg.

2003 Readers' Choice Awards

Heather Mead

Issue #115, November 2003

Thank you to everyone who participated in the voting—now, on to the results.

The 2003 *Linux Journal* Readers' Choice Awards saw the addition of a few new categories, mostly hardware, and the loss of a couple old ones—thank the heavens for built-in pop-up blocking in browsers. Overall, voting was up from last year; more than 7,500 people participated in 2003's four weeks of on-line voting. The results showed a bit of a shake-up in the Favorite Distribution category, but most of last year's winners are back this year.

Favorite Audio Tool

1. xmms
2. noatun
3. mpg123

xmms continues to rule this category, taking the top spot for the third consecutive year. In its first year on the official list, noatun claims second place. Your new favorite write-in choice is mplayer.

Favorite Backup Utility

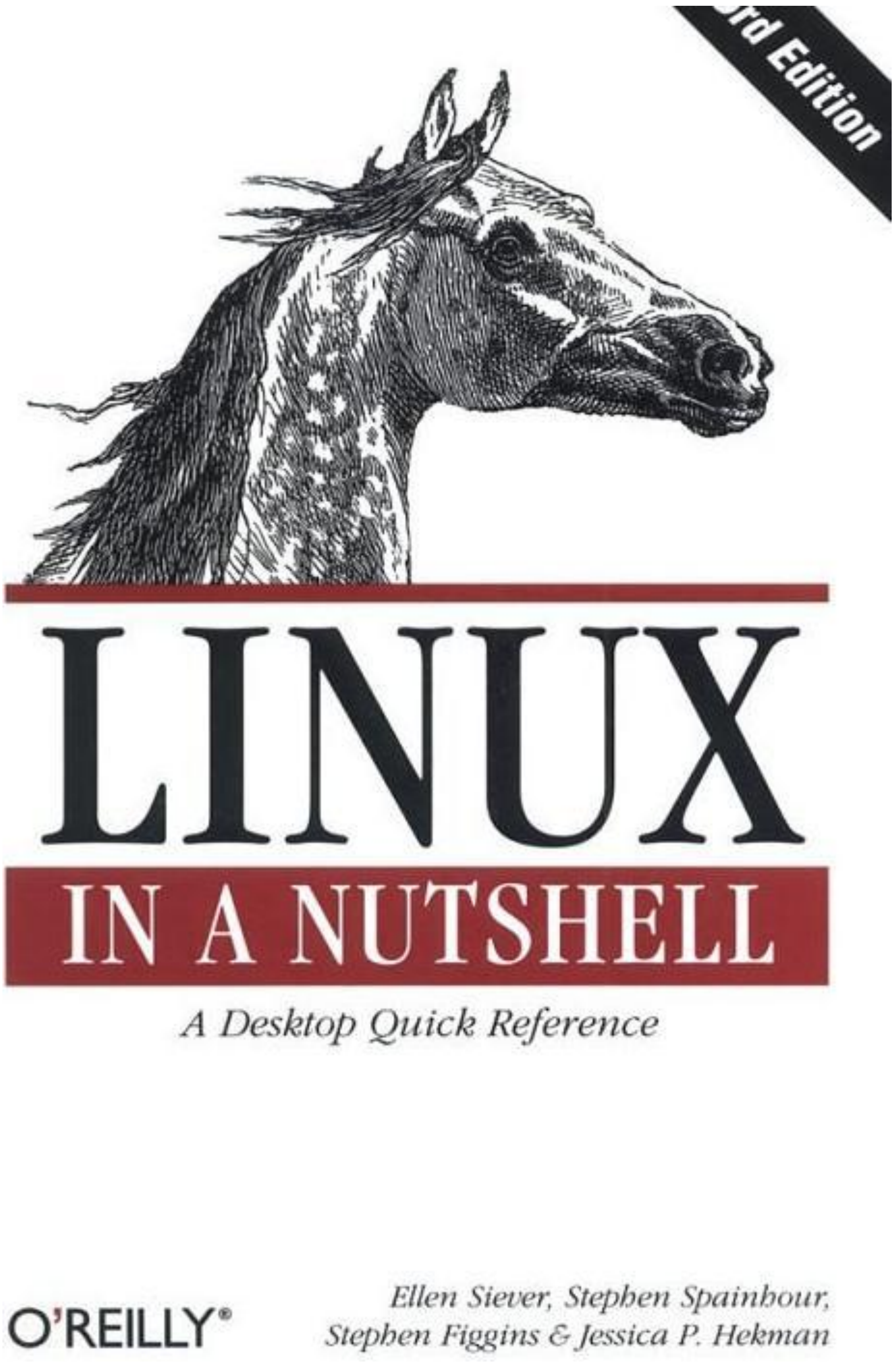
1. tar
2. Amanda
3. Arkeia

Other than the Favorite Workstation category, Favorite Backup Utility seems to be the category least dependent on commercial offerings. In a repeat of last year's winners, tar, Amanda and Arkeia take the tops spots. rsync is the favorite write-in. Thankfully, fewer of you still seem to believe backups are only for wimps.

Most Indispensable Linux Book

1. *Linux in a Nutshell*, 3rd Ed., Ellen Siever, et al.
2. *Linux System Administration*, Vicki Stanfield and Roderick W. Smith
3. *Running Linux*, 3rd Ed., Matt Welsh, et al.

Once again, the top three titles are the same, although *Linux System Administration* and *Running Linux* switched places this year. Many of the purists opt to rely solely on man pages. Judging by the sheer volume of write-ins and new releases that come through the *LJ* offices for review, the Linux book market is picking up speed.


3rd Edition
LINUX
IN A NUTSHELL
A Desktop Quick Reference
O'REILLY®
Ellen Siever, Stephen Spainhour,
Stephen Figgins & Jessica P. Hekman

Favorite Web Browser

1. Mozilla
2. Konqueror
3. Galeon

Here, too, the top three choices are the same this year, with Konqueror and Galeon switching spots in 2003. Netscape's popularity continues to decline; 6.x received a mere 236 out of 7,362 votes in this category. Firebird is making a lot of noise as a write-in, coming in seventh place overall.

Favorite *Linux Journal* Column

1. Cooking with Linux
2. Kernel Korner
3. Paranoid Penguin

Ah, Marcel; *il est un homme savant, gentil and trés drôle*. How can we not respect a man as concerned with raising our sysadmin awareness as with educating our wine palates? Many people also like Kerner Korner and its rotating author bylines—must be all the 2.6 news.

Favorite Database

1. MySQL
2. PostgreSQL
3. Oracle 9i

After a brief respite last year, when InterBase claimed third place, Oracle returns to the top three this year. Although MySQL retains the top spot, PostgreSQL continues to narrow the gap. Firebird, the independent database based on InterBase code, is once again the favorite write-in vote.

Favorite Desktop Workstation

1. Homemade
2. Monarch AMD 2000+ System Special
3. Los Alamos ULBx

Perhaps we should have named this new category Open-Source Junkyard Wars. Although Monarch, Los Alamos, Apple G5, Dell and a few Sun machines received votes, almost 90% of those who voted in this new category selected Homemade as their favorite workstation. Picture it: two teams of four people, $100, two days and a bin of recycled parts—just imagine what we could build.

Favorite Distributed File-Sharing System

1. Gnutella
2. Freenet
3. MORPHEUS

Every time I see Gnutella, I think Nutella, that yummy hazelnut/cocoa spread from Europe. Apparently, Gnutella is almost as good. Freenet cracked the top three this year, pushing audiogalaxy down to fifth position, while spots one and three are a repeat of last year's. eDonkey is all over the write-ins, as are Kazaa, mldonkey and Bit Torrent.

Favorite Distribution

1. Debian
2. Red Hat
3. Mandrake Linux

In a year of major talk of Linux on the desktop and boasts of super quick and easy installation tools, Debian claims first place in the distribution category for the first time ever. Is this a revolt by the hard-core technophiles? Or, is the seductive power of apt-get luring in new recruits?

Favorite Programming Beverage

1. Coffee
2. Water
3. Tea

The beauty of this category is the write-in section, which offers a wondrous and intriguing display of personal tastes. The top spots are always coffee, water and tea. But the write-ins range from vodka to an assortment of microbrews to Yerba Mate and Tang. This year's winner for nastiest-sounding beverage: coffee with cinnamon and red pepper. PS: kudos to the person who correctly spelled Merlot with both an R and a T.

Favorite E-mail Client

1. Evolution
2. KMail
3. Mozilla

Last year's attack on the top three position by GUI e-mail clients holds on for another year. mutt, in fourth place, had only half as many votes as Mozilla. Evolution, in its second year, bested KMail by only 151 votes.

Favorite Embedded Distribution

1. Qtopia
2. MontaVista
3. Lineo

2003 is the second year the embedded category has been a part of these awards, and it received twice as many total votes as last year. Qtopia and MontaVista switched positions this year, and SnapGear Embedded jumped straight to fourth place its first year on the market. The most popular write-in is the Familiar Project for PDAs.

Favorite Linux Game

1. *Frozen Bubble*
2. *Quake 3*
3. *TuxRacer*

More addictive than crystal meth, more entertaining than a Michael Jackson scandal and more dangerous than a lawyer defining “intellectual property”, *Frozen Bubble* jumped straight to number one in its first year on the official nominee list. Will it be replaced next year by *Neverwinter Nights*, this year's first place write-in?

Favorite Graphics Program

1. The GIMP
2. ImageMagick
3. CorelDraw!

The GIMP won again; we know, we were shocked too. More interesting is the number of write-in votes that list Adobe Photoshop used via CrossOver Office.

Favorite Instant-Messaging Client

1. gaim
2. Jabber
3. Kopete

By a three-to-one margin, gaim is once again your favorite IM client. Second and third place went to different clients from last year. Last year's runner-up, Licq, dropped to fifth place with only 388 votes. irssi is the favorite write-in for 2003. And although I don't know a lot about guns, I am fairly confident in saying a 12-gauge Mossberg 500A is *not* an IM client.

Favorite Programming Language

1. C++
2. C
3. PHP

Quick, everyone to your keyboard: the flame war begins in 5, 4, 3, 2....In a reversal of last year's winner and runner-up, C++ moved into first place in 2003 by a mere 23 votes. Perl, meanwhile, got kicked out of the top three for the first time in the history of our awards. Continuing the C theme, C# is the favorite write-in vote.

Favorite Office Program

1. OpenOffice.org
2. AbiWord
3. KOffice

For 2003, we combined Office Suite and Word Processor into a general Favorite Office Program category, which was won handily by OpenOffice.org. Out of 6,650 votes, 4,317 went to OpenOffice.org, followed by 477 for AbiWord. Free, featureful office programs are a good thing. And kudos on the significantly fewer write-ins for MS Office—have you made the switch or are you not talking about it anymore?

OpenOffice.org 1.1
Office Suite

Favorite Processor Architecture

1. AMD Athlon
2. Intel Pentiums
3. AMD Opteron

The frenzy of all things Opteron extends to this year's Readers' Choice Awards, where it premieres as your third-favorite architecture. Last year's third-place choice, AMD Duron, slips down to fifth place, with fourth place going to PowerPC. For the third year in a row, though, Athlon is the top choice.

Favorite Portable Workstation

1. QLi 15" AMD Notebooks
2. QLi Pentium 4 Notebooks
3. QLi Centrino Notebooks

Another category new to this year's awards, Favorite Portable Workstation didn't receive nearly as many votes as other categories. And most of the votes we did receive came as write-ins. Almost everyone selected a laptop as their favorite, although the Zaurus picked up a handful of votes. Other popular write-ins were laptops by Dell and IBM, as well as Apple PowerBooks.

Favorite Network or Server Appliance

1. Cyclades AlterPath ACS

2. CommuniGate Pro
3. SnapGear SME 550

We added this category to the awards for 2003 because of the sheer multitude of offerings on the market. The winners plus the many write-in votes demonstrate the variety of products that can fall under this heading. It seems a lot of you are using Cyclades AlterPath ACS (advanced consoler server) to manage networks remotely. And Linksys routers are near and dear to many voters' hearts.

Favorite Server

1. SGI Altix 3000
2. IBM DB2 OLAP Server
3. Tyan Thunder K8S

It was our snazzy February 2003 Altix cover story, wasn't it? Or, did voters stumble across the big Altix machines on display at LinuxWorld New York 2003 last January? Either way, the love affair with the Altix 3000 is officially underway. Dell, HP, Sun and Compaq split the write-in votes, while some wonder why you'd buy a server when you can build one yourself—homemade was the first choice of the write-in voters.

Favorite System Administration Tool

1. Webmin
2. YaST
3. Ximian Red Carpet

With the plethora of sysadmin tools now available, we figured it was time to give them their own category. Automatic updating tools had the most mentions. Capturing more than 30% of all votes, Perl-based Webmin is the winner. Coming in at a distant second is SuSE's YaST. Many voters, of course, said they need only a command line and vi or vim in some combination to handle their sysadmin tasks.

Favorite Text Editor

1. Vim
2. vi and clones
3. GNU Emacs

Last year, Vim beat vi by twice as many votes; this year it received three times as many. The rest of the top three holds steady this year as the Emacs folks find more and more things they can do with it. But where's the love for Elvis? A mere 14 votes this year is the best we can do? As for Kate, having made the move to the official list this year, it winds up in fourth place.

Favorite Development Tool

1. GCC
2. KDevelop
3. Emacs

Like The GIMP, GCC wins its category every year and always by a sizable lead. You feel almost sorry for their competitors. Although Emacs is in third place again, last year's second-place winner, Kylix, slipped to sixth place this year as KDevelop returns to the top three. Anjuta, which combines Glade, text-editing tools and a simulator in a single IDE, is the favorite write-in this year.

Favorite Linux Training

1. SuSE Linux Training
2. Linux Lunacy Cruise
3. Sun Wah-Pearl Linux OpenLDAP Workshop

We know official Linux training is anathema to some of you (like reading a manual for just about anything), but now more training classes and programs are available than ever before. SuSE's is the most popular distribution-related training, followed by the RHCE program. The write-ins covered everything from man pages and users to self-study and Usenet. We're glad to see so many of you are finding the Linux Cruise useful as well as fun.

Favorite Linux Web Site

1. Slashdot
2. LinuxFR
3. Freshmeat

In the closest race this category has seen in years, Slashdot beat LinuxFR by only 343 votes—pretty impressive when you consider 6,588 people voted. It's nice to see the rampant silliness of this year's freedom fries and freedom toast didn't extend to Da Linux French Page. Still, Slashdot is the first choice for updates on supercomputers, candidates for governor of California and two-hour Cadillac commercials—I mean *Matrix* sequels.

Favorite Web-Hosting Service

1. Rackspace
2. Hurricane Electric
3. eZ Publish CMS

Perhaps this category is too new for regular Linux users to have developed a consensus. Only 893 people voted in this category, and 637 of those votes were write-ins. Rackspace is the only service to garner more than 10% of the vote. Or, perhaps people felt like one voter who said he found it hard to name his favorite because "I love/hate my host!" As usual, many of you take the DIY approach to Web hosting as well.

Favorite Desktop Environment

1. KDE
2. GNOME
3. Window Maker

This category received the most votes of any except Favorite Distribution, which makes sense given the push for Linux on the desktop. With 44% of the votes, KDE is the winner for the sixth consecutive year. GNOME holds on to second place with 23% of the votes. Ion is the new favorite write-in vote, as last year's favorite, fluxbox, comes in fourth in its first year on the official list. Oh good, I was beginning to think I wouldn't see this write-in comment this year, but you didn't disappoint: "None, they all suck!"

Heather Mead is senior editor of *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

Introducing Scribus

Peter Linnell

Issue #115, November 2003

Scribus is a full-featured desktop publishing application that works with a paste-up model, not the typewriter model of word processors. Learn to create industry-standard PDF documents for print.

In contemplating the seemingly eternal question—Is Linux ready for the Desktop? —the recent release of Scribus 1.0 adds one more major reason to say yes. A frequent complaint is “I can't use Linux on the desktop because I am missing application *X*.” Scribus ably fills a missing piece—a graphical WYSIWYG page layout application. Linux users and their *nix cousins now have a versatile and user-friendly desktop publishing application with fresh approaches to the challenges of using Linux as a desktop publishing (DTP) platform. Moreover, Scribus brings other new capabilities beyond DTP with its ability to create PDF (portable document format) Web forms and interactive PDF documents.

Thanks to the increasing polish of important support libraries, like freetype2, Ghostscript and CUPS, Linux desktop publishing is a reality. Scribus cleverly works around some of the potential limitations of Linux and UNIX as DTP platforms, by the extensive and flexible use of PDF as an output file format and to a lesser extent, PDF import. PDF, whose format is copyrighted by Adobe but licensed at no charge for other developers' use, offers flexibility, stability in format and broad application support. PDF is similar to PostScript and is well documented by Adobe; the draft PDF 1.5 reference manual is a slim 1,000+ pages. Plus, PDF is supported on almost every modern computing platform available.

DTP Applications vs. Word Processors

At the most simplistic level, creating documents with Scribus is like grade-school cut and paste. When using the drawing tools, working with Scribus is like having a canvas. When dealing with text and images, it is like using a pasteboard. Composing documents with a word processor is more like working with an intelligent typewriter. Conceptually, it helps to think of Scribus as a pretty face wrapped around a great PDF engine—an engine that greatly reduces the complexity in creating either press-ready high-resolution PDF files or fully interactive PDFs. One of the challenges in creating PDFs, especially press-ready PDFs, is the necessity to know some of almost 100 distilling options in Adobe Acrobat Distiller.

Scribus is most definitely not a word processor; it belongs to the family of applications known as page layout programs. Well known DTP applications include Adobe's PageMaker (the original) or InDesign and Quark XPress. Scribus is unique because it is licensed under the GPL, and no other DTP application with its professional features runs on Linux. Scribus has been ported to the BSDs, HP-UX and Solaris as well. With the help of KDE-Cygwin, an experimental version of Scribus also runs on Microsoft Windows 2000. Work on a Mac OS X version with the GPL'd Qt for OS X is underway as well.

Table 1. A Comparison of DTP and Word Processor Features

Feature	DTP	Word Processor
Layout	Pasteboard	Intelligent typewriter
Automation	Scripting	Macros
Color support	CMYK and RGB	RGB only
Drawing tools	Real PostScript vector	Metafiles or low-res bitmap
Four-color printing	Color management	N/A
Precision	High—up to 3,600 dpi	Adjusts to screen resolution
PDF	Built-in export	Via third-party drivers (OpenOffice 1.1 has a built-in PDF exporter)

A page layout application is different from word processors in the sense that it is concerned with laying out text, images and drawings with a much higher level of precision and control by the designer, which is not possible and sometimes not desirable with modern word processors. Where a word processor excels at processing words, only a page layout program can combine text plus images and other artwork with exactness and ease. Scribus, for example, has many bits of code to optimize PostScript output, control color reproduction, layer images with text and manage high-resolution artwork.

In a page layout application, text is an object, like pictures or shapes, and is contained within frames. This enables precise placement and flow of text on a page. With Scribus you easily can create effects like flipping or running text at angles or auto-flowing multicolumn text. You can use layers or place objects on top of each other, otherwise known as masking, to achieve impressive visual effects. Scribus has special controls for typography to adjust the layout and spacing of individual letters within words, known as kerning. You can control the placement of all objects to within hundredths of an inch or hundredths of a millimeter. Try thinking of a page layout file as a wrapper—the last step in composing a document of any kind destined for print professionally, in-house or a Web downloadable PDF, where object placement, exact color matching and formatting are a necessity.

Some of Scribus' major features include:

- Creating state-of-the-art ISO standard PDF/X-3 conforming high-quality press-ready PDF files for use in commercial pre-press. It is the *only* one to support this directly—a DTP first.
- Creating fully scripted and interactive PDF documents, which include external links, such as Web links and presentation PDFs à la MS PowerPoint or OpenOffice's Impress. You also can create calculated fields and send user-entered form data to a Web site.
- Using Python as a scripting language. Most serious DTP applications are scriptable. Python gives Scribus a uniquely powerful and platform-neutral scripting language. Most other DTP applications use a custom scripting language or AppleScript, which does not cross platforms. This ability also enables you to run third-party Python modules, like the imaging modules, or to run other Python applications, such as PySol.
- Having full support for CMYK (cyan, magenta, yellow and black) color used in commercial printing, including optional color management, color separations and importing spot colors in EPS (Encapsulated PostScript) files.
- Supporting Unicode text and fonts with freetype2, as well as right-to-left languages, like Arabic or Hebrew. At the time of this writing, Scribus has been translated into 19 languages, most recently Czech, Russian and Indonesian.

Getting and Installing Scribus

Installing Scribus is straightforward. You can choose to compile from source or grab one of the third-party packages of Scribus. Details and locations are in the Scribus on-line documentation. Compiling from source is the usual `./configure && make && make install`. You should have the latest updated versions of Trolltech's Qt, freetype2, libtiff, libpng and ghostscript available for your distribution. Scribus 1.0 is optimized for Qt 3.1.2, the latest stable version at the time of this writing. If you wish to enable color management, you need the littlecms libraries and some ICC profiles. Scribus also needs the development libraries for libtiff, libjpeg and libpng. The biggest issue we have seen when compiling is that the QTDIR environment variable either is not set or is set incorrectly. The documentation contains detailed notes about compiling.

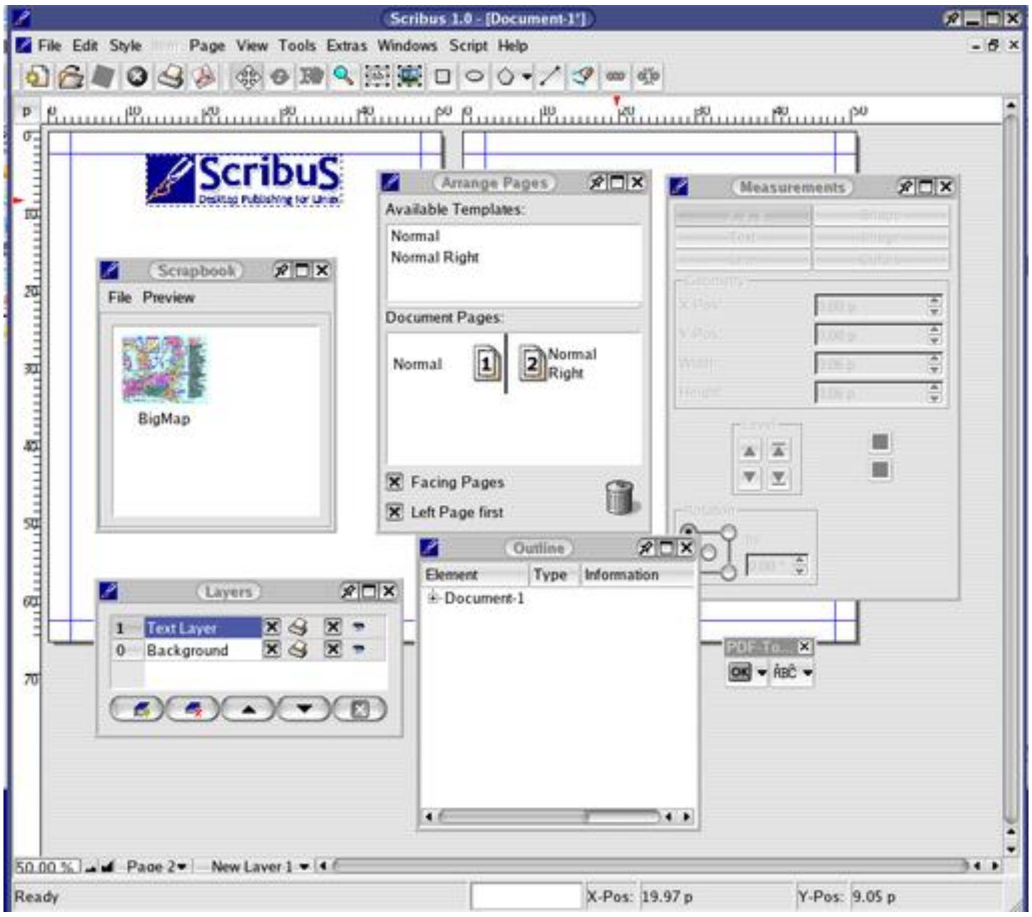


Figure 1. Scribus tools—Scribus has easy-to-use palettes, all of which have tooltips explaining their functions. Clockwise from the top: 1) The Page palette manages pages and page templates. Adding a page is drag-and-drop simple. 2) The Measurements palette provides frequently used editing tools. 3) The PDF toolbar shows how toolbars can be dragged from the desktop. 4) Outline is a tree diagram of every object in the document. This assists in selecting objects in complex documents. 5) Layers allows you to stack objects on top of each other for stunning effects. 6) The Scrapbook stores frequently used items.

Scribus' user interface has been optimized for DTP. You can drag guides right from the rulers. Measurement units can be changed with a simple click. The status bar gives users a lot of information and precise measurements of objects and tools. Although Scribus runs under any window manager, it has special features when run under KDE, such as desktop drag and drop, and uses KDE-style plugins.

Creating Your Masterpiece

Unlike word processors, you do not simply open a new empty document and start typing. First, you must create a text frame and then either start typing or importing some text. My preferred method is to use a text editor, edit, spell-check and then save and import the text. Similarly, you do not open The GIMP, then copy and paste. First, create an image frame then place the image on the page. Image files can be huge, so they are linked externally. This also ensures that the latest up-to-date image is in your document. Keeping track of commonly used objects is made easier with the Scrapbook.

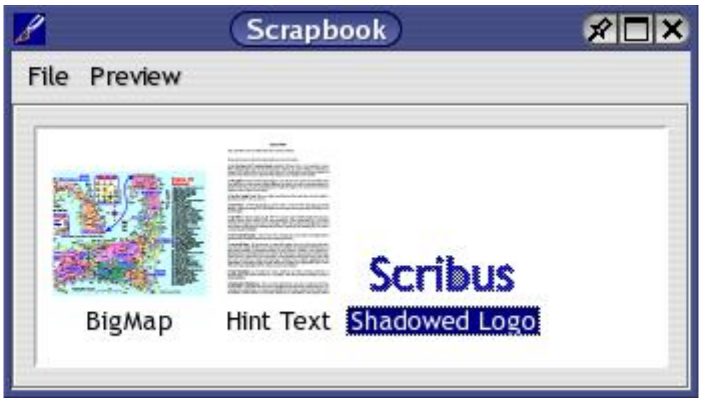


Figure 2. User-friendly—the Scrapbook enables the easy re-use of objects between pages and documents. You can save scrapbooks separately, and they can be saved for projects or single documents. Shown here is an EPS of a full-page map, a full page of text and a logo. Almost any kind of object can be saved by right-dragging it from the page to the Scrapbook.

To help with creating your publication, Scribus comes with a set of useful and user-friendly drawing tools, including styled lines, polygons, Bezier curves, tinting or shading of drawn objects and gradients, which is a special way of blending one or two colors to add shadowing or dimension to an object. Scribus also can manipulate type the way that illustration or drawing programs like CorelDraw or Adobe Illustrator do, by converting fonts into PostScript curves or outlines, which can be twisted, scaled, sheared and stretched at will. You also can fill these with gradients or images and attach text to a path. Scribus also can export SVG, including text and images. All images within SVG files are converted to PNGs.

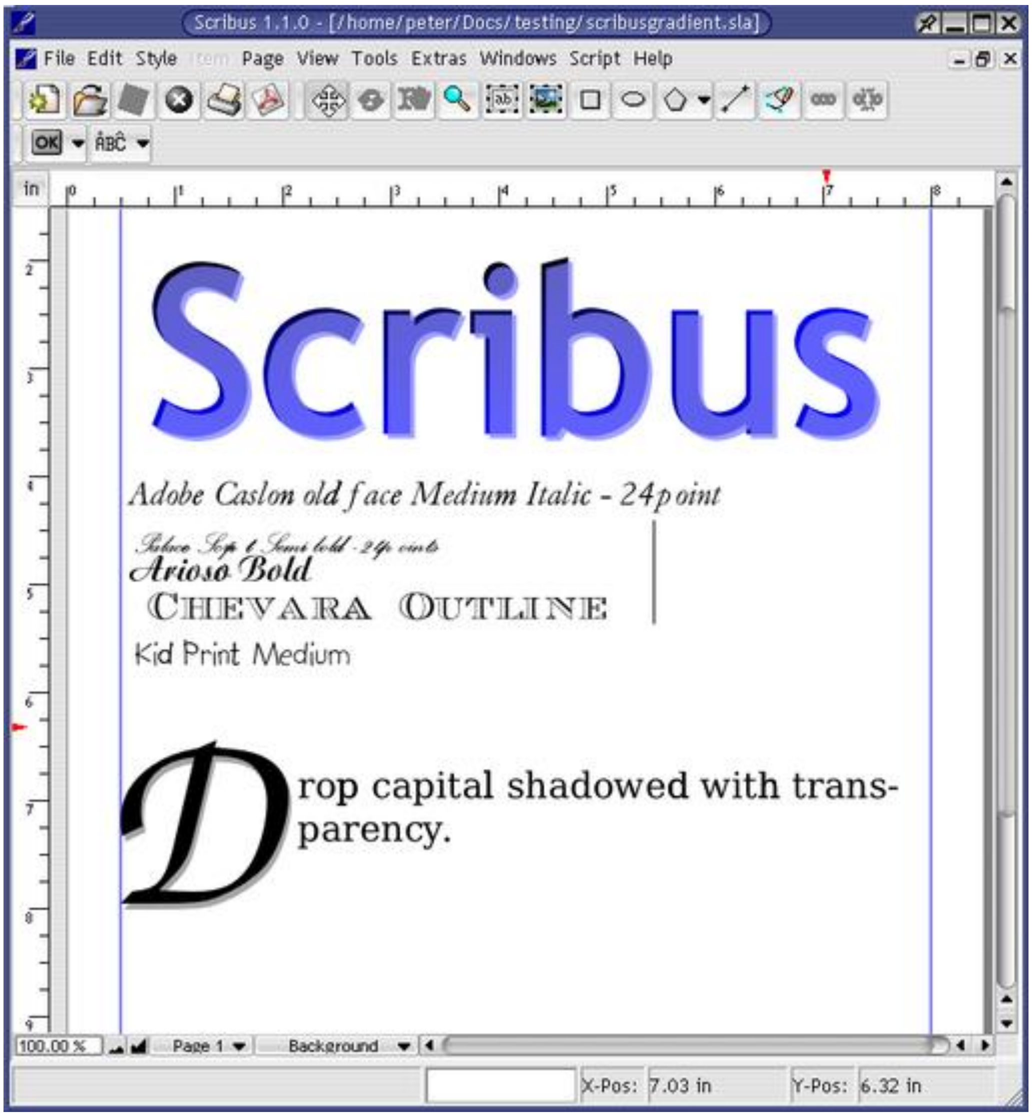

Scribus 1.1.0 - [/home/peter/Docs/testing/scribusgradient.sla]
File Edit Style Item Page View Tools Extras Windows Script Help
Scribus
Adobe Caslon old face Medium Italic - 24point
Arioso Bold
CHEVARA OUTLINE
Kid Print Medium
Drop capital shadowed with trans-parency.
100.00 %
Page 1
Background
X-Pos: 7.03 in
Y-Pos: 6.32 in

Figure 3. Advanced PDF export—Scribus can modify font outlines and create sharp, high-resolution gradients with layers in the object and transparency as well. This kind of output requires advanced Level 3 PostScript and PDF 1.4 operators, previously available only in proprietary DTP applications. (This is a snapshot of Scribus 1.1.0.)

User Preferences

DTP users are accustomed to extensive keyboard shortcuts, and Scribus is no exception. Functions can be operated by user-defined keyboard shortcuts, menu choices and considerable usage of right-click context menus. Font paths can be added or disabled, along with font substitution preferences on the fly. Caution is advised to ensure your fonts are installed correctly with up-to-date fonts.dir and fonts.scale files. Scribus is necessarily quite fussy about font paths and has a lot of code to sniff out the exact capabilities of your installed fonts. A common problem new users find with Scribus is they think that Scribus cannot find fonts that have been installed. To the contrary, Scribus rejects installed fonts that are not 100% in order. The documentation contains an extensive explanation of this. Fonts are mini programs in a sense and have bugs too.

Text Handling

Scribus can import ASCII text files as well as clipboard text. This is enhanced by the ability to specify the encoding of text. Thus, you can place Unicode text, as well as text created in Cyrillic and Latin encodings. You also can switch the direction of text to right-to-left scripts like Arabic and Hebrew. Expect to see additional import filters added in the next versions.

Special font handling features allow you to rotate, flip and scale text. Scribus automatically can hyphenate text in many languages and can allow you to adjust kerning or spacing between letters in a font. If you add AFMs (ASCII font metrics) files for a given font, Scribus uses this to adjust the spacing between letters automatically. Scribus also supports Type 1 and TrueType fonts equally.

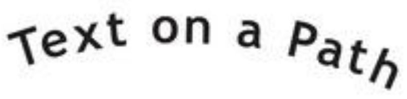


Figure 4. Scribus has one of the easiest to use tools for creating text on a path. This allows users to have text follow an irregular object and can be used for styling text or, for example, naming roads in a diagram.

Image Handling

Scribus can import many common image formats, such as PNGs, TIFFs and JPEGs. All images can be scaled within Scribus, rotated, flipped and layered. If an image file has an ICC color profile embedded, Scribus reads the tags and uses them in the littlecms color management. You easily can import EPS files, as well as the first page of a PDF. To make it easier for you to place EPSes and PDFs, Scribus automatically generates a low-resolution preview. EPS files do not have previews natively.

PDF and PostScript Exporting

Scribus has one of the best PDF export engines on the planet—a bold claim, but in my opinion, second only to the latest version of Adobe's InDesign. The PDF exporter is not only easy to use, but generates high-quality PDFs, as long as you use properly prepared images and choose good quality fonts. Part of the genius in the PDF exporter is the way it masks confusing options but does not constrain the power of the user or the use of advanced features, like PDF/X-3. In testing, I used the same images and fonts to create large high-resolution, four-color CMYK PDF from both Scribus and other high-end DTP applications. The differences were indiscernible. Colors, fonts and text rendered exactly the same, and the images were rendered with the same fidelity throughout. It also fully supports PDF 1.4 features, such as 128-bit encryption and transparency in the PDF. Scribus can export high-quality EPS and raw PostScript files if needed as well.

Power of XML

Scribus documents and preference files are XML-based, completely open and documented. This makes adding features much simpler, and their text-based nature makes them robust. DTP file formats internally are some of the most complex in the PC world. File corruption frequently can be a problem, and DTP files can be unstable on less-than-perfect networks as well. I have found the XML format in Scribus to be almost crash-proof. Even with bleeding-edge CVS versions of Scribus, I rarely have lost a file, and even broken ones can be fixed with a text editor. PageMaker and Quark users have long been used to frequent saves and some novel tricks to repair damaged documents.

Scribus in the Commercial Print World

The design decision early in the launching of the Scribus Project to concentrate on the PDF format was prescient. After many years of false starts, the commercial print world is adopting PDF as the preferred method of file exchange. [*Linux Journal* sends pages to the printer as PDFs—Ed.] It is not uncommon for publications to require PDF files from advertisers. Thus, some magazines are composed entirely of PDFs, with the complete publication re-exported as a PDF. Why? PDF eliminates a host of application and cross-platform incompatibilities, especially with fonts. In addition, mature preflighting tools now exist for verifying PDFs as being press-ready. The emergence of the PDF/X ISO standards also has pushed PDF adoption.

That said, Scribus was not designed to be a “Quark-killer”. Scribus is meant to give Linux and UNIX users comparable tools, which have been, until now, the province of expensive proprietary applications, available only on Mac and Windows operating systems. DTP users and shops can be rather conservative about upgrades and changing applications. Reliability is critical—a missed print run can cost thousands of dollars. Plus, these applications take months, sometimes years to master. Productivity can suffer in the switch-over, even amongst newer versions of the same application.

The support for PDF/X-3 in Scribus is another way to make Scribus files accepted in the print world. The newest raster image processors (RIPs) can support PDF/X-3, preserving the original RGB (red, green and blue) images and ICC profiles until the last moment when the files are converted to printing plates.

Summing Up

As a personal aside, as a Linux newbie and nonprogrammer at the time, volunteering to work on the documentation was one of the best experiences I have had in computing. Coming from the Windows/Mac/Novell world, this gave me a more sophisticated understanding of Linux/UNIX methodologies. Many excellent open-source projects and developers almost always can use help with documentation and testing. For nonprogrammers, this is a small repayment for the many programs we enjoy so freely.

The history of DTP on Linux is, well, brief. In 2000, Adobe publicly beta tested a release of Framemaker that runs on some flavors of UNIX. It then disappeared. For a short time, a company called Chilliware offered a DTP application called Ice Sculptor. The company closed shortly after the release. Although DTP is in some respects a niche application, Scribus brings new reach to the Linux desktop.

Q & A with Franz Schmid

Q: Why did you begin Scribus?

A: I needed a program for Linux to make menus and cards for my in-laws' small hotel in Bavaria, but there was nothing for Linux like the DTP programs that ran on my Mac. I originally wrote this only for myself in Python, until a friend suggested I put it on the Web. I was really surprised with the response.

Q: How did you come up with the name Scribus?

A: I was thinking of calling it something like “Open Page”, but it was not unique enough for me. Scribus comes from Latin for the name of official writers in Rome, like we use the word scribes in English. It makes sense in many languages.

Q: Why did you pick Qt?

A: When I decided to switch to C++, Qt was the only C++ kit with full documentation. Scribus was and is my first C++ project. Python is great for proof of concepts, but it was slow in some functions.

Q: Who is on the Scribus team?

A. Well, at the moment, it is me and Paul Johnson, who has been a member since 0.8 of Scribus. He started the anoncvs, helped with code review and does many other things like supporting users on the mailing list. Peter Linnell joined earlier with testing and documentation. Some valuable contributions have come from other users. Our mailing list is quite active, and I have received some nice e-mails from users who appreciate the quick responses. We have a group of users who are really active on the mailing list, and this helps free me to have time to code. We are probably not normal for open source in that we all are in our 30s and 40s. We all have regular jobs and families. But for DTP, especially, it helps to have some experience and knowledge of the industry.

When to use TeX, when to use Scribus?

For years, a large part of the UNIX/Linux world has equated DTP with TeX and its derivatives—for good reason, in the publication of long technical documents, scientific, mathematic and other text-heavy documents, TeX excels. TeX can create press-ready files as well; entire books have been created with TeX. However, although one can add images and other artwork, TeX is neither intuitive nor efficient for composing highly graphically oriented documents. DTP is actually one of the best uses of WYSIWYG. The methodology of TeX is quite different from visual WYSIWYG DTPs. It's like trying to describe a painting in HTML code—you can do it, but it is not easy. In the Linux world, there is most certainly room and a need for both.

The Secret Sauce of Color Management

One of the advanced features of Scribus is the option to use the littlecms color management libraries. Released under the LGPL, littlecms has become a refined and versatile package for a number of color-related tasks. End users of the profiling tools give it high marks for accuracy and constant improvement.

Proprietary color management methods have been a closely guarded secret until now. How does littlecms accomplish this? First, there exist ISO standards for color and color profile formats that are open and published by the ICC at color.org. Second, to greatly simplify, conversions are made with 3-D lookup tables to convert one color space to another. The secret sauce is the algorithms used by the color management module to adjust the differences between color spaces. The challenge is mapping from one type of color to the other while minimizing the effects of gamut compression. This is the result of the CMYK color space typically having a smaller gamut, or range, of colors that can be rendered by a given device. For example, certain brilliant greens can be created on an RGB monitor but are difficult to render with CMYK colors on a printed page. The thickness, brightness and ink adsorption of the paper also affect printer gamuts. The latest versions of littlecms have something called Black Point Compensation, another trick in adjusting the colors just so, matching as closely as possible the screen and scanner to the final print destination.

Littlecms offers not only a cmm (color management module), but additional utilities that can assist with color-oriented tasks. Some command-line tools allow for embedding or tagging ICC profiles in image files, but also a set of profiling tools can be used to create ICC profiles for your monitor and scanner. A good ICC profile of your monitor is really the first necessity in setting up useful color management in Scribus. My testing of the littlecms monitor profile yields good results for a visual profiler. In high-end DTP, profile creation and calibration is done with special equipment and software, which can run in the thousands of dollars.

Using The GIMP with Scribus

One of the limitations of using The GIMP in the pre-press and DTP world has been the lack of CMYK support. Simply put, The GIMP works in RGB or grayscale. The printing world uses CMYK—the four primary ink colors used in four-color printing. A reading of The GIMP mailing list suggests this might be added to GIMP 2.0, as it has been a long-standing request. Moreover, a lot of discussion has surrounded patents relating to CMYK color. To me, this is a lot of nonsense. As Scribus and the littlecms libraries show, CMYK colors themselves are not encumbered by patents. As long as you do not use patented processes, you are in the clear. Fortunately, a couple of workarounds exist for using The GIMP in the CMYK world: 1) Scribus itself automatically converts RGB colors to CMYK, if the printing options are set to printing in PDF export. 2) Alistair Robinson, who has contributed code to Scribus before, has created a simple, but effective way to output CMYK TIFFs with The GIMP, by cleverly separating the RGB layers and using the gray channel for the CMYK colors. Separate is a GIMP plugin available here: www.blackfiveservices.co.uk/separate.shtml.

There are two other options for CMYK support in Linux: Corel PhotoPaint 9 still is downloadable via FTP. It is free, as in free beer, but unsupported and difficult to install on newer distributions. Caldera Graphics offers both a free “light” version and a commercial version of Cameleo, which includes scanning and image conversion tools. Both have ICC color management support. More detailed information is in the Scribus documentation.

Watch Those DPIs!

One frequent stumbling block for beginners is image resolution. Typically, most images on the Web are 72–96 dpi (dots per inch). When creating DTP files, much higher resolutions often are needed, typically a minimum of 200 dpi or often 300 dpi. For example, this magazine has an optimal resolution of 266 dpi. I have output images in The GIMP as high as 1,200 dpi with excellent results.

Resources

Linux DTP Links: www.atlantictechsolutions.com/scribusdocs/sclinks.html

Littlecms: www.littlecms.com

Scribus Documentation On-Line: www.atlantictechsolutions.com/scribusdocs or home.comcast.net/~scribusdocs

Scribus Home Page: web2.altmuehlnet.de/fschmid or scribus.planetmirror.com

Scribus Mailing List: nashi.altmuehlnet.de/mailman/listinfo/scribus

Peter Linnell is an IT consultant and principal of Atlantic Tech Solutions in New England, specializing in networks, pre-press and DTP. A self-described “Windows refugee”, the Scribus Project is the first open-source project he has worked on. He is working studiously on gaining a Red Hat Engineers' certification to complement his Microsoft and CompTIA certifications. He eschews all claims to geek status, hopefully proven by marrying a lovely French lady, preferring Bordeaux to Jolt cola and traveling to Europe as much as possible.

Archive Index Issue Table of Contents

Advanced search

Writing Secure Programs

Cal Erickson

Issue #115, November 2003

Your new code may be secure, but what about the large project you inherited? Use a tool to find and prioritize possible security issues.

The main focus of most writing about security is network security and physical security. Not much is written about writing secure programs. A lot of what you need to write secure programs is common sense, but due to time constraints and design shortcuts, it is rarely used. Any good programmer knows the concepts but usually does not have the time to implement them; there is a lot of pressure to produce a lot of code and get the project done.

In the early 1970s, the concept of structured programming was alive. Not only was the program structured but the whole project had structure; there were technical specifications, design specifications, detail design specifications, design walk-throughs and code walk-throughs. This made projects bigger and longer, but when finished, the code was debugged easily and often worked with few changes. Some of these projects took many years to produce. However, there was little external influence from networks, the Web and time to market.

Today, a lot of the structured development process has literally disappeared. But security starts with the design of the program or application and depends on coding standards established by the organization where the work is being done.

It is very unlikely that any code will be 100% secure; no code ever is. But, what can be done to make sure code is solid and secure? This article offers some ideas on what to consider and explains three tools to help write secure code. When designing and implementing an embedded system, more care is required for the coding. With the assistance of the tools in the Resources section, a lot of coding errors can be checked. The ultimate judge of secure code is left up to the implementer of the code and the ability of the implementer to understand what is secure.

Check for Errors

Every function returns some type of status, returned either directly or as part of errno. Checking these should be simple. In C++, the exception-handling capability is easy to use but can be complicated to set up. Exception handling has improved greatly over the past few years, once the C++ standard was finalized. When practical, it should be used. Previous practice had been to ignore errors, because it was thought the data being passed was valid. This has been proven to be a bad assumption.

Data buffer overflows have led to many security fixes in the past years. When writing for an embedded system, checking for error returns is important. Decisions need to be made about whether the error is benign and can be ignored. If the error is not benign, maybe it can be corrected. If it cannot be corrected, does the system perform a soft reset or a hard reset? In some cases a soft reset, causing the action in error to be restarted, is all that is required. This is the basis for some fault-tolerant systems. Depending on the type of device, a hard reset may not be a bad thing. Other times, some form of recovery is a must.

Looking for Strings

Instead of using sprintf, strcpy and strcat, use functions like strncpy and strncat. These functions make sure the buffer does not overflow and discard any excess. Do not use fgets when reading data, as this allows overflows. These may seem like simple changes, but they are easy to forget, as is string handling, one of the most exploited areas of programs. Automated test programs check for these problems quite nicely, but the tests can be misleading. Some uses of the string function may be flagged as a problem but prove to be fine in the context where used. This is where the ability and knowledge of the implementer plays an important role. The logs generated by the tools need to be scanned to determine what code has been flagged and needs to be changed.

Memory Leaks and Buffer Overflows

Memory leaks in and of themselves do not necessarily create security risks. However, they can be exploited if the memory is shared by several procedures and structures.

Buffer overflows are by far the most common security issue. If a buffer is allocated on the stack, it can be overflowed to wipe out or change the return address of a function. When a function returns then, it returns to the new address instead of to the proper address. Some buffer attacks also can occur on the heap. These are more difficult to create, but they still can be done.

Programs written in C are most vulnerable to these attacks, but any language that provides low-level memory access and pointer arithmetic can be problematic. Pointer arithmetic is one area that should have bounds checking.

The GNU C compiler has an extension available, which needs to be included when the compiler is built, that implements bounds checking. It is used as an option that adds code to the program. During testing, the code can be turned on and used. During deployment the code would not be present. The reason the code should be turned off is it prints messages when the bounds are breached. If the system in place is a workstation, the messages can be left on, but an embedded system typically has no console.

An idea that might occur here is all the buffers should be statically allocated; then the problem goes away. In truth, the notion that a buffer is of fixed length can be exploited. The data being moved to the buffer still can be longer than the buffer. When it is moved it overflows, and the same problem happens. To lower the risk, the data movement should move up only to the maximum allowed for the buffer. Dynamic reallocation of strings permits programs to handle inputs of arbitrary sizes. The problem with this is the program could run out of memory. On an embedded system, such a mistake is fatal. On a workstation, the virtual memory system may start to thrash and create a performance bottleneck. In C++, the std::string class has the dynamic growth approach. If the class' data is turned into a char * pointer, a buffer overflow can happen. Other string libraries may not have these problems, but the implementer needs to be aware of the limitations.

Validate the Input

If a program is receiving data on which it must operate, there should be some type of validation that the data is correct, does not exceed the maximum size and is free of non-valid types. For instance, if the data is limited to uppercase letters from A to Z, the function should reject anything else. It also should check to make sure the length of the data is valid. Many years ago, everyone thought of data as 80 characters, the size of a punch card. Today, data literally can be any size; it can be text or binary or encrypted. It still has some type of limit though. This should be checked, and if it fails, reject it.

Not only should you check for the maximum size of a record or piece of data but, in some cases, check for a minimum size. Strings should be checked for legal values or legal patterns. If the data being checked contains binary data that needs to be kept that way, it may be better to use the common escape character to signal that the data is binary. If the data is numeric, range checking should be done. If it is any positive integer, check if it is less than zero. If there is a maximum value, check for that. The file limits.h defines the maximum and minimum values for most values, so it is easy to check for system limits.

Help Tools

The dilemma most developers get into is the code already exists, and there is little time and manpower to spend checking for potential security issues. After all, the code is not broken, so why fix it? This attitude prevails in a lot of organizations. Once the code has been found susceptible, however, fixing it becomes a high priority, as does assigning blame.

What can be done to find potential problems short of code inspection? I have learned of three tools that are capable of finding potential problems and flagging them in a report. These tools could be used on an embedded system, but most development is done in a cross-hosted environment. Do the heavy work on the host workstation, and leave the fine-tuning to the target. The information on where to get the tools is listed in the Resources section.

Flawfinder, RATS and ITS4 are three packages that scan the source tree and display a report about potential problems. The display is a list of what is wrong, in which source module and at what line. All of this information also is weighted as to its degree of vulnerability. Listing 1 shows a snippet from a Flawfinder execution on the sample code. The severity level is from 0 to 5, with 0 being very little risk and 5 being high risk.

Listing 1. Flawfinder Example

```
Flawfinder version 1.21, (C) 2001-2002 David A. Wheeler.
Number of dangerous functions in C/C++ ruleset: 127

Examining ../../example_code/msgqueue/mksem.c
../../example_code/msg_queue/msgtool.c:73  [4] (buffer) strcpy:
  Does not check for buffer overflows when copying to destination.
  Consider using strncpy or strlcpy (warning, strncpy is easily
  misused).
../../example_code/msgqueue/mksem.c:34  [4] (shell) system:
  This causes a new program to execute and is difficult to use safely.
  Try using a library call that implements the same functionality if
  available.
../../example_code/pipes/fifo/fifo_out.c:28  [4] (race) access:
  This usually indicates a security flaw.  If an attacker can change
  anything along the path between the call to access() and the file's
  actual use (e.g., by moving files), the attacker can exploit the race
  condition. Set up the correct permissions (e.g., using setuid()) and
  try to open the file directly.
../../example_code/process_control/proc_mem_info/proc_mem_info.c:139
  [4] (buffer) sscanf:
  The scanf() family's %s operation, without a limit specification,
  permits buffer overflows. Specify a limit to %s, or use a different
  input function.
../../example_code/msg_queue/sender/snd_thread.c:70  [3] (random)
  srand:
  This function is not sufficiently random for security-related
  functions such as key and nonce creation. Use a more secure technique
  for acquiring random values.
../../example_code/dlopen/dltest.c:30  [2] (misc) fopen:
  Check when opening files - can an attacker redirect it (via
  symlinks), force the opening of special file type (e.g., device
  files), move things around to create a race condition, control its
  ancestors, or change its contents?
../../example_code/msg_queue/receiver/rcvr.c:51  [2] (buffer) char:
  Statically-sized arrays can be overflowed. Perform bounds checking,
  use functions that limit length, or ensure that the size is larger
  than the maximum possible length.
../../example_code/dlopen/another_dlopen_test/obj.c:15  [1] (buffer)
  strlen:
  Does not handle strings that are not \0-terminated (it could cause a
  crash if unprotected).

...

Number of hits = 139
Number of Lines Analyzed = 5491 in 2.67 seconds (2527 lines/second)
Not every hit is necessarily a security vulnerability.
There may be other security vulnerabilities; review your code!
```

Even though several messages are returned, the implementers can choose to fix or ignore the potential problems. Some developers might argue that these tools should change the code, but it is much better to change code selectively rather than to make wholesale edits un-aided. The Flawfinder program uses an internal database called a ruleset. This ruleset is a list of the common security flaws. These flaws are general issues that can have an impact on C/C++ and a number of specific problematic runtime functions.

Conclusion

Writing secure code can be easy. Thinking about what is being written and how it can be exploited has to be part of the design criteria. Testing methods should be devised to check for various types of attacks or misuse. Fully automating these tests is a luxury that can go a long way to getting a superior product to the consumer. The techniques and tools discussed here are only helpers. The development of secure programs still rests in the hands and minds of the developers.

Resources

Flawfinder, authored and maintained by David A. Wheeler: www.dwheeler.com/flawfinder

ITS4, authored by John Viega, copyright held by Reliable Software Technologies: www.rstcorp.com/its4

RATS (Rough Auditing Tool for Security), authored, maintained and distributed by Secure Software, Inc.: www.securesoftware.com

Splint Secure Programming Lint, maintained by the Secure Programming Group, University of Virginia, Department of Computer Science: www.splint.org

Cal Erickson (cal_erickson@mvista.com) currently works for MontaVista Software as a senior Linux consultant. Prior to joining MontaVista, he was a senior support engineer at Mentor Graphics Embedded Software Division. Cal has been in the computing industry for more than 30 years, with experience at computer manufacturers and end-user development environments.

Archive Index Issue Table of Contents

Advanced search

Kernel Korner

The New Work Queue Interface in the 2.6 Kernel

Robert Love

Issue #115, November 2003

Interrupt latency is a key factor in the performance of a system. Work queues are one of several tools available to the driver writer to avoid doing time-consuming work when interrupts are disabled.

For most UNIX systems, Linux included, device drivers typically divide the work of processing interrupts into two parts or halves. The first part, the top half, is the familiar interrupt handler, which the kernel invokes in response to an interrupt signal from the hardware device. Unfortunately, the interrupt handler is true to its name: it interrupts whatever code is executing when the hardware device issues the interrupt. That is, interrupt handlers (top halves) run *asynchronously* with respect to the currently executing code. Because interrupt handlers interrupt already executing code (whether it is other kernel code, a user-space process or even another interrupt handler), it is important that they run as quickly as possible.

Worse, some interrupt handlers (known in Linux as fast interrupt handlers) run with all interrupts on the local processor disabled. This is done to ensure that the interrupt handler runs without interruption, as quickly as possible. More so, *all* interrupt handlers run with their current interrupt line disabled on all processors. This ensures that two interrupt handlers for the same interrupt line do not run concurrently. It also prevents device driver writers from having to handle recursive interrupts, which complicate programming. If interrupts are disabled, however, other interrupt handlers cannot run. Interrupt latency (how long it takes the kernel to respond to a hardware device's interrupt request) is a key factor in the performance of the system. Again, the speed of interrupt handlers is crucial.

To facilitate small and fast interrupt handlers, the second part or bottom half of interrupt handling is used to defer as much of the work as possible away from the top half and until a later time. The bottom half runs with all interrupts enabled. Therefore, a running bottom half does not prevent other interrupts from running and does not contribute adversely to interrupt latency.

Nearly every device driver employs bottom halves in one form or another. The device driver uses the top half (the interrupt handler) to respond to the hardware and perform any requisite time-sensitive operations, such as resetting a device register or copying data from the device into main memory. The interrupt handler then marks the bottom half, instructing the kernel to run it as soon as possible, and exits.

In most cases, then, the real work takes place in the bottom half. At a later point—often as soon as the interrupt handler returns—the kernel executes the bottom half. Then the bottom half runs, performing all of the remaining work not carried out by the interrupt handler. The actual division of work between the top and bottom halves is a decision made by the device driver's author. Generally, device driver authors attempt to defer as much work as possible to the bottom half.

Confusingly, Linux offers many mechanisms for implementing bottom halves. Currently, the 2.6 kernel provides softirqs, tasklets and work queues as available types of bottom halves. In previous kernels, other forms of bottom halves were available; they included BHs and task queues. This article deals with the new work queue interface only, which was introduced during the 2.5 development series to replace the ailing keventd part of the task queue interface.

Introducing Work Queues

Work queues are interesting for two main reasons. First, they are the simplest to use of all the bottom-half mechanisms. Second, they are the only bottom-half mechanism that runs in process context; thus, work queues often are the only option device driver writers have when their bottom half must sleep. In addition, the work queue mechanism is brand new, and new is cool.

Let's discuss the fact that work queues run in process context. This is in contrast to the other bottom-half mechanisms, which all run in interrupt context. Code running in interrupt context is unable to sleep, or block, because interrupt context does not have a backing process with which to reschedule. Therefore, because interrupt handlers are not associated with a process, there is nothing for the scheduler to put to sleep and, more importantly, nothing for the scheduler to wake up. Consequently, interrupt context cannot perform certain actions that can result in the kernel putting the current context to sleep, such as downing a semaphore, copying to or from user-space memory or non-atomically allocating memory. Because work queues run in process context (they are executed by kernel threads, as we shall see), they are fully capable of sleeping. The kernel schedules bottom halves running in work queues, in fact, the same as any other process on the system. As with any other kernel thread, work queues can sleep, invoke the scheduler and so on.

Normally, a default set of kernel threads handles work queues. One of these default kernel threads runs per processor, and these threads are named events/*n* where *n* is the processor number to which the thread is bound. For example, a uniprocessor machine would have only an events/0 kernel thread, whereas a dual-processor machine would have an events/1 thread as well.

It is possible, however, to run your work queues in your own kernel thread. Whenever your bottom half is activated, your unique kernel thread, instead of one of the default threads, wakes up and handles it. Having a unique work queue thread is useful only in certain performance-critical situations. For most bottom halves, using the default thread is the preferred solution. Nonetheless, we look at how to create new work queue threads later on.

The work queue threads execute your bottom half as a specific function, called a work queue handler. The work queue handler is where you implement your bottom half. Using the work queue interface is easy; the only hard part is writing the bottom half (that is, the work queue handler).

The Work Queue Interface

The first step in using work queues is creating a work queue structure. The work queue structure is represented by struct work_struct and defined in linux/workqueue.h. Thankfully, one of two different macros makes the job of creating a work queue structure easy. If you want to create your work queue structure statically (say, as a global variable), you can declare it directly with:

```
DECLARE_WORK(name, function, data)
```

This macro creates a struct work_struct and initializes it with the given work queue handler, **function**. Your work queue handler must match the following prototype:

```
void my_workqueue_handler(void *arg)
```

The arg parameter is a pointer passed to your work queue handler by the kernel each time it is invoked. It is specified by the data parameter in the DECLARE_WORKQUEUE() macro. By using a parameter, device drivers can use a single work queue handler for multiple work queues. The data parameter can be used to distinguish between work queues.

If you do not want to create your work queue structure directly but instead dynamically, you can do that too. If you have only indirect reference to the work queue structure, say, because you created it with kmalloc(), you can initialize it using:

```
INIT_WORK(p, function, data)
```

In this case, p is a pointer to a work_struct structure, function is the work queue handler and data is the lone argument the kernel passes to it on invocation.

Creating the work queue structure normally is done once—for example, in your driver's initialization routine. The kernel uses the work queue structure to keep track of the various work queues on the system. You need to keep track of the structure, because you will need it later.

Your Work Queue Handler

Basically, your work queue handler can do whatever you want. It is your bottom half, after all. The only stipulation is that the handler's function fits the correct prototype. Because your work queue handler runs in process context, it can sleep if needed.

So you have a work queue data structure and a work queue handler—how do you schedule it to run? To queue a given work queue handler to run at the kernel's earliest possible convenience, invoke the following function, passing it your work queue structure:

```
int schedule_work(struct work_struct *work)
```

This function returns nonzero if the work was successfully queued; on error, it returns zero. The function can be called from either process or interrupt context.

Sometimes, you may not want the scheduled work to run immediately, but only after a specified period has elapsed. In those situations, use:

```
int schedule_delayed_work(struct work_struct *work,
                          unsigned long delay)
```

In this case, the work queue handler associated with the given work queue structure will not run for at least *delay* jiffies. For example, if you have a work queue structure named my_work and you wish to delay its execution for five seconds, call:

```
schedule_delayed_work(&my_work, 5*HZ)
```

Normally, you would schedule your work queue handler from your interrupt handler, but nothing stops you from scheduling it from anywhere you want. In normal practice, the interrupt handler and the bottom half work together as a team. They each perform a specific share of the duties involved in processing a device's interrupt. The interrupt handler, as the top half of the solution, usually prepares the remaining work for the bottom half and then schedules the bottom half to run. You conceivably can use work queues for jobs other than bottom-half processing, however.

Work Queue Management

When you queue work, it is executed when the worker thread next wakes up. Sometimes, you need to guarantee in your kernel code that your queued work has completed before continuing. This is especially important for modules, which need to ensure any pending bottom halves have executed before unloading. For these needs, the kernel provides a function to wait on all work pending for the worker thread:

```
void flush_scheduled_work(void)
```

Because this function waits on *all* pending work for the worker thread, it might take a relatively long time to complete. While waiting for the worker threads to finish executing all pending work, the call sleeps. Therefore, you must call this function only from process context. Do not call it unless you truly need to ensure that your scheduled work is executed and no longer pending.

This function does not flush any pending delayed work. If you scheduled the work with a delay, and the delay is not yet up, you need to cancel the delay before flushing the work queue:

```
int cancel_delayed_work(struct work_struct *work)
```

In addition, this function cancels the timer associated with the given work queue structure—other work queues are not affected. You can call cancel_delayed_work() only from process context because it may sleep. It returns nonzero if any outstanding work was canceled; otherwise, it returns zero.

Creating New Worker Threads

In rare cases, the default worker threads may be insufficient. Thankfully, the work queue interface allows you to create your own worker threads and use those to schedule your bottom-half work. To create new worker threads, invoke the function:

```
struct workqueue_struct *
create_workqueue(const char *name)
```

For example, on system initialization, the kernel creates the default queues with:

```
keventd_wq = create_workqueue("events");
```

This function creates all of the per-processor worker threads. It returns a pointer to a struct workqueue_struct, which is used to identify this work queue from other work queues (such as the default one). Once you create the worker thread, you can queue work in a fashion similar to how work is queued with the default worker thread:

```
int queue_work(struct workqueue_struct *wq,
               struct work_struct *work)
```

Here, wq is a pointer to the specific work queue that you created using the call to create_workqueue(), and work is a pointer to your work queue structure. Alternatively, you can schedule work with a delay:

```
int
queue_delayed_work(struct workqueue_struct *wq,
                   struct work_struct *work,
                   unsigned long delay)
```

This function works the same as queue_work(), except it delays the queuing of the work for *delay* jiffies. These two functions are analogous to schedule_work() and schedule_delayed_work(), except they queue the given work into the given work queue instead of the default one. Both functions return nonzero on success and zero on failure. Both functions may be called from both interrupt and process context.

Finally, you may flush a specific work queue with the function:

```
void flush_workqueue(struct workqueue_struct *wq)
```

This function waits until all queued work on the wq work queue has completed before returning.

Conclusion

The work queue interface has been a part of the kernel since 2.5.41. In that time, a large number of drivers and subsystems have made it their method of deferring work. But is it the right bottom half for you? If you need to run your bottom half in process context, a work queue is your *only* option. Furthermore, if you are considering creating a kernel thread, a work queue may be a better choice. But what if you do not need a bottom half that can sleep? In that case, you may find tasklets are a better choice. They also are easy to use, but they do not run in a kernel thread. Because they are not run in process context, no context switch overhead is associated with their execution; therefore, they may offer you less overhead.

Work Queue Function Reference

Statically create a work queue structure:

```
DECLARE_WORK(name, function, data)
```

Dynamically initialize a work queue structure:

```
INIT_WORK(p, function, data)
```

Create a new worker thread:

```
struct workqueue_struct
*create_workqueue(const char *name)
```

Destroy a worker thread:

```
void
destroy_workqueue(struct workqueue_struct *wq)
```

Queue work to a given worker thread:

```
int
queue_work(struct workqueue_struct *wq,
           struct work_struct *work)
```

Queue work, after the given delay, to the given worker thread:

```
int
queue_delayed_work(struct workqueue_struct *wq,
                   struct work_struct *work,
                   unsigned long delay)
```

Wait on all pending work on the given worker thread:

```
void
flush_workqueue(struct workqueue_struct *wq)
```

Schedule work to the default worker thread:

```
int
schedule_work(struct work_struct *work)
```

Schedule work, after a given delay, to the default worker thread:

```
int
schedule_delayed_work(struct work_struct *work,
                      unsigned long delay)
```

Wait on all pending work on the default worker thread:

```
void
flush_scheduled_work(void)
```

Cancel the given delayed work:

```
int
cancel_delayed_work(struct work_struct *work)
```

Robert Love is a kernel hacker involved in various projects. He is a mathematics and computer science student at the University of Florida and a kernel engineer at MontaVista Software. He can be reached at rml@tech9.net.

Archive Index Issue Table of Contents

Advanced search

At the Forge

Server Migration and Disasters

Reuven M. Lerner

Issue #115, November 2003

Every site is different and needs an individual disaster plan. Develop your own disaster plan with some rules for recovering from catastrophic failures.

Experience counts for quite a bit in the world of system administration. Sysadmins scarred by bad hardware, devastating software problems and security break-ins are more likely to put together effective backup strategies, security policies and disaster recovery plans. The question is not whether a disaster will happen to your servers but when the disaster will occur and what form it will take.

I'm writing this in mid-August 2003, less than one week after I moved my own server (lerner.co.il) to a new virtual co-location facility. And, much of this column was written in blacked-out New York City, where I was planning to spend several hours at business meetings—and ended up getting to experience firsthand a large-scale technological disaster. Oh, and when I wasn't moving my server or sitting in the dark, I was without Internet connectivity for the better part of a week, as I was moving to a new apartment in Chicago.

So this month, we take a brief pause from our discussion of Bricolage and other content management software and instead consider how to handle server migrations and disaster plans for Web/database sites. Of course, every site and server is different and deserves to be given individual attention for the best possible planning. But with a little forethought, it shouldn't be too difficult to move your server from one location to another, to handle catastrophic hardware or software failure or even to run in the face of large-scale disaster, as the entire northeastern United States experienced this summer.

Server Relocation

I have moved my server several times over the last few years, and each experience has gone more smoothly than its predecessor. To be honest, moving to a new machine does not need to be difficult or painful, but it does need to be planned carefully. Every step needs to be taken with the assumption that you will need to roll it back at some point.

The simplest possible type of server to move from one machine to another is a static Web site or one that uses basic CGI programs. In such cases, you need to ask only a few questions:

- Does the Apache configuration include the modules that you use? If you are a heavy user of mod_rewrite or if you enjoy the benefits of mod_speling (yes, with one l), you should double-check to ensure that these modules are available. If they were compiled into your server statically, running `httpd -l` lists them. If they were compiled as dynamic modules (DSOs), however, you should check in the libexec subdirectory of your Apache installation, where available DSOs are placed. Each DSO can be loaded optionally by including an appropriate LoadModule directive in the Apache configuration file.
- Under what user and group does Apache run? System administrators have different opinions regarding the user and group IDs under which Apache should run. Some use the default, running it as the nobody user. Others (like me) prefer to create a special Apache user and group, adding users into the apache group as necessary. Still others use the suexec functionality in Apache, compiling it such that it can run as one or more users. In any case, be sure your chosen Apache user/group configuration on your new server is set up in /etc/passwd and /etc/group, as well as in Apache's own config files.
- Where is the DocumentRoot? By default, Apache assumes that DocumentRoot is in /usr/local/apache/htdocs. This default, which can be changed with the DocumentRoot directive in the Apache configuration file, depends on the operating system or distribution you are running. If you are using an RPM version of Apache, as is the case with Red Hat-style distributions, the DocumentRoot might be in /var/www or in another directory altogether. This should not affect the URLs within your programs and documents, but you should double-check the directory into which you're copying the files before assuming that the destination is the right place.
- On what languages and modules do your CGI programs depend? If your site uses CGI programs, then at least one of those programs probably depends on an external module or library of some sort. CGI.pm, the Perl module for CGI programs, has been included in Perl distributions for a

number of years, but it continues to be updated on a regular basis. So, if you depend on features from the latest version, you must double-check. This goes for other modules that you use; one client of mine was using an old version of Perl's Storable module and discovered (the hard way) that upgrading to the latest version broke compatibility when communicating with legacy systems.

DNS

The linchpin of any server migration is the process of moving DNS records. Although people prefer to use names, such as www.lerner.co.il, network connections use numeric IP addresses, such as 69.55.225.93. Translating the human-readable names into computer-usable numbers is the role of DNS, the Domain Name System. Intelligent manipulation of DNS records is a critical part of any server transfer.

The main problem with DNS is not host-to-IP translation but, rather, the fact that DNS results are cached. After all, you want to avoid a DNS request to your server for each HTTP request that someone makes. Such requests would place undue load on your server and would unnecessarily delay the servicing of HTTP requests.

So when you make a DNS request, you're not actually asking the original, authoritative server for an answer. Rather, you're asking your local DNS server for an answer. If it can provide one from its cache of recent results, it will do so without turning to the main server. In other words, `nslookup www.lerner.co.il` executes a DNS request against your ISP's DNS server. That server might return a result from its cache, or it might turn to the authoritative server for the lerner.co.il domain.

When you move a server from one machine to another, then, you want to reduce the TTL (time to live) setting on the DNS server to a low number, so that DNS servers caching this information do not return false answers. I've found that reducing the TTL to 300 seconds (five minutes) is more than adequate. Once the system has been migrated completely, you can increase the TTL to a more typical value, such as six hours, to reduce the load on your DNS servers.

If you are moving your HTTP server from one provider to another, here is an outline of what you can do to have a successful migration:

- Make sure a DNS server at your new provider is able and willing to serve DNS (forward and backward) with your current IP addresses and hostnames. That is, the DNS server at your new provider should point people to your old provider. Set the TTL to five minutes.

- Update the WHOIS records for your domain, indicating that your new provider is the authoritative DNS server. It may take one or two days for this to filter through the entire DNS system. If your new DNS server is providing results identical to your old one, the only ways to tell if things have worked are to perform a WHOIS lookup or to use `nslookup -type=ns yourdomain.com`.
- Once the WHOIS records have been updated, start moving things over. Make sure that all of the software you need is configured correctly, that all modules are set correctly and that the DNS servers have been updated. If your new DNS server isn't responding to queries for your domain, you will be in deep trouble when the WHOIS records point to the new server as an authority.
- When everything seems to be identical (running rsync from the old system to the new one is a good way to ensure that it is), switch the DNS definitions such that the hostname resolves to the new IP address, rather than to the old one.

Depending on the type of server you're running, you might want to turn off the HTTP server on the old system to reduce some of the confusion that might occur as a result of the switchover. For example, switching off the old HTTP server before you switch DNS ensures that the log files do not have any overlap, allowing you to append them together and use Webalizer or Analog to look around appropriately.

At this stage, everything should be working correctly on the new system. But, you should check as many links as possible, particularly those that invoke CGI programs, server-side includes and nonstandard modules or those that require unusual permissions. As always, your HTTP server's error log is your best friend during this process; if and when things go wrong, you can consult the error log to see what is happening.

Databases

Of course, all of the above assumes that you're working on a relatively simple site. Most modern Web sites, however, include relational databases somewhere in the mix for a variety of reasons. They are used for consistency and flexibility, as well as for development speed and the incorporation of a commonly used paradigm that's easy to debug and use.

Relational databases store information in one or more tables, commonly grouped into a database. (Yes, it's somewhat confusing to say that a database server contains one or more databases, each of which in turn contains one or more tables, but that's the way it is.) To move a database from one system to another, you need to move both the database schema (the table, view and function definitions) and the data itself. Of course, a database typically is owned by a particular database user (who generally is not the same as a UNIX user) and has specific permissions set on it.

If your Web site incorporates a database, you thus need to move that database from the old system to the new one, including all of the owner- and permissions-related information. How you do this depends on the database you're using and if any tricks are involved.

MySQL databases that use ISAM/MyISAM tables (the default and still the most popular option) simply can be copied from one MySQL system to another. Typically, all of the files associated with a database are in a directory with the name of the database under /var/lib/mysql. You thus can copy the foo database by copying the directory /var/lib/mysql/foo, including any files it might contain, into /var/lib/mysql/foo on the new system. (Be sure the database is shut before performing this operation.) Start up the server on the new system, and it should work fine.

Things aren't quite this easy with PostgreSQL, which keeps track of database schemas and data in a low-level binary format. Using tar or rsync to copy a PostgreSQL database is highly unlikely to work—and if it does work, it probably involves massive data corruption and might even crash the back-end database server. Instead, you should use the pg_dump tool, which allows you to turn a PostgreSQL database into a set of CREATE, INSERT and COPY statements in plain-text format. For example:

```
pg_dump -U mydb mydb > /tmp/mydb-dump.txt
```

The `-U mydb` portion indicates that we want to use the mydb database user. You might need to substitute a different user name. You then can put this dumped output into a working database with:

```
$ createdb -U mydb mydb2
$ psql -U mydb mydb2 < /tmp/mydb-dump.txt
```

Following these two commands, two databases (mydb and mydb2) are available, both owned by the mydb user.

MySQL makes it easy to handle such situations because of its built-in master/slave system. One database can be the master, accepting all SQL commands and queries, and another can follow along, allowing you to replace the master with the slave in case of catastrophic failure.

In Case of Blackout…

As I mentioned earlier, I was “lucky” enough to be in New York City during the massive power outage that affected millions of people in the United States and Canada this past August. Ironically, the outage occurred about an hour after I toured a potential client's server farm, where he demonstrated how his company backs up all its data to a remote location in Connecticut. (After all, what are the odds of something affecting both New York City and Connecticut? At least, that's what I was thinking while nodding in approval an hour before the blackout.)

If you're like me, the blackout didn't affect your servers, which are located outside of New York. And most co-location facilities have backup generators that can handle most or all of the facility's power needs in case of an emergency.

But, if your servers are located in your office or if you use only a simple UPS to ensure they continue running, the servers probably would be unavailable during a blackout like the one that we saw in mid-August, which lasted more than 48 hours in some parts of the Northeast. If your server is critical to your business, you might seriously want to consider moving it to a co-location facility.

But even co-located servers crash and go off-line; I can tell you this from personal experience over the past few years. This means that if you depend on your server, you should be backing it up on a regular basis. Moreover, you should be migrating it continuously to a server at another physical location, preferably hosted by a different company. But the differences should stop there—you want the software configurations to be as similar as possible. If you use rsync for all of the HTML pages, templates, software libraries and CGI programs on your server, and similarly automate a database dump and restore on the second server, the second server should be a close-to-exact replica of the first and should be available to go live at a moment's notice.

You even can go one step further and use both servers simultaneously. Of course, this is a far more difficult task, as it requires you either to use a single database server (creating a single point of failure) or to synchronize the databases at frequent intervals. But it's definitely possible, particularly on sites that are largely static, as we know from the success of Akamai, which has a hugely redundant array of servers all over the world. The more static the site, the easier it is to replicate and keep up and running.

This is still one advantage that commercial database software, such as Oracle, has over PostgreSQL and MySQL. Although the latter offer master/slave replication to varying degrees, the synchronization is neither as advanced nor as robust as what Oracle offers. Over time, however, I expect this situation will change, as such architectures grow increasingly common.

Conclusion

Moving to a new place is hard, and moving your server to a new location or computer also is hard. But by coming up with a good migration plan, moving incrementally and checking your work at every point with tools such as nslookup, dig, telnet and the HEAD and GET programs that come with Perl's LWP (or the curl toolkit that performs similar operations), you can have a smooth and simple migration.

With only a few changes, a migration plan also can be used as a backup plan, ensuring that your servers continue to work and are accessible even in the wake of a great disaster. You cannot plan for every potential pitfall, but if your organization depends on its Web site, it's worth investing the time and money to ensure that it remains on-line.

Reuven M. Lerner, a longtime Web/database consultant and developer, is now a first-year graduate student in the Learning Sciences program at Northwestern University. He lives with his wife and two young daughters in Chicago, Illinois. You can reach him at reuven@lerner.co.il.

Archive Index Issue Table of Contents

Advanced search

Cooking with Linux

Diners, Start Your Processors!

Marcel Gagné

Issue #115, November 2003

Our Linux chef's favorite racing games range from the unpredictable to the super-realistic. Get ready to burn rubber!

It's true, François, I feel exactly as you do. Although I realize the theme of this issue is high-performance computing, when I think *high performance*, I think *race cars*. In a strange way, *mon ami*, the link is a powerful one. After all, what pushes the boundaries of computing performance like great 3-D simulation? Think of it—high-performance racing driving high-performance computing. One might call that a delicious, perhaps intoxicating relationship, *non*?

Ah, just in time. Our guests have arrived, François. Welcome, *mes amis*, to *Chez Marcel*, home of fine Linux fare, exceptional wines and edge-of-your-seat racing action. Please sit, and make yourselves comfortable. I hope you enjoy the décor today. I had François paint racing stripes on all the tables and chairs in honor of this high-performance computing issue.

François! To the wine cellar, *immédiatement*! We need something to excite the senses. As I recall from my earlier quality-control tour of the cellar, the 1999 Margaret River Chardonnay from Australia is certainly exciting and has enough spirit to go the distance.

While we wait for François to return with the wine, I should tell you that every item on tonight's menu requires an accelerated 3-D video card and the appropriate XFree86 drivers, including the Mesa 3-D development libraries for compiling. All the systems here in the restaurant are ready, but if you need some information on setting up 3-D acceleration on your home Linux system, see my *Linux Journal* May 2003 article, “Battles inside the Computer”, for some information on direct rendering and testing your card's performance.

The first car racing simulation I remember wasn't on a computer. It was a simple, electric slot car track. The action on the figure-eight strip of black plastic was exciting. Although it was three dimensional—after all, nothing is 3-D like reality, *non*? —it was from an overhead viewpoint, a kind of overhead 3-D. That's the spirit behind the first item on tonight's menu, Harry Storbacka's *Race*.

To get *Race* up and running (or to get yourself up and running in *Race*), you can either download the static binary from the Web site or build from source. Both are available from the *Race* Web site at race.sourceforge.net. Obviously, the easiest thing to do is extract the binary package, but should you decide to build it from source, make sure you have the clanlib, xml2 and ode development libraries. After extracting the package, it simply should be a matter of running **make** as indicated below:

```
tar -xzvf race-0.9.0-src.tar.gz
cd race-0.9.0
make
./race
```

The installation is less than graceful (at least for now). I found that I had to play with the Makefile (specifically to deal with the path to my xml2 libraries), so running the available static binary certainly is much easier. Extract the source (`tar -xzvf race-0.9.1-0-static-linux.tar.gz`), change to the directory and run `./race-0.9.1-static`. The game starts by letting you choose a few settings, including the track. You also can click Continue until the race starts. As I mentioned, the view now is from above. If you are slow on the old gas pedal, the other cars on the track will start pushing on you. The action has a kind of twisted realism to it. As the tires spin, smoke starts to rise from their tires. Press A (think accelerate) to start moving. The cursor keys turn the wheels left and right.

After spinning out on the curves a few times, I was delighted to learn that such small details were remembered in the game. When I came around for my second lap, the skid marks still were on the road. It's a cool effect.

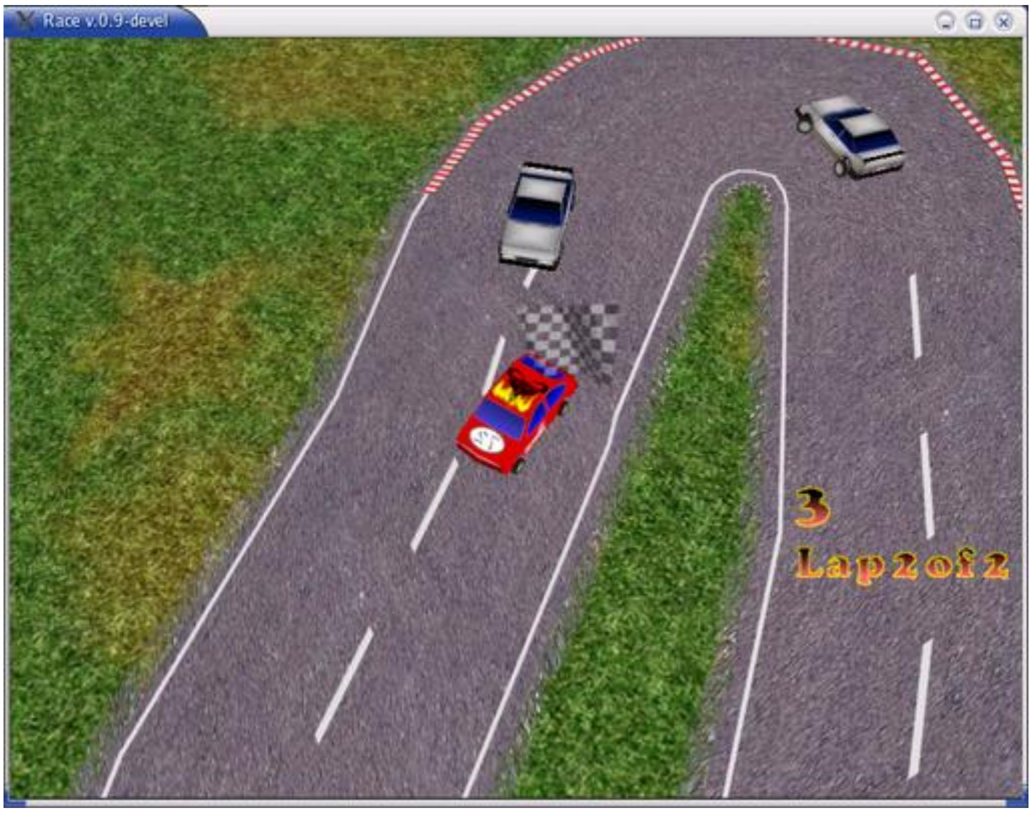


Figure 1. *Race*'s game memory remembers where you skidded out.

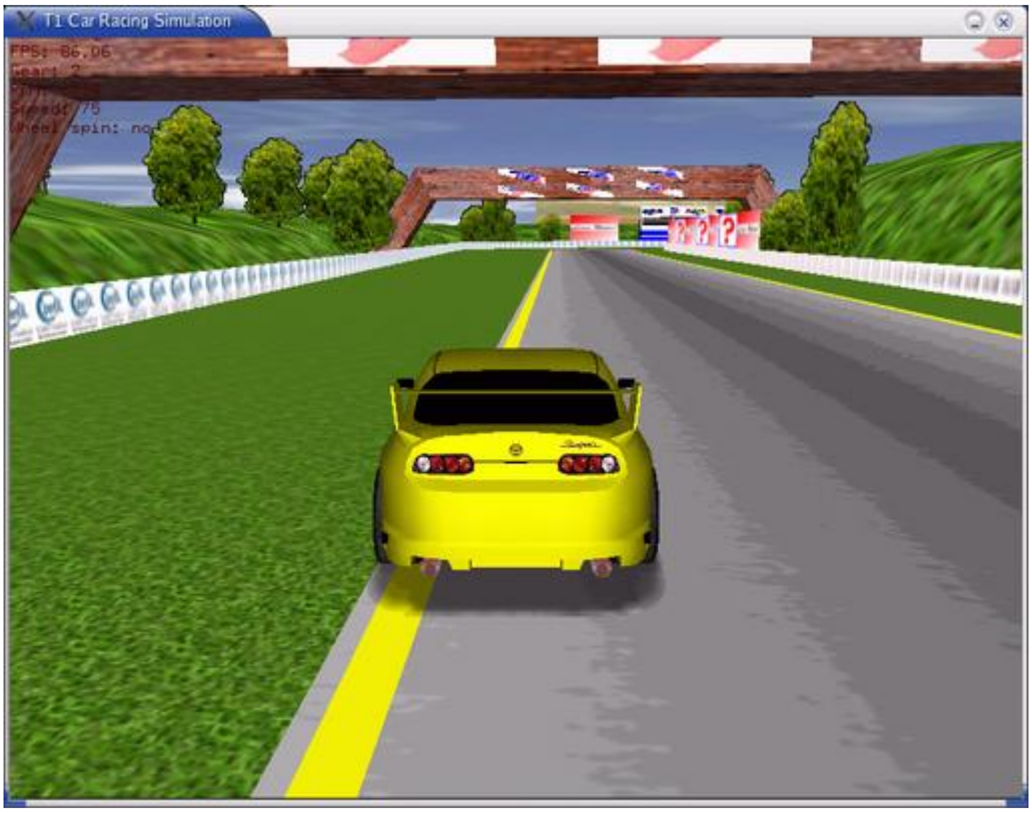
The real thrill of racing starts when you get behind the wheel of a car (even virtually), which explains the excitement and attraction of sit-down racers at your favorite arcade. Out there in the Linux world, you'll find a number of simulators of this type. Some are quite mature and professional, but as with the real world, cars and engines are always under development, pushing the envelope as they try to eke out a few extra revs. So it is in the world of open-source development. I'll show you a couple of these now.

One particularly promising entry is Alex Pozgaj's *T1 Car Racing Simulation* (t1-crs). As of this writing, the game was listed as alpha. It still was a lot of fun, though perhaps not totally playable; nevertheless, it shows great promise. If you would like to take it for a spin (in Alex's Toyata Supra), visit the *T1* Web site at t1-crs.sourceforge.net. Armed with the source, follow these steps:

```
tar -xzvf t1-crs-0.1.2a.tar.gz
cd t1-crs-0.1.2a
./configure
make
```

There's no install as of yet. To play the game, stay in the build directory and type `src/t1_crs`.

Cursor keys control left and right motion as well as the gas and brake pedals. Make sure you gear up before starting or you may find yourself going backward. The letters Q and A on your keyboard let you gear up and down, which is absolutely necessary if you want to start moving.



```
FPS: 86.06
Gear: 2
[illegible]
Speed: 75
Wheel spin: no
```

Figure 2. Trying to Stay on the Road with the *T1 Car Racing Simulation*

The authors of the next item on tonight's menu, foobar and judeo, call their creation *OpenGL Race Game*, but I'll call it *Canyon Racer* to differentiate it from the other OpenGL race games featured today. *Canyon Racer* is another game currently in development that is nonetheless a lot of fun to play. With a hint of flavor from *Star Wars*' pod racers, this game puts you on a futuristic, floating vehicle careening along canyon walls. The action is fast and a little wild as you try to stay inside those walls. Above and to the left you'll find a partial map alerting you to upcoming turns. I must confess, *mes amis*, I had enough trouble avoiding the walls, never mind looking at the map.

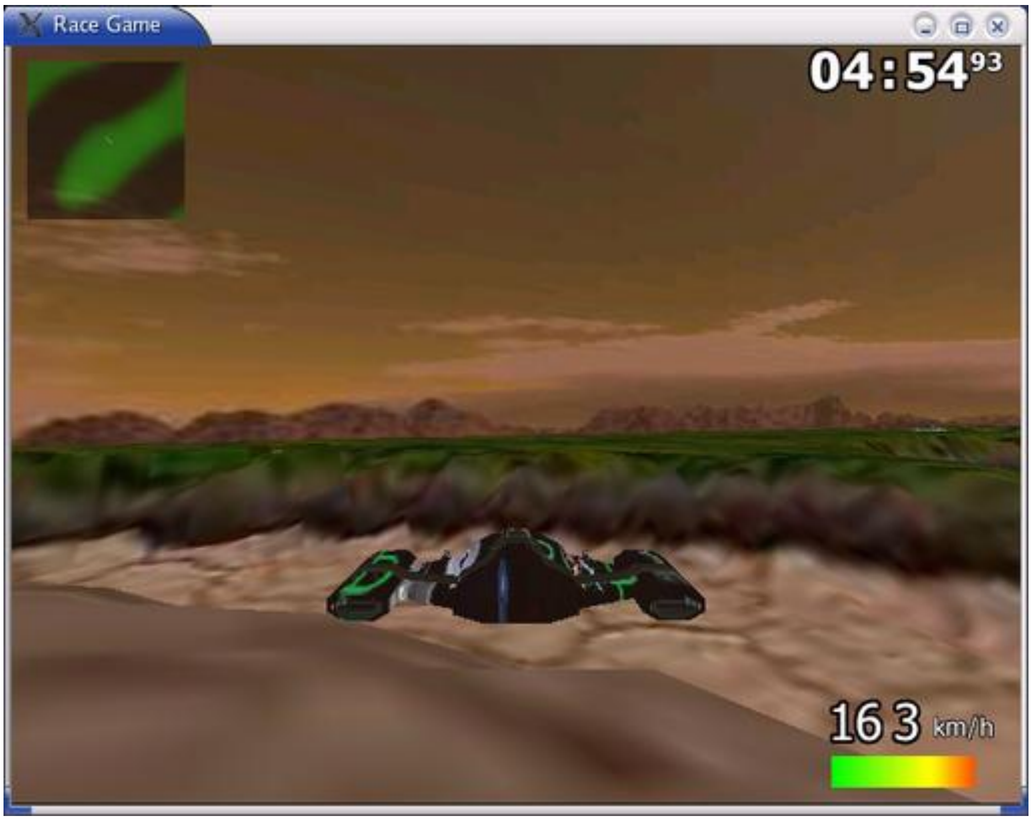


Figure 3. The Futuristic *Canyon Racer*

For a copy of *Canyon Race*r, go to the Project Z Web site at projectz.ath.cx/?id=70 and pick up the source. As with basically all of these games, you do need a 3-D accelerated video card. To build the game, you need the OpenGL and SDL (mixer and image) libraries. With the necessary prerequisites in place, the rest is easy:

```
tar -xjvf racer-0.5.tar.bz2
cd racer-0.5
make
```

Because there is no install script, game play starts from the build directory. Type `./race`, and you are on your way. Movement in the game is through the keyboard. The easiest thing is to use the cursor keys, but there are letter equivalents as well. Pressing W moves you forward; S is reverse, and A and D are left and right. Pressing the spacebar puts on the brakes. To play full screen, press the F1 key.

As mentioned, the developers consider this a game in its early stages (anyone want to help them?), but it is fun. The game's development nature shows up when you lose control and fly over a canyon wall into the *non-world*.

My favorite race game (and the most advanced in this roundup), is *TORCS*. The *TORCS*' Project leader Eric Espié and his team have put together a mature and advanced race simulation with beautiful graphics, photo-realistic scenery, real-time action and a lot of different cars (more than 40 at the time of this writing). If you find yourself getting bored with *TORCS*, perhaps it's time for you to get into the action. *TORCS* lets you program your own cars, robot opponents and race tracks as well. This is a game for the serious racer.

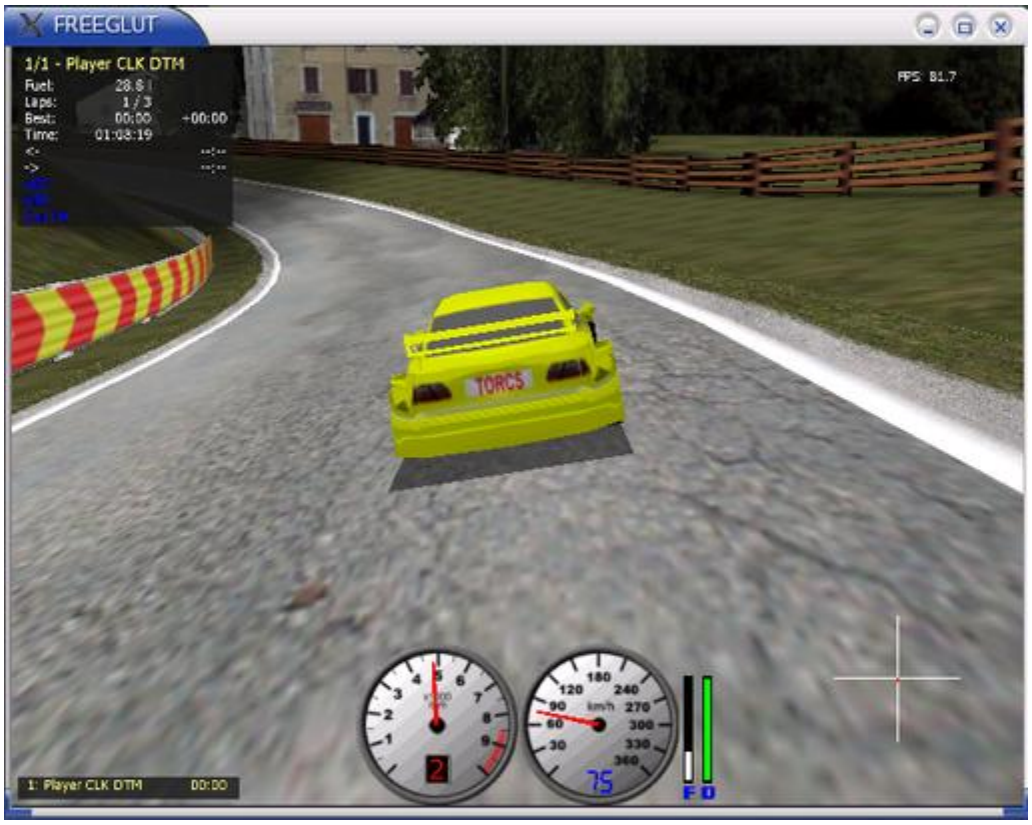


Figure 4. Taking to the Open Road with *TORCS*

For a copy of *TORCS*, visit torcs.sourceforge.net. The site provides binary packages for Red Hat, SuSE, Mandrake, Debian and others, as well as source (the software is GPL'd, after all). For those who really want to live on the high-performance racing edge, CVS downloads also are provided.

Building from source is pretty standard stuff, but there are a number of prerequisite libraries for 3-D development (namely Mesa and GLUT), as well as plib. The easiest way is to download one of the binary packages. Should you go that route, make sure you get everything you need. At the very least, pick up the base TORCS and TORCS-data packages. Although this is all you really need to get started, download and install some TORCS-robots, TORCS-data-cars and the TORCS-data-tracks-base packages as well. This gives you robot opponents to race against, some seriously cool places to race and that wide selection of cars I was telling you about.

You can start *TORCS* by typing `torcs`. The first thing you'll see is a simple screen offering a single-player race and an option for setup. If you are impatient, head straight for a single-player race, but you will want to come back and tweak some of those setup options. *TORCS* works with keyboard, mouse or joystick access, and the setup lets you tune these choices. From the setup, you also can change your player name, choose a car or track, select the type of transmission (manual or automatic) and so on. Even under the basic race menu, you can make choices, such as what kind of car you want to drive and where you want to race. I personally like being at the wheel of that red Ferrari on the Alpine track.

The action is fast and guaranteed to make your heart race. Onscreen displays show your position in the race, including current lap, as well as the status of your opponents (you sometimes pass them sitting in the ditch). It also has the standard speedometer, tachometer and fuel gauges. The top right-hand corner shows a constant frames-per-second readout, giving you an idea of your graphic card's performance, a fascinating mirror to your virtual race car's performance (my NVIDIA card ran about 75fps on average). The physics of the game are great as well. As with the overhead race game I covered at the beginning, too fast a turn sends you careening and leaving skid marks, and if you find yourself on a slope without any forward acceleration, the car starts to roll downhill.

Time has screamed by tonight, *mes amis*, and already I see we are approaching the checkered flag and closing time. Still, there is always time for a final glass of wine and a final race, *non*? François, if you would be so kind as to do the honors. Have another sip of wine, *mes amis*, then set down your glasses and get ready to put the pedal to the metal. Until next time, *mes amis*, let us all drink to one another's health. *A vôtre santé Bon appétit!*

Resources

Harry Storbacka's Race: race.sourceforge.net

OpenGL Race Game (aka *Canyon Racer*): projectz.ath.cx/?id=70

T1 Car Racing Simulation: t1-crs.sourceforge.net

TORCS: torcs.sourceforge.net

Marcel's Wine Page: www.marcelgagne.com/wine.html

Marcel Gagné (mggagne@salmar.com) lives in Mississauga, Ontario. He is the author of the newly published *Moving to Linux: Kiss the Blue Screen of Death Goodbye!* (ISBN 0-321-15998-5) from Addison Wesley. His first book is the highly acclaimed *Linux System Administration: A User's Guide* (ISBN

0-201-71934-7). In real life, he is president of Salmar Consulting, Inc., a systems integration and network consulting firm.

Paranoid Penguin

Secure Mail with LDAP and IMAP, Part I

Mick Bauer

Issue #115, November 2003

Set up a secure, scalable mail system that uses your existing LDAP server to authenticate IMAP users connecting from anywhere.

In the September 2003 issue, I ended a series on building an OpenLDAP server. With this current column and the next, I discuss in-depth one of LDAP's most compelling applications: providing authentication and address book information for IMAP users. These aren't more LDAP articles, though. The focus is on IMAP itself (Cyrus) and how it can leverage LDAP and its own security features to provide secure remote mail services. In other words, assume you already have (or know how to get) an LDAP server running and populated with user accounts.

Delivery vs. Transport Agents

First, a little background on IMAP's role in the e-mail food chain. IMAP, the Internet Message Access Protocol (specified in RFC 3501), is a protocol for mail delivery agents (MDAs). Whereas mail transport agents (MTAs), such as Postfix and Sendmail, move mail between networks, MDAs move mail from MTAs to destination mailboxes. To use a simile from my book *Building Secure Servers With Linux*, if an MTA is like a mail truck that moves mail between post offices, an MDA is like a letter carrier who delivers mail from the local post office to your house.

An IMAP-based MDA system has two parts: an IMAP server, which houses user mailboxes and receives mail from some MTA, and a group of users running IMAP client software. The three most popular open-source IMAP servers are University of Washington IMAP (UW IMAP), Cyrus IMAP from Carnegie Mellon University and Courier IMAP from Inter7 Internet Technologies. Popular IMAP client applications include Netscape/Mozilla Communicator, Ximian Evolution, Microsoft Outlook Express, KMail, mutt, pine and Apple Mac OS X Mail. IMAP clients are beyond the scope of our purposes here, but they're relatively easy to configure and use. Furthermore, most IMAP clients interoperate with most IMAP servers, so there isn't much to explain anyhow.

Which IMAP Server?

When building an IMAP system, the first choice an e-mail administrator must make is which server to use. What are the major differences between UW IMAP, Courier IMAP and Cyrus IMAP? Because features are added to the latter two fairly frequently, I'm going to cop out of telling you the answer in much detail. What I can say is:

1. Of the three, UW IMAP is the least flexible, as it supports only local-user-account mail file delivery; each local user's inbox is stored as a single flat file, /var/mail/myusername. This has two disadvantages: each mail user also must be a system user, and only one process may write to any given user's inbox at any given time, potentially resulting in file-locking complications.
2. Courier IMAP, actually part of the Courier Mail Server, was designed to support qmail's maildir system. In it, users have their own mail directories that store messages as individual files, which is better both from a performance standpoint and for obviating file-locking problems. Courier also can store mail in databases (see point 3); recent versions of Courier IMAP also support LDAP authentication.
3. Cyrus IMAP can be more complicated to set up than UW IMAP or Courier IMAP, mainly due to the Cyrus SASL authentication libraries on which it depends. However, Cyrus IMAP uses its own user and mail databases, both completely separate from the underlying OS, which allows you to add mail users without adding system user accounts. Also, using databases rather than flat files to store messages has an obvious performance benefit.

Personally, I've used Cyrus IMAP the most, so it is the MDA this article discusses. Refer to the features lists on the respective home pages of UW IMAP, Courier IMAP and Cyrus IMAP (see Resources) to decide which is the best fit for your environment. If your choice is different from mine, I hope some of the concepts in the rest of this article still are helpful to you.

Getting and Installing Cyrus IMAP

As you know, I'm a big fan of binary packages due to the version control and patch management features provided by a good package manager. To my thinking, the major distributions' package managers all are quite good. Accordingly, I recommend you install Cyrus IMAP from your distribution's update service or installation media if at all possible. You also need Cyrus SASL, an authentication back end Cyrus IMAP requires. SMTP AUTH also uses this, so you already may have it installed.

Thus, in SuSE 8.2 the RPMs you need are cyrus-imapd and cyrus-sasl2. In Debian 3.0, you need the deb packages cyrus-common, cyrus-imapd, libsasl2 and sasl2-bin. Both SuSE and Debian users should be aware that earlier versions of your respective distributions may have Cyrus SASL packages based on old (pre-v2.0) versions of Cyrus SASL. The method of authenticating Cyrus IMAP against LDAP that I'm about to describe depends on SASL v2.0 or later, however. If your distribution version has a pre-2.0 SASL package, you may need to obtain and compile Cyrus SASL source code (available at ftp.andrew.cmu.edu/pub/cyrus-mail).

For Red Hat 9.0, you have to do a little more work than you do for the latest versions of SuSE or Debian, because Red Hat hasn't provided Cyrus IMAP packages since Red Hat 7.1. You should install the RPMs cyrus-sasl, cyrus-sasl-plain and cyrus-sasl-md5, which are part of the standard Red Hat 9.0 distribution. But, you need to get Cyrus IMAP itself in the form of an SRPM from home.teleport.ch/simix (graciously maintained and provided by Simon Matter in Switzerland).

If you've never dealt with source RPM (SRPM) files before, don't worry: the command to build a binary RPM from an SRPM is simply `rpm --rebuild [--target` *`yourarch`*`]` *`srpm.name.src.rpm`*, where *srpm.name.src.rpm* is the name of your SRPM file, and *yourarch* is your machine's architecture (such as i386, i586, i686). For example, when I ran this command on my Pentium III server I used `rpm --rebuild --target i686 cyrus-imapd-2.1.12-7.src.rpm`. Although the --target setting is optional, if you're going to have a large IMAP user database, optimizing Cyrus IMAP for your CPU type reportedly yields noticeable speed improvements over the default i386 build.

rpm then automatically compiles several new binary RPMs, customized for your local system architecture. These RPMs are written into /usr/src/redhat/RPMS/ (the precise subdirectory being whatever you specified after `--target` or i386/ by default). These RPMs are cyrus-imapd, cyrus-imapd-utils, cyrus-imapd-devel and perl-Cyrus. Install them with the `rpm -Uvh` *`filenames`* command.

Configuring SASL

We have two goals for the remainder of this article: to leverage our existing LDAP server to authenticate IMAP users and to configure our Cyrus IMAP server to accept only SSL-encrypted connections from end users. In past articles, I've extolled the virtues of centralizing authentication, so hopefully the value of using LDAP for this step is a given by now. I may have explained in earlier columns how dangerous clear-text e-mail retrieval is. In normal POP3 and IMAP transactions, your user name, password and all subsequent e-mail data traverse the network unencrypted. Therefore, they are exposed to eavesdropping attacks, especially if you retrieve your e-mail over a wireless network or from the Internet, the largest untrusted network of them all.

Back to SASL. Because Cyrus IMAP and Cyrus SASL both come from Carnegie Mellon University, and because the Cyrus team is understandably reluctant to reinvent the wheel, Cyrus IMAP depends on Cyrus SASL for its authentication functionality. This may seem confusing: isn't this function what we're using LDAP for? Yes it is, and SASL is indeed redundant insofar as SASL was designed to use *its own* user database to authenticate users.

Besides using its own database, however, SASL also can be used to broker authentication transactions with other authentication sources, such as PAM or LDAP. The simplest way to do this is by configuring saslauthd, the SASL authentication dæmon, whose behavior is controlled primarily by the file /etc/saslauthd.conf. Listing 1 shows a sample saslauthd.conf file.

Listing 1. Sample /etc/saslauthd.conf

```
ldap_servers: ldap://localhost/
ldap_search_base: dc=wiremonkeys,dc=org
ldap_bind_dn: cn=servers,dc=wiremonkeys,dc=org
ldap_bind_pw: password_goes_here
```

ldap_servers specifies a space-delimited list of LDAP server URIs. In Listing 1, I've specified a clear-text LDAP connection to the local LDAP process. I instead could specify the encrypted ldaps protocol instead of ldap, a remote fully qualified domain name or IP address instead of localhost or both, as in ldaps://ldap.wiremonkeys.org.

ldap_search_base is the base (shared) part of your users' distinguished names (DNs). ldap_bind_dn and ldap_bind_pw are the DN and password you wish saslauthd to use to connect to your LDAP server. I recommend creating a special LDAP record for this purpose. In Listing 1, servers is the name of a special LDAP account with an objectClass of simpleSecurity Object, which means that besides its DN and objectClass, it has only one other attribute, userPassword.

If nothing else, having a dedicated server account in LDAP means that in your LDAP logs you can distinguish LDAP lookups by back-end processes and servers from end-user-initiated queries. This would be harder if IMAP used, for example, your personal LDAP account. For still more granular auditing, you even could use a different LDAP account for each service that performs LDAP queries, for example, cyrus and postfix.

Those are the options I use in my own /etc/saslauthd.conf file, but they aren't the only ones available to you. Cyrus SASL is distributed with a file, LDAP_SASLAUTHD, that documents these and other saslauthd.conf options. It's located in the source code distribution's saslauthd directory, but if you install SASL from a binary package, this file is placed wherever your distribution puts package documentation, probably some subdirectory of /usr/share/doc.

Besides editing /etc/saslauthd.conf, you also need to make sure saslauthd is started with the -a ldap option. On Red Hat, this is done by editing the file /etc/sysconfig/saslauthd so the parameter MECH is set to ldap. On SuSE, you edit the same file, but the parameter is called SASLAUTHD_AUTHMECH. Other distributions using sysconfig may have a different parameter name, and/or you may need to customize /etc/saslauthd. Again, the desired end result is for saslauthd to be passed the option -a ldap on startup.

Once you've configured and restarted saslauthd, you're ready to configure your IMAP service. As it happens, this is the easy part.

Configuring Cyrus IMAP

Most of Cyrus IMAP's behavior is controlled by a file named, predictably, /etc/imapd.conf. Listing 2 shows a sample imapd.conf file.

Listing 2. Example /etc/imapd.conf File

```
configdirectory: /var/lib/imap
partition-default: /var/spool/imap
admins: cyrus wongfh
sievedir: /var/lib/imap/sieve
sendmail: /usr/sbin/sendmail
hashimapspool: true
sasl_pwcheck_method: saslauthd
sasl_mech_list: PLAIN
tls_cert_file: /var/lib/imap/slapd3.pem
tls_key_file: /var/lib/imap/slapd3key.pem
tls_cipher_list: HIGH:MEDIUM:+SSLv2
```

As you can see, many of the options in imapd.conf simply define paths to various things Cyrus IMAP needs. I won't cover these in detail (see the imapd.conf(5) man page for complete documentation), but let's at least discuss the nondefault settings in Listing 2.

admins: specifies the Cyrus IMAP users who may administer the IMAP system using the cyradm tool. (We'll cover cyradm in a future column.) By setting sasl_pwcheck_method: to saslauthd and by already having configured saslauthd to use LDAP, we've configured Cyrus IMAP to use LDAP for *all* authentication. So even though, for example, the user cyrus may exist on the local Linux system (in /etc/passwd), cyrus also needs to have an LDAP entry. When you run cyradmin and are prompted for cyrus' password, provide the password defined for Cyrus in the database, not cyrus' Linux password, if indeed the Linux account has one. In other words, any account names you specify after admins: must exist on whatever user database is specified by sasl_pwcheck_method.

When you installed Cyrus IMAP, whether from binary packages or from source code, a new user (cyrus) should have been created and given ownership of most Cyrus IMAP files. As with any other good service dæmon, Cyrus IMAP runs as a special nonprivileged user rather than root most of the time.

The three other settings in Listing 2 that I had to customize were tls_cert_file:, tls_key_file: and tls_cipher_list:. These are analogous to OpenLDAP's slapd.conf parameters TLSCertificateFile, TLSCertificateKeyFile and TLSCipherSuite, respectively. I mention this because the certificate/key files specified here are the same ones I used for OpenLDAP on this system. I did this because, in my example scenario, I'm running Cyrus IMAP on the same server on which I'm running OpenLDAP; there's no reason to use different server certificates and keys for each service. I did, however, copy both files from /etc/openldap to /var/lib/imap to simplify ownership/permissions management.

If my LDAP service was running on a separate host, I would create a new TLS certificate/key pair for my LDAP server, using exactly the same procedure I described in my August 2003 column:

```
openssl req -new -x509 -nodes -out slapdcert.pem \
-keyout slapdkey.pem -days 365
```

Regardless, remember to make both your certificate and key file owned by cyrus and your key file readable *only* by its owner.

If you install Cyrus IMAP from source, it uses default SSL keys that fail if an IMAP client attempts to connect using TLS rather than SSL encryption. Aside from the reliability issue, it's never, ever a good idea to use default (placeholder) certificates or keys for anything. Either leverage a server certificate/key you've created already (if applicable) or create a new pair. Your IMAP server will be both more reliable and more secure.

That's it; Cyrus IMAP now can be restarted (`/etc/init.d/cyrus-imapd restart`) and users added with cyradm. We'll have to save that part, plus getting your local MTA to deliver mail to IMAP, for next time.

Resources

Courier IMAP Home Page: www.inter7.com/courierimap.html

Cyrus IMAP Home Page (source, documentation and so on): asg.web.cmu.edu/cyrus/imapd

The Exchange Replacement HOWTO, an excellent reference for using Cyrus Imap with LDAP: www.arrayservices.com/projects/Exchange-HOWTO/html/book1.html

UW IMAP Home Page: www.washington.edu/imap

Mick Bauer, CISSP, is *Linux Journal*'s security editor and an IS security consultant for Upstream Solutions LLC in Minneapolis, Minnesota. Mick spends his copious free time chasing little kids (strictly his own) and playing music, sometimes simultaneously. Mick is author of *Building Secure Servers With Linux* (O'Reilly & Associates, 2002).

Archive Index Issue Table of Contents

Advanced search

EOF

Extreme Linux: Not All That Far Out There

Jason Pettit

Issue #115, November 2003

Your next 512-processor Linux box might be a NUMA system, not a cluster.

Not much longer than a year ago, common Linux wisdom said the operating system scaled reliably only to eight or maybe 16 processors. This widely held belief that Linux somehow was limited in scalability deterred many users from considering it as the underlying operating system for their largest enterprise and most complex technical applications. It was a disappointing limitation for those who wanted to take advantage of the benefits of open-source software on every platform in their environments.

What a difference a year can make. In that time, a variety of advances made by the Linux development community and system vendors has enabled a whole new class of scalable computers running Linux. Today, 16-, 32- and even 64-processor systems running Linux are installed around the world, solving complex problems and running a variety of high-end enterprise applications.

However, to members of the high-performance computing (HPC) community, even 64-processor scalability isn't enough. Commonly needing to harness the computing power of hundreds and sometimes thousands of processors to solve their basic problems, HPC users still are asking for more. Many computational models have resource requirements and communications complexities in excess of what today's clusters adequately can address. HPC users who have adopted Linux have had to accept sub-optimal cluster performance. The fact remains that some hardware solutions available today have scalability that far exceeds what the current standard Linux kernel supports. Also still hanging around is this question: Will systems based on Linux ever scale to the point where they can address the unique needs of this demanding user community?

SGI believes the answer is yes, and it introduced the Altix 3000 line of servers and superclusters in January 2003. The Altix systems have a non-uniform memory access (NUMA) architecture, and they currently offer 64-processor Linux scaling. Over 100 Altix systems are installed worldwide in a variety of configurations, many deployed as multinode systems in configurations with more than 128 processors. For example, The Netherlands Organisation for Scientific Research installed at SARA has 416 processors. With an underlying architecture that can support much more than the current 64-processor limit, it should come as no surprise that Altix customers already have been asking how they can help scale Linux further.

In response to this customer demand, SGI is working to scale Linux to 128 processors running a standard 2.4 Linux kernel by working with the Linux community and a variety of prestigious HPC sites. The global effort, announced at LinuxWorld San Francisco in August 2003, includes the Pacific Northwest National Laboratory, the University of Queensland in Australia, the Computing Center at Johannes Kepler University in Austria, the US Naval Research Lab (NRL) and the NASA Ames Research Center. The group plans to assist with the development and testing of 128-processor functionality and demonstrate early progress toward its goal at the annual Supercomputing tradeshow in November 2003. The next goal is to have a fully supported product by early 2004.

In addition, SuSE also has joined the effort and is evaluating whether to include the work in a future release of the company's SuSE Linux Enterprise Server product. This achievement would break yet another glass ceiling for Linux and open-source software.

What might be surprising is the group's belief that scaling Linux to 128, 256 or even 512 processors won't be that difficult. The basis for this conclusion is founded partly on the flexibility of the Altix architecture and the experience SGI and the users have had with the company's IRIX operating system, which runs on nearly identical Origin hardware.

For example, the NASA Ames Research Center previously has worked with SGI to build systems to model the Space Shuttle and to perform other complex research. It currently has a 1,024-processor single system image Origin installed. It is uncertain whether Linux can ever reach that level of scalability, but the prospects are encouraging. NRL, the first site to test the code changes that support 128-processor scalability, is enthusiastic about the early results. Meanwhile, working from a strong hardware foundation and experience, SGI engineers believe Linux can achieve much more and are enthusiastic about the outcome of simulations that scale the operating system well beyond 128 processors.

“Clearly the community has already done most of the group's work to make Linux very capable. Scalable systems simply provide the magnifying glass that exposes minor problems in the operating system that need to be fixed to deliver optimum performance”, said Jack Steiner, a principal engineer working on Linux scalability at SGI.

The work will be given back to the community, and ultimately the scalability work being undertaken should benefit all Linux users as it is incorporated into community kernels. At the current pace of testing, less than a year from now people will be saying, “You know, Linux scales reliably to only a few hundred processors.”

Jason Pettit has been working with Linux systems since 1998. As product manager for Linux and Altix 3000, Jason is leading SGI's charge to scale Linux to even greater heights.

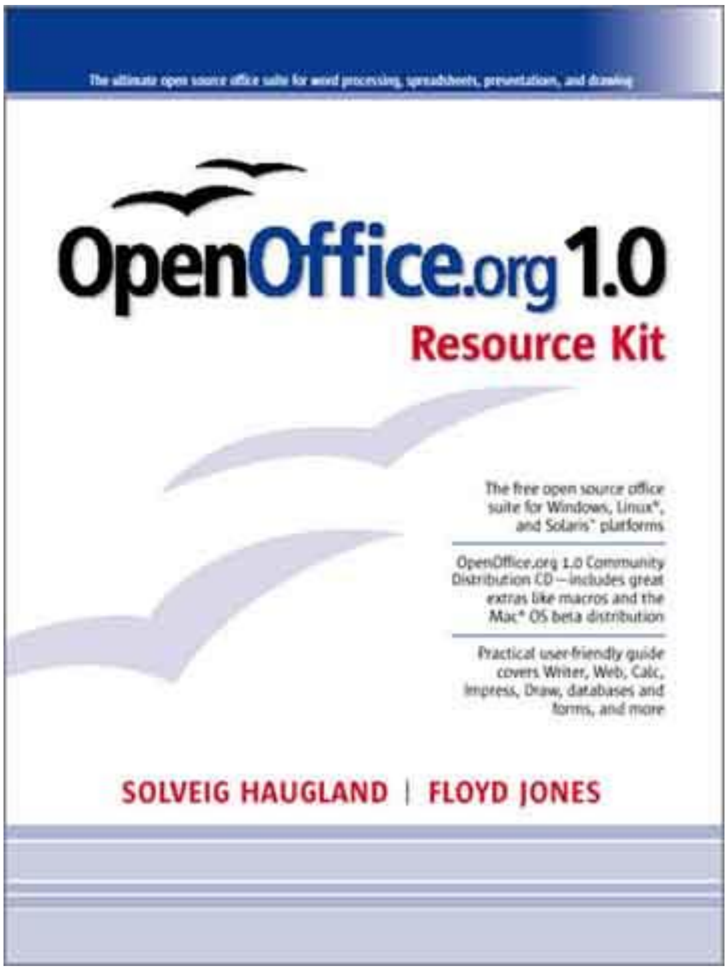
Archive Index Issue Table of Contents

Advanced search

OpenOffice.org 1.0 Resource Kit by Solveig Haugland and Floyd Jones

Kenneth Wehr

Issue #115, November 2003


The ultimate open source office suite for word processing, spreadsheets, presentation, and drawing
OpenOffice.org 1.0
Resource Kit
The free open source office suite for Windows, Linux®, and Solaris™ platforms
OpenOffice.org 1.0 Community Distribution CD—includes great extras like macros and the Mac® OS beta distribution
Practical user-friendly guide covers Writer, Web, Calc, Impress, Draw, databases and forms, and more
SOLVEIG HAUGLAND | FLOYD JONES

Prentice Hall PTR, 2003

ISBN: 0-13-140745-7

$39.99 US

Based on Sun's StarOffice and maintained by a worldwide community of developers, the OpenOffice.org Project provides a full-featured office application suite with a language-independent API and XML-based file formats. The *OpenOffice.org 1.0 Resource Kit* consists of a book and a CD-ROM. The authors, Solveig Haughland and Floyd Jones, are veterans of the technical training field, and it shows in the quality of the text. The CD contains the OpenOffice.org 1.0 release, although two minor upgrades have become available since it was pressed. It also includes templates, macros and examples. The authors provide additional resources at www.getopenoffice.org.

The first five chapters of the book are devoted to basics, such as installation, migrating existing data and printer issues. OpenOffice.org is superb at converting Microsoft Word, Excel and PowerPoint files into its own open formats. The book shows how to use the handy AutoPilot, which can perform batch conversions of your existing data.

The next six chapters cover word processing. The organization of this section is quite intuitive; you'll easily learn how to create a simple letter. When you're ready to write your memoirs, you won't need to buy another book—it's all there: complex formatting options, page layout, linking cross-references and indexing. And don't forget office goodies such as mail merges, label printing and business cards.

Chapters 13–17 cover Web development. Serious Web designers may find this section useless, but the casual user will be able to create a home page without learning HTML.

The next several chapters deal with the Calc spreadsheet, Impress for creating presentations and the underrated Draw. Basic topics are organized neatly along with the more advanced ones, and neither seems to get in the way of the other. The final three chapters of the book explain how to incorporate data from a database.

An appendix covers macros, and makes the book into an all-in-one tutorial and reference, with high marks in all the important areas. It's comprehensive, well organized and has a great signal-to-noise ratio.

Superior open-source software alone isn't always enough to supplant the old way of doing things. Document it, however, and they will come. The *OpenOffice.org 1.0 Resource Kit* goes a long way toward fulfilling that prophecy.

Archive Index Issue Table of Contents

Letters

Readers sound off.

More Small-Business Articles, Please

LJ has published several articles on the use of Linux in Hollywood and its adoption by large corporations. I would like to see how many small businesses, like mine, run on open-source software. I've been using Linux for the past five years and switched one of my associates to it at the time. How many more like me are out here in the real world? I think there are quite a few, and it would be nice to see the numbers publicized.

—

Rich

Show Me the Money

Even though I agree with the Open Source movement for operating systems, as it creates a standardized solid foundation for further software development, what is the purpose of extending it to other genres of software? One thing I have noticed people fail to mention about the Open Source movement is how do the programmers get paid? I am a university student who eventually wants to design software for a living. What's the point of learning extremely complex technology if somebody is willing to make it for free in his or her spare time? It's like asking an electrical engineer to develop the electrical system for a public building for free because it benefits the public. How are software engineers going to survive out there if they're killing themselves with open source? Our job is not easy; it requires a high-level understanding of mathematics and logic and takes years to learn. Why should we do it for free?

—

M. Shah

Doc Searls says the software industry is growing up and becoming more like the construction business. The engineer gets paid for his work, but the building owner doesn't have to pay for a $99 client access license for everyone who walks in the building. In mature industries, customers expect to be able to hire Vendor B to fix something if Vendor A doesn't work out. If you want to make a living in software in the long run, you'll have to fulfill those expectations. You might want to read *Secrets of the Wholly Grill: A Novel about Cravings, Barbecue, and Software* by Lawrence G. Townsend. To get an idea of what it's about, imagine applying some one-sided restrictions from proprietary software licenses to products you might buy at the supermarket. —Ed.

Speed Up Red Hat 9

Kudos on another excellent and informative issue [*LJ*, August 2003]. I found the “Eleven SSH Tricks” article particularly useful. I am writing to comment on Marco Fioretti's review of Red Hat 9, beginning on page 90. It noted excessive startup time for the desktop and OpenOffice. This is a result of Red Hat's conservatism with regards to default system settings. There is a configuration file, /etc/sysconfig/harddrives, that can fix this. It contains the settings to enable DMA mode, multisector I/O, look-ahead functionality and other goodies for modern, fast hard drives. Enable those settings, and you will find that Red Hat 9 is quite speedy indeed.

—
J. Mark Brooks

Ada Isn't Awful

In the August 2003 Letters to the Editor section, Donald Daniel states that “Pascal inspired Ada, which was awful.” Ada was inspired by a need for a good programming language, and it fulfills that need very well. I have yet to meet an Ada basher who really knows the language.

The letter titled “Fighting C++ Rumors” actually confirms the fact that compiled C++ libraries are incompatible with one another. To say that this problem will soon be a thing of the past doesn't cut it; I've been waiting 20 years! For other compatibility issues, get a copy of *C++ Primer Plus* by Stephen Prata and look for the highlighted compatibility notes. Or, stop waiting, move to Ada and don't bash it until you know it. If developers want to make an educated decision as to what language should be used, read my article in *CrossTalk* magazine at www.stsc.hill.af.mil/crosstalk/2003/02/index.html.

—
Dennis Ludwig

Thanks for the Backup Hint

Just got the August 2003 issue, the tip on piping binary data to a remote shell was exactly what I needed to back up my somewhat over-full iBook. Great.

—
Ken Moffat

Mr. Smith, LDAP Won't Let Us Hire You

Mick Bauer's LDAP articles have been fairly well written, but I have a couple of issues with Part III [*LJ*, September 2003]. There are, of course, the broken DNs on page 32, which are easy enough to spot if one reads the whole article; the DNs later on are fine. However, they lead into one of the most problematic situations I've seen in the directory business. I'll call this situation John Smith.

We have 24 John Smiths working for our company. One can add a middle initial, but this is just kludging it. We have five John R Smiths. We could use a middle name; still a kludge. We now have John Ronald Smith, two John R Smiths that refuse to give their middle names and two John Richard Smiths. They're not related, and in fact, are both John Richard Smith III. (This is an actual situation at a company at which I once worked, although the names have been changed.)

Using a unique identifier is considered by many to be the best practice. There's even an attribute for it in the LDAP standard, uid, which is mentioned in another context in Mick's article. This would make one's DNs look like uid=wongfh,ou=engineering,dc=wiremonkeys,dc=org. The only restrictions on uids are that they must be unique to that branch of the tree, and they really should be unique to the LDAP server. It's also a good idea to have a standard on how one makes a uid. The only bad standard I've seen is one that isn't consistently followed, although many groups prefer one never reassigns a uid that once belonged to a deleted entry.

This is a controversial topic; many people, including some of the original LDAP server authors, are rather devoted to cn-based DNs. However, it can cause problems that are easily avoided, for very little cost. I've consulted for a group that uses cn-based DNs on their e-mail server, which accepts DNs as one user name format. Several times a year, they ask me how they can prevent e-mail intended for an ex-employee from going to a new employee with the same name. They always get the same answer. IMHO, it's really about how long you want to endure the pain.

—
Ed Grimm

Mick Bauer replies: First, I apologize for the typos you pointed out. They were caused by my copying and pasting working code from my test server, and then inconsistently replacing the test server's real domain name with my example domain name. Second, thanks very much for sharing your insights on the “John Smith” problem. You've convinced me that a uid-based naming convention is a much better-scaling approach than a cn-based system. I think the latter works fine for small organizations, but I readily admit that it can't scale as well as the uid approach. This is an excellent illustration of how complicated and subtle LDAP design and administration can be. There are no one-size-fits-all approaches to LDAP.

Refund Struggle Continues

I was wondering what ever became of “The Toshiba Standoff” (/article/6318). It was such a compelling story that I just gotta know if anything ever panned out for Adam Kosmin.

—
William Totman

Adam struck out, but Steve Oualline got a refund for the unused proprietary OS that came with his laptop. It takes persistence, but it's possible (www.linuxjournal.com/article/7040). —Ed.

Easy Personal Video Recorder

I just read Marcel's article in the September 2003 *Linux Journal*, taking great notice of the MythTV part. I'm going to have to toot my own horn for just a minute here (well, and some others). With the help of the community, and Axel Thimm, the maintainer of ATrpms and the MythTV RPMs for Red Hat Linux, I've put together a streamlined guide for the creation of your own MythTV system using Red Hat Linux 9. I have the process of creating a full-blown, fully functional MythTV system from bare metal down to two to three hours, but then, I've done it a few times. The process is greatly streamlined by installing nearly every component from RPMs, using apt (pvrhw.goldfish.org/tiki-page.php?pageName=rh9pvr250).

—
Jarod C. Wilson

Linksys Switches Chipsets

In an article in the September 2003 issue of *Linux Journal*, actually a Sidebar to the article “Linux Makes Wi-Fi Happen in New York City” entitled “Pebble Linux: Debian for Wi-Fi Access Points” Kurt Starsinic mentions the popular Linksys

WPC11 wireless card. As of WPC11 v.4, this card is no longer Linux-compatible. The Prism chipset has been replaced with another, and Linksys technical support will not say if or when Linux drivers will be available. They have even evidently been told not to identify the new chipset: RealTek.

—
Jeff Simmons

Compatible Wireless Cards?

The September 2003 issue, centering on wireless networks and Linux, was quite enjoyable. However, there was something missing that was sorely needed. There was no mention of what wireless network cards could be used with Linux. The Hardware HOWTO lacks any such supported list as well. After dredging through manufacturer Web sites, such as Adaptec and Linksys, my outlook on converting my current LAN to wireless is not a good one. Can you list what wireless network cards were used in any of the various articles?

—
Wally Barnes

There's a good card list at seattlewireless.net/?HardwareComparison. You also can check your distribution's hardware compatibility list on the Web. As the previous letter shows, it's hard to keep up with hardware in print. —Ed.

Starting a Community Net

I've been a *Linux Journal* reader now for around seven years, and my brief stint as a part-time copy editor was something I was proud of and look back on fondly. *LJ* has been my favorite magazine since I picked up my first issue at a local computer show at the University of Washington in 1996 or 1997. Recently, I bought a Mac, and with it came an offer for a free subscription to *MacWorld*, which I accepted. Last night, after reading both magazines, I realized exactly why *Linux Journal* has been my favorite magazine for so long. Where other magazines make me want to buy things, *Linux Journal* encourages me to *try* things. Your pages are consistently filled with informative articles about projects, not products. The September 2003 issue's focus on community networks has strengthened my desire to build a free node, and hopefully, a community around it. Thank you for continuing to produce an excellent magazine.

—
Nathan E. Sandver

Send us the location and ESSID for that community net when you have it up. —Ed.

Yay Phil Hughes, Boo SCO

I've subscribed over the past year and, frankly, was not going to re-up. The reason is simply that I'm just an average guy—a home user and most (but not all) of your articles are over my head. But I've been following the SCO situation, and today I read the Open Letter from SSC to SCO [www.linuxjournal.com/article/7087]. In effect, you guys are fighting for my right to use Linux. Even though I'm not a professional geek, the least I can do is support people that are supporting me (and many, many others) in our ability to use the best OS out there. Expect my renewal in the next couple of days. I wish you all at *LJ* well.

—
Michael Presley

More on Automation, Please

I read Tad Truex's article (*LJ*, September 2003) "Put a Sump Pump on the Web with Embedded Linux" and found the article interesting. I also found myself chuckling over what the broader audience would think of such an idea. That said, in a bigger context, there is the field of industrial automation and electrical engineering that uses SCADA systems for industrial process control. Would it be possible to feature the subject, using Linux and other OSS solutions in a future article, using some of your experts' experience?

—
Jason Houlihan

SE Linux outside US?

I just picked up the August 2003 issue of *LJ*, mostly because of the SE Linux article by Russell Coker. After reading it (with much interest) I decided to surf to the NSA site to look at some docs and possibly acquire the downloads concerning SE Linux. Although hardly anything to do with the American government surprises me anymore, I was slightly surprised that access to the site is blocked, probably for non-US based IPs, as I am accessing the site from The Netherlands; all access to the site was blocked even the top-level nsa.gov.

—
Peter van der Kleut

Try Russell's site at www.coker.com.au/selinux. —Ed.

gwc vs. gramofile

I read with interest Tom Younker's article on converting vinyl LPs to digital formats [*LJ*, September 2003, page 80]. Recently I have been trying to convert my wife's extensive collection of old Russian LPs to CDs. Initially, I tried using gramofile as described by Mr Younker, but a few months ago I discovered a more recent project called the Gnome Wave Cleaner (gwc.sourceforge.net). I now use this exclusively and find that it is an excellent piece of software.

I believe that the major benefits of gwc over gramofile are: 1) As Mr Younker says, “swig has progressed, but gramofile hasn't.” 2) gwc has a nice GUI interface (gwc.sourceforge.net/main.jpg), whilst gramofile is ncurses based. 3) The click and noise removal algorithms in gwc work very well and are easy to control. As an added bonus, if there is a recalcitrant click that the auto-algorithm can't get a grip on, you can zoom in using the GUI, select the offending millisecond of sound and silence it by hand. It disappears without a trace. I certainly haven't carried out extensive testing of click removal speed and effectiveness, but gwc is fast enough for me, and nearly all clicks are removed. 4) gwc has a sonogram view that is very handy for tracking down the few remaining clicks in a track.

Finally, don't just take my word for it. Read James Tappin's excellent page on ripping 78rpm records (www.tappin.me.uk/Linux/audio.html). He used to recommend the use of gramofile, but he now uses gwc and describes it as “best-of-breed”.

—
Richard Simpson

Drive This by M. Shah's House, Will You?

Don, attached is a photo of my car with *the* California LINUX license plate installed. When I first registered for the plate around 1993, it was installed on a rather beat-up 1986 Toyota pickup. As you can see, Linux has helped us grow our business, and the plate has been transferred to a slightly nicer vehicle.

By the way, need a DSL connection from a clueful ISP?

—
Dane Jasper

BECOME A PATRON (https://www.patreon.com/bePatron?u=10705728)

(/)

(/content/linux-journal-ceases-publication-awkward-goodbye)

Linux Journal (/taxonomy/term/29)

Linux Journal Ceases Publication: An

Awkward Goodbye (/content/linux-journal-ceases-publication-awkward-goodbye)

IMPORTANT NOTICE FROM LINUX JOURNAL, LLC:

On August 7, 2019, Linux Journal shut its doors for good. All staff were laid off and the company is left with no operating funds to continue in any capacity. The website will continue to stay up for the next few weeks, hopefully longer for archival purposes if we can make it happen.

–Linux Journal, LLC

Final Letter from the Editor: The Awkward Goodbye

by Kyle Rankin

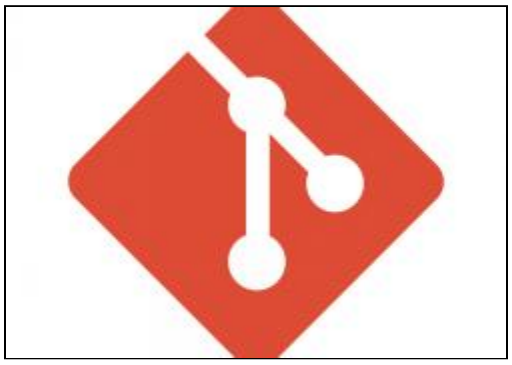
Kyle Rankin (/users/kyle-rankin) - August 7, 2019

(/content/oops-debugging-kernel-panics-0)

Oops! Debugging Kernel Panics (/content/oops-debugging-kernel-panics-0)

(/content/loadsharers-funding-load-bearing-internet-person)

Loadsharers: Funding the Load-Bearing Internet Person (/content/loadsharers-funding-load-bearing-internet-person)


(/content/documenting-proper-git-usage)

Documenting Proper Git Usage (/content/documenting-proper-git-usage)

Breaking News

Linux Mint 19.2 "Tina" Cinnamon Now Available, IBM Has Transformed Its Software to Be Cloud-Native and Run on Any Cloud with Red Hat OpenShift, Icinga Web 2.7.0 Released, Google Rolling Out Android Auto Design Updates and Kernel 5.1 Reaches End of Life (/content/linux-mint-192-tina-cinnamon-now-available-ibm-has-transformed-its-software-be-cloud-native)

Canonical Announces the Availability of Xibo as a Snap, Chrome 76 Released, Viruses Discovered in LibreOffice, Pop!_OS 18.10 Reaches End of Life, and Dutch Ministry of Justice and Security Warns of Microsoft Office Online Privacy Risks (/content/canonical-announces-availability-xibo-snap-chrome-76-released-viruses-discovered)

Collabora Announces xrdesktop, Blender 2.8 Released, Arduino Selects Auth0 as Its Identity Management Platform of Choice, Microway Showcasing Its Data Science WhisperStation at PEARC19 and KDE Plasma Maintenance Update (/content/collabora-announces-xrdesktop-blender-28-released-arduino-selects-auth0-its-identity)

Linux Ending Support for the Floppy Drive, Unity 2019.2 Launches Today, Purism Unveils Final Librem 5 Smartphone Specs, First Kernel Security Update for Debian 10 "Buster" Is Out, and Twitter Is Switching from Mesos to Kubernetes (/content/linux-ending-support-floppy-drive-unity-20192-launches-today-purism-unveils-final-librem-5)

see more news >> (/news)

(/content/bash-trap-command)

The Bash Trap Command (/content/bash-trap-command)

Mitch Frazier (/users/mitch-frazier) - August 7, 2019

If you've written any amount of bash code, you've likely come across the trap command. Trap allows you to catch signals and execute code when they occur. Signals are asynchronous notifications that are sent to your script when certain events occur. Most of these notifications are for events that you hope never happen, such as an invalid memory access or a bad system call. However, there are one or two events that you might reasonably want to deal with. There are also "user" events available that are never generated by the system that you can generate to signal your script. Bash also provides a psuedo-signal called "EXIT", which is executed when your script exits; this can be used to make sure that your script executes some cleanup on exit.

(/content/episode-24-chat-about-redis-labs-podcast-transcript)

Reality 2.0 Episode 24: A Chat About Redis Labs (Podcast Transcript) (/content/episode-24-chat-about-redis-labs-podcast-transcript)

Katherine Druckman (/users/katherine-druckman) - August 2, 2019

Doc Searls and Katherine Druckman talk to Yiftach Shoolman of Redis Labs about Redis, Open Source licenses, company culture and more. Listen to the podcast here.

(/content/simplifying-function-tracing-modern-gcc)

Simplifying Function Tracing for the Modern GCC (/content/simplifying-function-tracing-modern-gcc)

Zack Brown (/users/zack-brown) - July 26, 2019

Steven Rostedt wanted to do a little housekeeping, specifically with the function tracing code used in debugging the kernel. Up until then, the kernel could enable function tracing using either GCC's -pg flag or a combination of -pg and -mfentry. In each case, GCC would create a special routine that would execute at the start of each function, so the kernel could track calls to all functions.

(/content/what-does-it-take-make-kernel-0)

What Does It Take to Make a Kernel? (/content/what-does-it-take-make-kernel-0)

Petros Koutoupis (/users/petros-koutoupis) - July 23, 2019

The kernel this. The kernel that. People often refer to one operating system's kernel or another without truly knowing what it does or how it works or what it takes to make one. What does it take to write a custom (and non-Linux) kernel?

(/content/oracle-linux-btrfs-raspberry-pi)

Oracle Linux on Btrfs for the Raspberry Pi (/content/oracle-linux-btrfs-raspberry-pi)

Charles Fisher (/users/charles-fisher) - July 22, 2019

Enterprise comes to the micro server.

(/content/data-flash-part-iv-future-memory-technologies)

Data in a Flash, Part IV: the Future of Memory Technologies (/content/data-flash-part-iv-future-memory-technologies)

Petros Koutoupis (/users/petros-koutoupis) - July 19, 2019

I have spent the first three parts of this series describing the evolution and current state of Flash storage. I also described how to configure an NVMe over Fabric (NVMeoF) storage network to export NVMe volumes across RDMA over Converged Ethernet (RoCE) and again over native TCP.

see more recent articles >> (/recent)

Get Our Newsletter

Email

I give my consent to be emailed

SUBSCRIBE

Reality 2.0

A *Linux Journal* Podcast

(/node/1340785)

Subscribe to Feed (https://www.linuxjournal.com/podcast.xml)

More Episodes (https://www.linuxjournal.com/podcast)

Corporate Patron

Making IT Heroes Pulseway (https://www.pulseway.com?rfid=linuxjournal)

Community Events

Open Source Summit North America (https://events.linuxfoundation.org/events/open-source-summit-north-america-2019/)

August 21, 2019 - August 23, 2019
San Diego, CA
USA

Artificial Intelligence Conference (https://conferences.oreilly.com/artificial-intelligence/ai-ca)

September 9, 2019 - September 12, 2019
San Jose, CA
USA

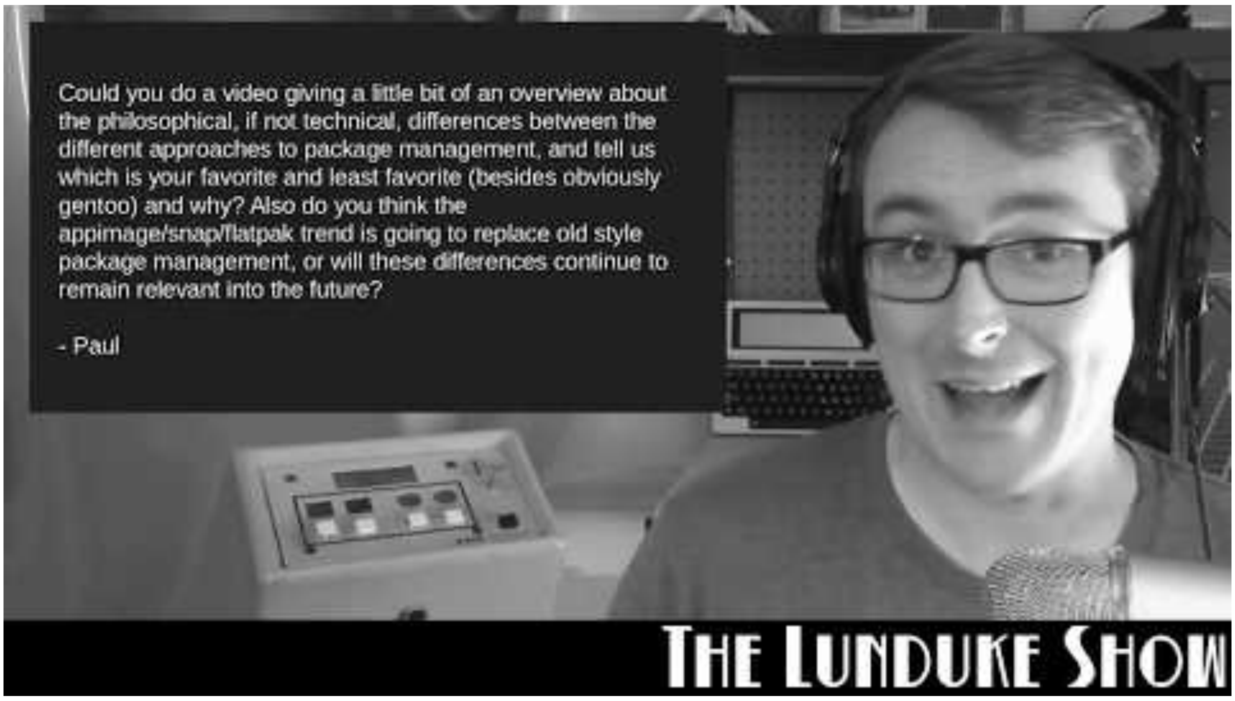
HPC on Wall Street (https://www.hpcandaionwallstreet.com/)

September 11, 2019 - September 12, 2019
New York, NY
USA

Strata Data Conference (https://conferences.oreilly.com/strata/strata-ny)

September 23, 2019 - September 26, 2019
New York, NY
USA

see more events (/events)

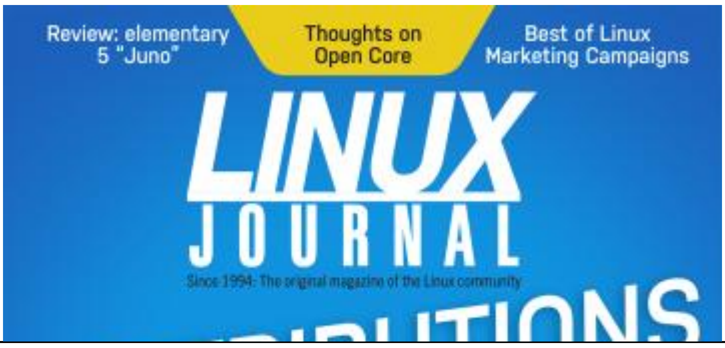

Could you do a video giving a little bit of an overview about the philosophical, if not technical, differences between the different approaches to package management, and tell us which is your favorite and least favorite (besides obviously gentoo) and why? Also do you think the appimage/snap/flatpak trend is going to replace old style package management, or will these differences continue to remain relevant into the future?

- Paul

Comparing Linux Package Formats - Deb, Flatpak, AppImage, etc.

(/video/comparing-linux-package-formats-deb-flatpak-appimage-etc)

Video by Bryan Lunduke (/users/bryan-lunduke) on July 18, 2019


Review: elementary 5 "Juno"
Thoughts on Open Core
Best of Linux Marketing Campaigns
LINUX JOURNAL
Since 1994: The original magazine of the Linux community

Advanced search

From the Editor

High-Performance Computing

Don Marti

Issue #115, November 2003

Read all about it—SARS, clusters and the blackout of August 2003.

Visit the Computer History Museum in Mountain View, California, and you'll get close enough to smell the machines that were the fastest computers of their time. Control Data Corporation's CDC 6600 and 7600, designed by Seymour Cray, are two historic systems at the museum. If you go at the right time you might even run into *Linux Journal*'s Michael Baxter, who can explain almost everything about Cray's designs except maybe the fake wood grain on the 7600.

CDC products served their time in the US national laboratories and other sites that buy the fastest machines, regardless of little details like backward compatibility. When CDC tried to enter the business computing market, it sank without a splash. Cray himself went on to found Cray Research, but as long as there has been a computer hardware business, high-performance computing (HPC) success has spelled failure in the business computing market.

As we go to press, the latest hot system on order for a national lab is the Lightning cluster from Linux NetworX, which will be put to work on tasks vaguely described as having to do with “safety and reliability of the nation's nuclear weapons stockpile” at Los Alamos.

Will Linux clusters stay in the HPC niche? Big vendors are putting their money on “no”. Oracle is dropping UNIX boxes for cheap racks of generic machines. Penguin Computing acquired Beowulf-originator Donald Becker's cluster company, Scyld. Dell and IBM will sell you turnkey clusters with service contracts—maybe not with one click from the Web site, but close.

Linux supercomputers already wallow in the bargain basement of price-performance, using technologies on the commodity market or intended for the commodity market, such as x86 and AMD64 processors, Gigabit Ethernet and Infiniband.

Martin Krzywinski and Yaron Butterfield give us an inspiring story of how a lab with Linux infrastructure got the first sequence of the SARS virus, under time pressure. Catch Linux bioinformatics fever on page 44.

Back in the day, cluster managers had to write their own network drivers and walk to the data center in the snow, but Steve Jones got help from his cluster vendor in bringing up a TOP500-class cluster at Stanford University (see page 72).

The more you learn about clusters, the more you might be tempted to order a whole bunch of boxes and integrate them yourself. That could be either the most money you ever saved or the biggest mistake you ever made. Make your homebrew cluster a successful one by putting some sample nodes through John Goebel's cluster hardware torture tests on page 62.

Finally, Reuven M. Lerner was a little late with his monthly column, thanks to a huge blackout on the east coast of the US and Canada. Find out how to prepare for one in his “Server Migration and Disasters” on page 14.

Don Marti is editor in chief of *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

On the Web

The Language Battle

Heather Mead

Issue #115, November 2003

Whether you're looking to do something new with your old favorite language or to try a new one, visit the *LJ* Web site for ideas.

Some arguments never seem to lose their popularity. Perhaps the new rule should be never talk about religion, sex, politics or programming languages—but how boring would that be? The publication of “Interview with Bjarne Stroustrup”, the creator of C++ (www.linuxjournal.com/article/7099), on the *Linux Journal* Web site, set off another battle in the ongoing programming language flame war. In the interview, Stroustrup discusses some of the history behind C++ as well as its evolution. The comment section, however, turned into an argument on whether C++ is a bad language or whether programmers merely use it incorrectly.

One of Stroustrup's rather sensible observations is people should use the best language for the task at hand. So with that in mind, this month we offer a recap of some language-specific articles available on the *LJ* Web site. With the increasing popularity of LDAP servers, it's good to have several management options available. In “An Introduction to perl-ldap” (www.linuxjournal.com/article/7086), Paul Dwerryhouse discusses how to use the Net::LDAP Perl module to “enable easy access to the data contained in LDAP directories from Perl scripts”. He then walks users through installing the module and demonstrates how to add, search for and modify entries.

If LDAP is your thing but Perl is not, check out “LDAP Programming in Python” (www.linuxjournal.com/article/6988) where Ryan Kulla explains how to use the python-ldap package with its object-oriented client API to work with LDAP directories. Kulla's goal is to “get you ready to write your own programs to automate the querying process of LDAP servers”. His example program demonstrates some of the basics that you can extend and explore on your own.

Continuing with Python, in “Host-Hopping Scripts in Python” (www.linuxjournal.com/article/6730), Mark Nielsen explains “how and why I used SSH, Python and Expect to transfer Weblogs to a central computer for processing”. A good example of using the right tools for the task, Mark's project also took advantage of tools already available on the network computers. The scripts included in the article show you exactly how he combined all the elements to gather together the systemwide Weblogs.

Not quite as volatile as the interview with Stroustrup, Aleksey Dolya's “Interview with Brian Kernighan” (www.linuxjournal.com/article/7035) explores some of the history behind the AWK programming language. The two also discuss C, how UNIX got its name, the early days of Bell Labs and teaching the next generation of IT workers.

If your personal favorite languages aren't mentioned here, do a search for them on the *LJ* site. We also have articles about working with Ruby, Objective-C, Java and others. And, be sure to check the *Linux Journal* Web site often; new articles are added daily.

Heather Mead is senior editor of *Linux Journal*.

Archive Index Issue Table of Contents

Advanced search

Best of Technical Support

Our experts answer your technical questions.

Unclean Shutdown Messes Up Large Directory

We have a directory containing a large number of small- and medium-sized files (approximately 100,000). When we installed Red Hat 8.0, we ran into a problem when the system has an unclean shutdown. We get a message about a too-large inode, and the system goes to single-user mode. The only way to repair this is to do an `e2fsck`, answering `y`. The directory is completely gone, and all the files are moved to the lost&found directory. This never happened under releases up through Red Hat 7.3. We were able to reproduce this on two different types of computers, both using Red Hat 8.0. We have the same problem using either ext2 or ext3 filesystems.

—
Joe Waytula

joseph.waytula@ipaper.com

What probably happened is that your system was powered off or crashed while you were writing to the directory. This causes the directory to be in an inconsistent state, and the directory is lost. Although ext3 is a journaling filesystem, it guarantees that your system will recover quickly, but not that no data can be lost. In this case, the data lost is your directory. ext2 and 3, without the hashed directory extension, don't like directories with that many files. It's slow, and as a result you have a big time window during which your directory can be potentially corrupted during a crash. I would recommend that you switch your filesystem to ReiserFS, XFS or JFS, as they all have better support for big directories.

—
Marc Merlin

marc_bts@google.com

Dell Laptop Hangs at Boot

I purchased a Dell D800 laptop with 256MB of memory and 30GB of hard disk. It also has an interchangeable CD-ROM and floppy. It came with MS WinXP installed, but, like my desktops, I wanted to have Red Hat Linux 8.0 on the rest of the disk. I allocated 6GB for Windows and proceeded to divide up the remainder of the disk for what Linux needed. I used GRUB to make the machine dual-boot between Windows and Linux. It looked like I could do everything just as I did with the desktops. The installation from CD-ROM went flawlessly. I did not make a boot floppy, because I did not want to switch out the CD-ROM at that point. However, after I congratulated myself on a smooth installation, when I tried to boot Linux, the machine stalled during the boot, but shortly after the boot started. The keyboard went dead and nothing would respond. The display did not go blank, but I had to turn the machine off and then on to reboot Windows. I tried re-installing twice afterward, but the same thing happened. It's as if an interrupt is stopping the machine. I've read about APM interrupts and laptops, but I am not sure what to do at this point. I really want Linux on my laptop.

—
Rob Borochoff

borochoff456@comcast.net

Whenever you have Linux questions about a specific laptop model, check out the excellent Linux on Laptops Web site: www.linux-laptop.net. It has a link to someone who has successfully installed Linux on this exact model, which will answer your questions on how to get Linux running properly on your hardware.

—
Greg Kroah-Hartman

greg@kroah.com

Try fetching or buying a copy of Knoppix (www.knoppix.com), which runs from the CD without installing, and see if it can boot, autodetect and autoconfigure your laptop's hardware. If it can, you have some valuable clues to use in configuring Red Hat or any other distribution. You might even find that you like running Knoppix straight from the CD or that you want to run its little installation program. I suggest it here because it currently has the best hardware autodetection and autoconfiguration.

—
Jim Dennis

jimd@starshine.org

termcap or terminfo?

I need to use an xterm in order to connect to a SCO machine. I can't find the definitions of the emulation. Does xterm use terminfo or termcap?

—
Aldo Gentile

agentile@lacapital.com.ar

All modern Linux distributions use ncurses, which uses terminfo rather than termcap. You might try running rxvt and using vt100 or vt220 as the terminal type.

—
Jim Dennis

jimd@starshine.org

There's More than One Way to Name Ethernet Cards

I have two Ethernet cards in the same machine. Is there a way to specify one card to be eth0 and the other to be eth1 always?

—
Ning Qian

nq6@columbia.edu

Yes there is. Look into the nameif program. It will name network devices depending on their MAC address. So, no matter which way the kernel probes your Ethernet cards, you always can name them the same way. Combine nameif with the hot-plug scripts (available at linux-hotplug.sf.net) and your network devices can be named properly whenever the device is found by the kernel.

—

Greg Kroah-Hartman

greg@kroah.com

There is an ether= kernel command-line parameter that can control the assignment of Ethernet device names and other resources for statically linked drivers, and there are command-line options and aliases in /etc/modules.conf to control them for loadable module drivers. The bootparam(7) man page (type `man 7 bootparam` from a terminal) provides details on the former; and the modules.conf(5) man page explains more than you will want to know about the latter.

—
Jim Dennis

jimd@starshine.org

If they are different cards, make sure you are using modules and select the order in /etc/modules.conf, like this:

```
alias eth0 e100
alias eth1 3c59x
```

—
Marc Merlin

marc_bts@google.com

New Products

Pioneer DVD-Video Recorder, LONE-Tar 4.0 and more.

Pioneer DVD-Video Recorder

A modular, Linux-based DVD-video recorder targeted at the industrial video market, the PRV-LX1 professional DVD-Video recorder is now available from Pinoeer Electronics. The recorder has real-time video recording capabilities, which streamlines the DVD video capture, compression, authoring and recording processes while operating like a VTR. The standalone device transfers video and audio content from multiple sources to DVD and offers the ability to customize DVD menus and chapter points. The base system comes with a DVD-R/RW drive and a 120GB internal HDD, with the option to add a second DVD-R/RW. Video inputs and outputs are component video, composite, S-video and DV; audio options are 2-channel balanced, 2-channel unbalanced and coaxial digital. The PRV-LX1 also has interfaces and connections for RS-422A control, Ethernet, VGA output, USB2, external sync and headphones.

Pioneer Electronics (US) Inc., 2265 East 220th Street, Long Beach, California 90810, www.pioneerelectronics.com.

LONE-TAR 4.0

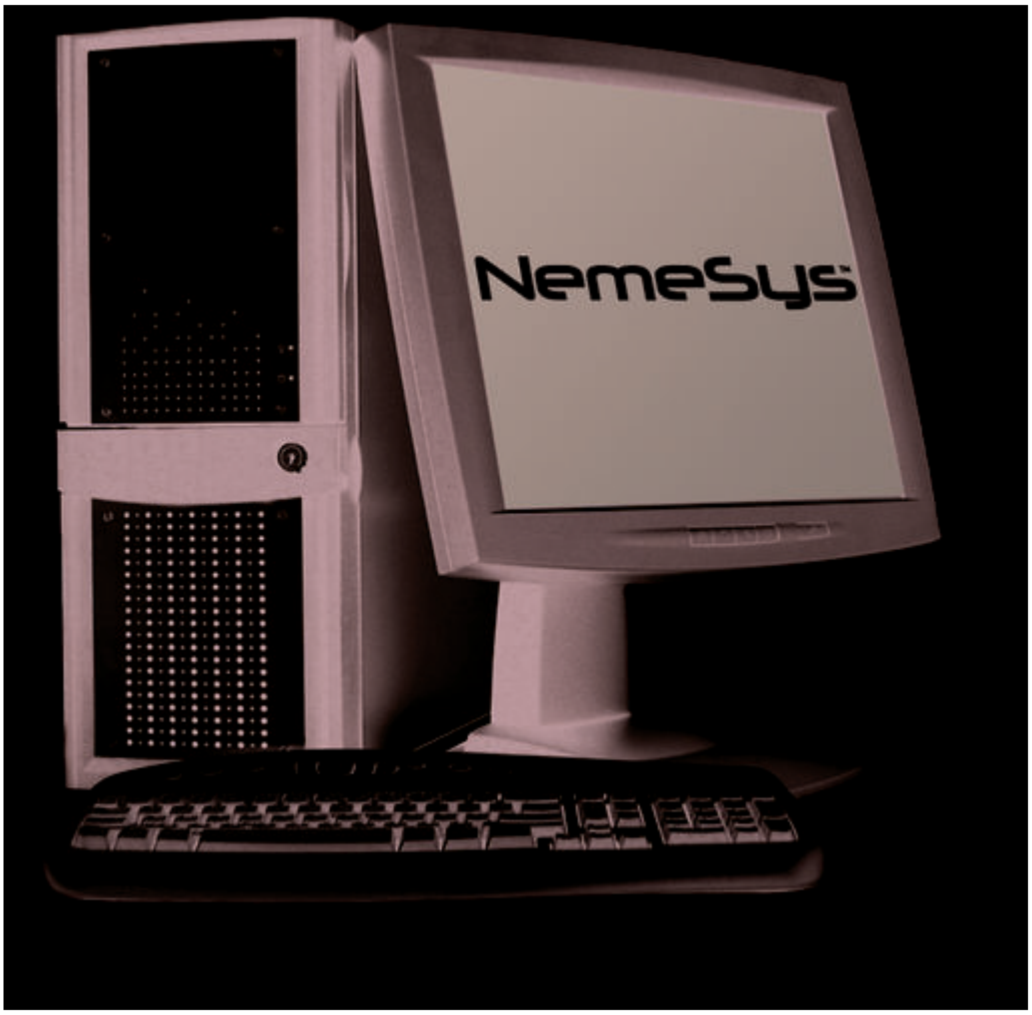
Lone Star Software Corporation announced the availability of LONE-TAR version 4.0 backup and recovery software. New features for version 4.0 include 256-bit encryption; backup creation on optical media, including DVD-RAM, DVD+R/RW and CD-R/RW formats; HP tape drive compatability; device autodetection; a license manager; redesigned GUI and character menu interface; on-line updates; and a self-extracting installation wizard. In addition, LONE-TAR 4.0 introduces a bootable LONE-TAR backup with RESCUE-RANGER disaster recovery, which combines backup and recovery on one piece of media. The device manager allows users to add, change and remove devices at any time and customize each with its own settings for compression and encryption.

Lone Star Software Corporation, 509 East Ridgeville Boulevard, Mount Airy, Maryland 21771, 800-525-8649, www.cactus.com.

NemeSys Graphic Workstations

RackSaver has introduced a line of high-end graphic workstations designed for design, digital content creation, rendering and other graphics-intensive operations. The NemeSys 720 series workstations come with dual Xeon or Opteron processors and memory options of up to 16GB of RAM. Up to eight hard drives can be stored in a workstation, allowing for up to 2TB of storage capacity. NemeSys workstations offer support for high-end graphics cards, such as the NVIDIA Quadro FX card family, including the new NVIDIA Quadro FX 3000 card. A variety of motherboards can be configured for the NemeSys, including Tyan, Intel, Arima, MSI and SuperMicro. DVD-R, DVD-ROM, CD-R/RW and CD-ROM options all are available.

RackSaver, Inc., 9449 Carroll Park Drive, San Diego, California 92121, 858-874-3800, www.racksaver.com.



iNAV 9200

The iNAV 9200 Multiprotocol Gateway Solution functions as a standalone gateway that bridges multiple communications protocols. This 1U system is designed to unite networks lacking a common communications language by translating different network protocols, such as legacy circuit-switched infrastructures and newer packet-based technologies. The iNAV 9200 is designed specifically for use in broadband access applications requiring a gateway function, including carrier DSL, cable and fixed wireless applications. It also can serve as a standalone multiservice switch or media gateway or perform packet routing/classification, ATM switching and ATM segmentation and re-assmebling. OEMs have options for circuit-based interfaces, including T1/E1/J1, T3/E3, OC-3/STM-1 and OC-12/STM-4.

Interphase Corporation, Parkway Centre, Phase 1, 2901 North Dallas Parkway, Suite 200, Plano, Texas 75093, 800-327-8638, www.iphase.com.

Lindows.com BusinessStation

Lindows.com's new BusinessStation offering is a low-maintenance computer designed for work terminals, public access and in-store information kiosks or any other function that needs to offer Web-based communications to customers, employees or back-end systems. Built on Lindows.com's WebStation design, BusinessStation comes with a network-based management tool that enables customization of 1–5,000 machines from any Web browser. Powered by LindowsCD, BusinessStation provides tools for Web browsing, instant messaging, audio/video playback and Web mail, as well as an office suite.

Lindows.com, Inc., 9333 Genesee Avenue, 3rd Floor, San Diego, California 92121, 858-587-6700, www.lindows.com.

Columbitech Wireless Suite

Columbitech announced the availability of a Linux client for its Columbitech Wireless Suite, which allows PDA application developers, car manufacturers and telematic developers to build wireless security applications without requiring a lot of CPU or memory power from the device. Columbitech Wireless Suite consists of three software components: Columbitech Wireless VPN, client software installed on a PDA or laptop; Columbitech Gatekeeper, installed on a server in the DMZ for handling authentication and load balancing; and Columbitech Enterprise Server, installed on a server in the corporate network handling the VPN sessions. Columbitech Enterprise Server includes Columbitech Wireless VPN Server and Columbitech WAP Connector, and it terminates both the encrypted VPN tunnel and WTLS-based WAP communication.

Columbitech, Inc., 11 Penn Plaza, 5th Floor, New York, New York 10001, 212-946-4820, www.columbitech.com.

Snap Server 14000 3TB

Snap Server 14000 3TB is a network-attached storage product that offers 3TB of storage capacity. The 14000 3TB enables enterprise IT workers to consolidate file servers into a centralized system for information sharing across heterogeneous networks with Linux, UNIX, Mac and Windows clients. Guardian OS is used in the 14000 3TB, which offers software for ADS integration, snapshots, network backups and data replication with server-to-server synchronization software. The high-availability architecture includes RAID5, hot-swappable disk drives, redundant components and dual Gigabit Ethernet ports.

Snap Appliance, Inc., 2001 Logic Drive, San Jose, California 95124, 408-879-8700, www.snapappliance.com.

Archive Index Issue Table of Contents

Advanced search