USENIX Association

Proceedings of the 17th Large Installation Systems Administration Conference

San Diego, CA, USA
October 26–31, 2003

The Realities of Deploying Desktop Linux

Bevis R. W. King, Prof. Roger Webb, and Dr. Graeme Wilford – University of Surrey

ABSTRACT

The School of Electronics and Physical Sciences has developed and deployed a Linux-based Desktop solution with thousands of users across many hundreds of machines. We have successfully delivered this solution across the whole range of staff and students within a large organization. It has reduced per-desktop support issues to a minimum and given a stable, reliable, up-to-date and flexible environment to our user community. What's more, it also provides Windows.

Our second generation variant is now deployed, and a third generation is in development. We've based our work on standard distributions and software tools and achieved a number of key objectives:

- Zero intervention network-based installation
- Full Remote Administration including complete unattended re-install
- Dataless clients with no need to back them up
- Extremely configurable: extensive support for different peripherals, varying software installations, and diverse hardware
- Automatic updating of all software: multiple simultaneous technologies providing a flexible solution
- Centralized authentication and authorization
- Exceptional levels of user choice: Linux, Windows and Hybrid environment

Our environment is based upon RedHat Linux 7.3, Ximian GNOME 1.4 and VMware 3.2 providing Windows NT 4.0. Our first generation environment was based upon RedHat Linux 6.2, Ximian GNOME 1.2 and VMware 2.0 providing Windows NT 4.0. This paper discusses those technologies we have have deployed with particular focus on the implications of each choice made. All of the scripts mentioned will be available as a tarball from our web site to aid others.

History and Overview

The School of Electronics and Physical Sciences at the University of Surrey is like many other Electronic Engineering and Computing Science departments at Universities around the world in that we run our own computing services distinct from the University's Central Computing service. We have a long tradition of using primarily Unix machines going back more then twenty years, although we've always supported remote access to filespace, email and applications from a range of other platforms. We have a small but experienced support team consisting of some core staff funded centrally, and some based within research centers. We have a user community of around 2,600 users being served by around 800 computers.

Some years ago, it became apparent that the cost/performance ratio of PC hardware was likely to reduce the attractiveness of our traditional Unix workstation environment, while at the same time political moves within the University were pressing for guaranteed interoperability between all computing systems on campus. The Vice-Chancellor wanted to be able to email all University Staff, and, if he wished, attach a Word or Excel document. One particular aspect of this requirement was that he wanted the IT staff held responsible for it working at all times. A separate initiative at the same time was aimed at removing all pirated software from all machines owned by the University. This effectively meant that the end user's administrative access to the machines would have to be removed. This meant an end to our previous "look after it yourself" attitude to Windows boxes.

This left us with one of two options:

a. Accept that most members of staff would have two machines; an official "Staff" workstation supported by the University's Central Computing Service and the technical workstation (*nix) needed for their actual work.
b. Develop a solution that combined the two roles in a single machine with centralized administration.

We decided on the latter course of action and the Hybrid workstation project was born.

Our initial planning phase included a review of the technologies available. We considered a range of operating system options and settled on RedHat Linux since it was the version all of the system administrators had the most experience of. While some administrators had experience of other distributions (e.g., Debian, Slackware, etc.), RedHat was the common ground. For desktops we considered GNOME, KDE and CDE; we settled on GNOME partially for licensing reasons (The license on the Qt Library, an integral part of KDE, was still perceived as a serious problem then) and partially because of Sun's stated intent of going for GNOME. We hoped for a common desktop environment on both our new Hybrid machines and on our many SPARC/Solaris workstations.

We have had a long history with using a wide range of PC emulation/access products under Unix including Insignia's SoftPC and NTrigue products, SunPC (both with and without co-processor cards) and Windows Terminal server. None of the products we'd previously used seemed that suitable for deployment, so we decided to try a (then) promising new technology called VMware.

The main objectives were:

- To make the operating system installation as simple as possible, and include the Windows environment within it.
- To provide centralized software installation including package management and updates.

We had previous experience of updating a dual-boot workstation environment using Linux to re-install a Windows NT disk image nightly and found it had made an enormous difference to the supportability of Windows to have the machine forcibly reset to a known-good state each night.

Administration and Installation

The key objectives here were to have a simple and automatic installation system, automatic package updating, and to have software package configuration managed centrally. Ideally this should work for operating system updates, for optional package updates and for the Windows applications.

Our zero-intervention network-based installation system is based upon the kickstart system provided by RedHat, which permits a full RedHat distribution to be downloaded over the net and automatically installed. First a server volume is created and the contents of the main RedHat distribution CDs copied into it. The volume is then exported via NFS.

You then install a machine over the network giving the various options you desire; at the end of the install, the RedHat installer (called anaconda) drops a file called "anaconda-ks.cfg" into root's home directory (/root) on the newly installed machine. This file is a template for reproducing that set of install selections on another machine.

This file can easily be modified by hand (some examples are in the supporting tarball) for fine tuning and is then placed in /tftpboot on the server and its filename specified in the DHCP entry for the client machine. To do the install on a new machine, we boot from a custom bootnet image written onto a CD-ROM. The image is nothing special apart from a default boot option of ks to invoke kickstart mode automatically. Once the kernel has loaded and the kickstart configuration file retrieved from the server, the CD-ROM can be removed and the machine left to install unattended.

The operating system component package selection is handled directly by kickstart, and the lists are within the downloaded kickstart file. We have added some additional definitions to allow anaconda to install the Ximian GNOME packages instead of the ones supplied by RedHat. We have a number of standard configurations which reflect the desired partitioning on various different capacities of hard disc, but these can be disparate physical machine types.

Adding additional packages to be installed by the kickstart system is actually not that hard to do. For a major subsystem like the Ximian GNOME desktop, first you need to collect together the necessary RPMs. This is very easy with Ximian – run the installer on a testbed machine and at the end of the session all the RPMs you need will be in /var/cache/redcarpet/packages. Next you copy the necessary RPMs into the kickstart areas . . ./RedHat/RPMS tree and remove the clashing original RedHat packages. You then edit the comps file in the kickstart area . . ./RedHat/base and rename the GNOME entry in there to something like Ximian GNOME and list all of the packages you just installed with the section within braces (less their version numbers). At the end of this process, you run the command genhdlist and the kickstart image is rebuilt and knows these packages. (Genhdlist can be found in /usr/lib/anaconda-runtime on RedHat 7.3 – this is part of the anaconda-runtime RPM which you may need to install manually).

Subsequent re-installations and upgrades do not even require physically visiting the machine. For this we use a small Perl script called "rekick" which installs a kickstart kernel (vmlinuz-kick) and associated initrd (initrd-kick.img) into the /boot in the root filesystem and then modifies the /boot/grub.conf to make the default boot option to use them. It then reboots the machine from this kernel and initrd image pair, which in turn starts the normal kickstart process and the machine is completely reinstalled from the server. Providing you can access the machine via telnet, rsh or ssh and become root, the machine can be remotely re-installed or upgraded.

This has actually proven exceptionally useful in service; it has been quite surprising how many of the reports of problems from users have been solved by simply re-installing the machine, returning it, if you like, to a known good state.

As is usual with the kickstart process, once the initial operating system install is completed, a post-install script is run. We have a very extensive post-install script mechanism which tailors the machine to the desired final configuration based upon configuration files placed on a central NFS file server. This includes configuring for a first boot run of our rsync-based automatic update distribution system.

One of the key concepts of the system is that clients are dataless. All the applications and operating system code are present locally, thus assuring reasonable system performance, but no user data is. This means that the fix-quickly-or-wipe strategy can be applied to the operating system without danger of harming the user's work, and so machines with suspected hardware faults can be swapped out for off-line examination.

One of the neat features of the post install system is that it runs a script specific to the machine (if present, or a default script if not) taken from a central repository on a file server. These files can even be owned by either group administrators or end-users; this enables configuration control of specific machines without requiring root access to the systems. This also allows for a wide range of customization; for example, most machines are pre-configured to allow the casual attachment of USB ZIP drives and Scanners. Additional script components add support for local inkjet printers, DVD-ROM drives, CD-R and DVD-R writers, internal IDE ZIP drives, USB memory keyrings and the like.

All the systems are kept automatically up-to-date (with, from experience, the exception of new kernels). Two technologies are used for this: the first is RPM based, and the second is a locally-written system based on rsync. For RPM packages which mainly consist of operating system and GNOME updates, we use a perl script called autoupdate-ecm which invokes the autoupd script by Gerald Teschl (see Appendix A – this is in the supporting tarball). This has been configured to scan a local RPM repository via NFS and to update those RPMs that have changed since the last time it was run. In turn, this repository is updated nightly from the appropriate update sites of RedHat and Ximian. We very rapidly learnt that kernels should not be updated in this way; allowing automatic kernel updates always seemed to cause problems with VMware, ALSA and the X server, all very crucial services. We also discovered that Ximian's RPMs did not always contain correct dependency information, causing failures if allowed to be updated automatically. (Recently Ximian have been much better at this). This meant that, in the end, we reduced the automatic update system to just pulling RedHat non-kernel updates, and manually performed updates to the RPM repository for the Ximian Desktop tree and the kernel from time to time.

The second technology we use is primarily targeted at application packages. It is often fairly hard work to build a suitable RPM for every package you wish to install, and so we tend not to for locally built software. Since we've been a large Unix site for many years, we have some traditions about filesystem layout; one of these is a hierarchical naming convention for application directories with a couple of locally written scripts to glue it all together. It was important to us to mirror this behavior in the Linux environment, despite the fact that RedHat does not normally make use of /opt. We intended to use this mechanism for updates to /opt and /usr/local only. Over time, however, we added a few more paths, namely such directories as /usr/lib/perl5 and /usr/lib/mozilla_1.0.1, since both Perl modules and mozilla plug-ins are hard to install and manage via RPMs.

Each application is categorized by Vendor, suite, package, and version. For example, a locally compiled GCC 3.2 would live in /opt/GNUdevelop/gcc-3.2, allowing it to co-exist simultaneously with a number of other versions. Within /opt/GNUdevelop/bin there would be symlinks established to the "current" mainstream version: i.e., gcc3 would point to /opt/GNUdevelop/gcc-3.2/gcc; these in turn are all linked into /opt/bin. Each user then has /opt/bin in their path, the search order depending on whether they want local+redhat, or redhat+local as their environment.

On Solaris (and SunOS before it), we had used the track package from MIT's Project Athena to manage distributing updates from a master server to other servers and desktops. The concept here is that the master server holds everything that might be needed (other than extremely specialized items) and each other machine downloads only what it needs. This is determined by a subscription list held on the master server, whose owner can be a user who does not have root privilege on the master server. Over time, we had re-implemented the mechanism in Perl with rsync as the transport medium, while retaining the file format used by the track package. We called this script rsyncer and it is invoked by rsync.slave.sh from cron on a nightly basis on all machines. The rsync.slave.sh script includes a random delay to stop all the machines simultaneously accessing the master server in the middle of the night.

For this project, we added further additional functionality in the form of an include file mechanism to allow for configuring a group of machines using common file subscription lists. We also added a bait-and-switch mechanism for use with the Windows applications (more details of this later).

Authentication and authorization for the entire system is tied into our NIS domain and uses netgroups to control access to each machine. The appropriate /etc/passwd entries to enable access by specific user communities on a given machine is put in place by the post install script. A locally-hacked PAM authentication module adds the user as they login into a locally run Samba server's smbpasswd file, ensuring that the Windows domain authentication matches the Linux domain (there'll be more information on the local samba server in the next section).

Providing Windows

Thus far we've focused the tools of installation and management of a large cluster of Linux machines. However, most of the user community wants (or at least needs) access to Windows applications, and some don't even want to know that anything else exists. If we didn't satisfy this need, the whole project would fail since we would not be providing the single multi-purpose desktop required. With a very broad community to serve, from programmers writing Unix software in Emacs to Secretaries writing documents in Word, this is a challenging goal to achieve.

We've had long experience of using various PC integration products under Unix, starting with SoftPC on a Sun SPARCstation SLC back in 1991, migrating through various generations of emulators including SoftPC, SunPC, and WABI; and using Windows NT-based server solutions such as NTrigue, Citrix MetaFrame and Windows Terminal Server. We fairly rapidly concluded that the NT-based server solutions would not really be viable for this application because of the sheer number of desktops needing to be accommodated and the proportion of the usage of Windows applications we could expect with these systems. Also, the usage of local removable media with the server solution always proved significantly less than ideal.

Since the platform was effectively trying to address both political and technical objectives, we needed to provide as few reasons as possible for people to object to the system. It needed to be as broad in application as possible. With our previous experience of emulators, this was very hard indeed because certain pieces of the emulated environment seemed not to always interact with applications 100% in the way Windows would. For this reason, we concentrated on solutions which could run a vanilla distribution of Windows ideally without any more patches or drivers being required. Of the tools we looked at, VMware seemed the closest to this goal, presenting a virtual graphics card, ethernet card and IDE hard drive controller which default drivers within the Windows distribution could attach to.

For these reasons we selected VMware as our bridge to the Windows world. It's important to realize here that our objectives were not primarily cost saving; we were trying to provide a more capable system with lower support staff involvement, not to reduce the licensing costs for Microsoft software. This may well be different from other people's objectives in choosing to deploy a Linux desktop. Our previous experience with the dual-boot systems was that the number of user problems reported reduced markedly if the Windows operating system was re-installed from scratch on a nightly basis and locked down such that the user's ability to install "neat stuff" had been curtailed. This fitted in well with our dataless client policy.

We also needed to support Windows on a wide range of disparate types of PC hardware if we were to gain significant advantage from standardization and automation. This is where the VMware emulated devices mechanism was a revelation: we could run the same Windows image on everything from a low end Celeron processor, right up to the latest and greatest P4 processor, and even on notebook hardware, without any changes to the device drivers needed on the image. This moved all responsibility for handling the actual physical hardware of the machine into the Linux domain.

The VMware Windows installation consists of two components; a single large disc image file which contains a very basic "C:" drive for the system, and an E: drive onto which all the applications are installed. The C: drive file is presented to Windows as the primary master IDE drive on an IDE controller, and is formatted by Windows itself during the installation process. This means that the filesystem on this virtual disc is entirely windows and doesn't suffer from some of the name mangling problems encountered when translating a Unix filesystem into a representation of a Windows filesystem. It can hide files if it wishes to, make them read-only, and so on.

The other Windows drive, E:, is actually mounted on each machine from a local samba server, and is actually /opt/VMware/apps in the Linux filesystem. This means that our configurable rsync wrapper can determine which Windows world applications are available on each machine as well as which Linux world applications, and again all applications can be updated nightly from the master rsync server. Updating applications within the Windows environment becomes trivial since once installed on the librarian machine, it becomes available on every other machine overnight.

This does mean that all necessary support files (DLLs in particular) and registry entries for every application to be used on any machine are effectively present in the C: drive on all machines. This has not turned out to be as hard as we feared (or we've just been lucky) since we have always been able to get a version of each required DLL that works with all applications. Since the C: drive image file contains the core OS, this includes all the registry entries and necessary DLLs which in turn are automatically updated. The downside of this is that all the Windows images contain the same identity and so effectively have to operate as free-standing machines. Their only access to the outside network is via a virtual interface provided by Linux using NAT (Network Address Translation) technology. This is sufficient to provide full access to the internet and to other fileservers on the local network and has not proven to be a problem in operation. It also fulfilled a major design goal in that it makes the Windows environment significantly less susceptible to remote attacks via the network.

In addition to the E: drive, we used the local samba server to provide the users home directory on the Unix servers as H:, and the entire filesystem of the local Linux box as L: drive (allowing access to Unix-domain shared volumes, /tmp, and extra devices like ZIP drives and memory sticks).

The next issue was to provide the lock-down to avoid tampering. Placing all the applications in the E: drive meant that Linux could be relied on to ensure users could not alter those files. Whenever possible, applications were configured to default to writing user files into home directories on H:, so only the C: drive was left to secure. VMware conventionally offers three modes of operation; on-the-fly updates to the C: drive disc image file; the creation of a transaction log with commit or cancel options at the end of the session (termed undoable) and completely non-persistent. We wrote a wrapper script called "vm" which forced the VMware application into the non-persistent mode; here it keeps a log file in /tmp that does not ever get committed. Our vm script in addition creates a tree of symlinks in /tmp to provide the other expected files, (such as the C: disc image file, NVRAM settings, etc.), in a location where lock files and the like can be created by VMware as it pleases, without requiring that the user have write access to /opt. This allows us to set the entire C: disc image file tree to read-only within Linux. Since the log file grows with every action taken within Windows, we found that /tmp needed to be really quite big on personal machines (3 GB!) but could be smaller on student use machines where the lifetime of each user's session would be significantly shorter.

In addition to serving the various disk partitions, the local Samba server is configured to provide authentication to the Windows Login prompt, to provide access to printers via LPRng, and to handle NT roaming profiles for the user's own settings and document histories. We used a modified Samba server version provided by VMware Inc, which utilizes alternative shared memory locations to allow for the simultaneous operation of a regular Samba server and the one hosting the services for the local VMware application. In future we may need to rethink this as the current VMware supplied version of Samba (2.0.7) does not support the Domain Controller features we need to support a newer version of Windows such as Windows XP.

Our work to date has been based upon Windows NT 4.0, although pretty much any other version could have been used. We're considering updating it to Windows XP, but have recently had issues with VMware changing their pricing policies significantly to our detriment. We're currently reviewing other alternatives before investing significant extra time in development of a new VMware image containing Windows XP.

Since we are using a single C: drive image for all machines, it does mean that all application file associations and all possible start menu entries show up on every machine. To try and reduce confusion, we've created a small Windows application called "NoLicense" which pops up a dialog box informing the user that this application is not licensed for use on this machine. This message also gives them some guidance about what to do about this and who to speak to for advice. We've made use of a feature of rsync to install NoLicense.EXE in place of each application that is not present on the machine. Hence, if a machine does not have a license to run Excel, the EXCEL.EXE binary on that machine is actually our NoLicense.EXE application.

As soon as someone licenses an application for a given machine, all we need do is change that machine's configuration file, and run rsync.slave.sh. The necessary support files are installed into /opt/VMware/apps, and the binary is switched from being NoLicense.EXE to the real executable for the application. So long as it's already an application we have support for within the image, we can install it onto a machine in a matter of 20 seconds support staff time and about five minutes or so actual time. The user does not even need to reboot the session or machine; it can all be done remotely.

To install a new Windows application, we use a development machine. This is typically a server, and VMware is run remotely via X Windows to a system administrator's desktop. For this activity VMware is run as root, and directly rather than through our wrapper script. The setup program for the application is run, and assuming the installation goes smoothly, the transaction log is committed into the C: drive image file at the end of the session. This revised C: image file is then manually transfered to several sysadmin and power user desktops, along with the new directories as needed in the E: drive area (aka /opt/VMware/apps), where it is used for a couple of days to ensure smooth operation of both new and existing software. Once the new disc image has passed this QA process, it is copied from the development server onto the production rsync librarian server and is copied to all other machines overnight.

Finally, we want to mention one final feature of the system. A number of our users have expressed something approaching extreme panic at the thought of having to even see anything that is not Windows. Since VMware has a full screen mode in which the Windows environment takes over the whole screen, we sought to make that option available for them. We did this in conjunction with another feature allowing Windows to be suspended within VMware much as it is on a notebook PC. We suspended the session at the point where Windows was prompting for a login, and kept the suspended file. We then made a session script which uncompressed the suspended session and restarted it in full screen mode. Thus the user can approach the GNOME login screen, choose a session type of "Windows NT" and then log in to GNOME; once authenticated by Linux, the suspended session is resumed with VMware set to full-screen mode. The user then sees a Windows NT login screen with no other decorations or distractions; just VMware talking straight to the X server. When he logs out, the session terminates and returns to the GNOME login screen. The user need never know that he has actually been talking to Linux and the X server all along; rather, his experience appears to have been a vanilla Windows NT session. This seems to be an immense comfort to some users.

Problems and Pitfalls

We've now described the system we've developed in some detail. What remains is to discuss the lessons learnt.

In general, we are comfortable saying that our solution works.

There have been a lot of positives. Installing new machines is easy and quick. The software environment stays up to date and consistent on a large number of machines with minimum intervention. The users get new applications, updates and bug fixes at regular intervals. The stability of the system has proven to be generally good, with Windows NT in particularly being very stable indeed within the emulated environment. Until an unfortunate incident with a bad service pack from Microsoft, we'd not seen a single blue screen of death from NT across hundreds of machines in the space of nearly a year. Generally, user reaction has been positive, and many people seem happy with the system.

However, it would be wrong to say everything has gone right. Our University's preferred supplier of PC hardware seems to have a tendency to change their motherboards on an almost weekly basis, and we are constantly finding that new machines are delivered with bleeding-edge components whose drivers have yet to find their way into the standard Linux kernel. This has been a massive headache for us, involving regular rebuilds of Linux kernels with new drivers added in to support these new pieces of hardware.

We've also found that a very high level of QA is needed after each change to the master copy of the Windows NT C: drive image. For example, we've found if you add too many new printers in a single NT session, there starts to be a noticeable delay before the Properties dialog box appears. This seems to be a genuine NT problem, but it is easy for such a simple change to cause a problem that propagates to hundreds of machines in very short order. We had one Windows NT service pack that caused a spate of system crashes (blue screens of death) for no adequately explained reason; we were forced to back out that service pack in the end. All of these effects have caused us to be very cautious about updating the master image, leading to quite a long lead time on getting new software installed.

We've also had a spate of problems with removable media: the floppy drive doesn't always initialize properly under VMware, and you can't edit a Word document in place on a ZIP disc because of filename mangling issues. We've also had issues with people really not understanding the concept of mounting and unmounting removable media, and the refusal to hand back a ZIP disc when a application still has it open confuses people too.

The problems are not entirely in the Windows world either. GNOME reacts very badly to machines getting switched off while in operation. We frequently have to clean up Gconfd's files to allow people to get back into applications like Galeon, Evolution and Nautilus. Nautilus in particular seems to "go away" and it usually needs system administrator intervention to bring it back.

We have also discovered that the system needs a large amount of RAM, for our original RedHat 6.2 based system 256 MB was pretty much the minimum; for the current RedHat 7.3 based system 384 MB is a sensible minimum. It's not really a surprise but it is a constraint considering that two complete operating systems are being run on the same machine simultaneously. It probably is best not run on anything less than a 700 MHz Celeron processor.

One downside of the ability of the system to let us support a large number of desktops has been problems slipping through the net. In some cases we've found that people have stopped using the machines and brought in a notebook PC from home instead because they couldn't fix a problem with it themselves and have become frustrated. In most cases, we would have not considered the machine to be acceptable in the state we found it – be it through a hardware fault, lack of memory, or having a faulty interim solution to a driver problem that had never been given the final version. When faced with a machine that isn't "standard," most people have little concept of how it SHOULD behave and often assume that a fault "is just the way it behaves." This holds the potential for quite a few PR disasters.

Future Directions

At present we're deploying boxes with RedHat 7.3. We've been taking a look at RedHat 9, but we're yet to see what we'd consider acceptable stability from GNOME 2. Most notable in our standardized, dataless world view is the almost complete inability of GNOME 2 to survive when a second session tries to start using the same dot files as a currently running one. Our users do this all the time when they go to labs while still logged in at their desks, and it's a real problem. Hopefully by the time RedHat 9.1 comes along, we may be able to consider it as a base for the third generation Hybrid workstation.

We now have a problem with VMware's licensing policies, and as a result of this we're reviewing our usage of their technology. We are currently evaluating (again) a range of competitive products. While we still like VMware technically, we are reviewing whether all machines need the level of Windows application compatibility it provides or whether some users can be accommodated with a lower-cost solution; Win4Lin, Wine and CrossOver office are due to be considered.

Conclusions

In summary, we have many hundreds of workstations deployed with a wide user community, including our undergraduate computing labs, all based on a Linux environment. Through the use of virtual machine software these provide a complete Linux and Windows environment to the end users. From an administrative point of view, the centralized configuration, remote administration, automatic updates and zero-intervention installs reduce our per-desktop support issues to an absolute minimum. Linux-based desktops can really be deployed organization-wide, work well and even help to keep that other windowing environment manageable too.

Author Information

Bevis King is the Senior Software Support Officer of the Surrey Ion Beam Centre, part of the Advanced Technology Institute within the School of Electronics and Physical Sciences at the University of Surrey. He splits his time between developing software for the Surrey Ion Beam Centre, and being a Unix/Linux system administrator. Earlier, he was the system programmer for Unix systems within the core staff of the School Computing Service, also at Surrey. He has been at the University of Surrey since 1990, and was previously a Unix system programmer at the Department of Computing at Imperial College. Prior to that he was the Data Processing Manager at a small research company in the City of London. Since discovering Unix in 1985, he's been a devout believer and Unix/Linux proponent ever since. He has served on the committees of a number of user groups over the years including the UK Unify (RDBMS) User Group, the Gould User Society, and The European X Windows User Group. He's worked on many aspects of system administration but in particular has dealt with automated updating, email systems, Oracle DBA activities and Unix desktop software (CDE and GNOME). His personal home page can be found at: http://www.ee.surrey.ac.uk/Personal/B.King/ .

Roger Webb joined the Department of Electronic Engineering at the University of Surrey in 1983 as a Research Fellow with the Surrey Ion Beam Centre. He was employed to look after the computing facilities associated with the research group – a single pdp11. Before this he had spent three years as a post doc at the Naval Postgraduate School in Monterey California, making Molecular Dynamics Studies and Computer Animations, which is still the main area of his research activities. He did his Ph.D. work in the Electronic & Electrical Engineering Department of the University of Salford, on the Mathematical Modelling of Atomic Collisions in Solids.

Current research activities include the use of Molecular Dynamics Simulations to predict the behavior of cluster impacts on surfaces and the use and development of more simple simulations to predict the effects of energetic particle solid interactions. He became a lecturer in the Department in 1986 and after, foolishly, complimenting the Departmental Computer Resource on its help in installing the research groups' Sun network was asked to take over as the Director of the resource after acting as Deputy Director for a year! The moral of this story is never say anything nice about anything – something he now tries to stick rigidly to! He was promoted to senior lecturer in 1993, to Reader in 1997, reaching the dizzy heights of Professor of Ion Beam Physics in 2002. He is currently Deputy Director of the Ion Beam Centre at Surrey having passed on the mantle of director of the computing facilities in 1998. More details available at http://www.ee.surrey.ac.uk/Personal/R.Webb/ .

Graeme Wilford (Wilf) joined the University of Surrey in 1992 and obtained a Ph.D. in computational Physics in 1996. He has been involved in systems administration at Surrey for 10 years and has been heading up the Computer Support team for the departments of EE, Maths, Computing and Physics since 2000. A Linux and open-source enthusiast, he designed and lead the development of man_db for several years. These days, he likes to deploy Linux where other managers would buy an expensive black box. Wilf can be reached at G.Wilford@surrey.ac.uk .

References

SCS Hybrid Workstation Project – scripts and configuration files. Free downloads available. http://www.ee.surrey.ac.uk/SCS/Hybrid/ .

RedHat Linux – A leading GNU/Linux distribution. Product discussed is RedHat Linux, Standard Edition – versions 6.2 thru 9. Free download available. RedHat Inc, 1801 Varsity Drive, Raleigh, NC 27606, USA; +1 (919) 754 3700; customerservice@redhat.com; http://www.redhat.com/ .

Ximian GNOME Desktop – An advanced version of the GNOME desktop environment. Products discussed are Ximian GNOME 1.4, Ximian Evolution 0.10 thru 1.2.4, and Ximian Desktop 2 (GNOME 2.2). Free download available. Ximian Inc, 401 Park Drive, 3 West, Boston, MA 02215, USA; +1 (617) 375 3800; sales@ximian.com; http://www.ximian.com/ .

VMware – Virtual Machine software, product discussed is VMware Workstation, Linux-hosted version – versions 2.0 thru 3.2. Commercial Product, trial downloads available. VMware Inc., 3145 Porter Drive, Palo Alto, CA 94304, USA; +1 (650) 475 5000; sales@vmware.com; http://www.vmware.com/ .

Windows NT, Excel, Word – Software – Microsoft Inc. – http://www.microsoft.com/ .

Samba/Rsync – Samba: SMB (MS Windows compatible) File Sharing software, Rsync: optimized network file transfer software. Version discussed is 2.0.x. Free download available. http://www.samba.org/ .

Autoupdate – Automatic updating of RPM packages from update repository. Written by Gerald Teschl. Free download available. http://www.mat.univie.ac.at/~gerald/ftp/autoupdate/index.html .

Perl – Flexible scripting language written by Larry Wall and others. Free download available. http://www.perl.org; http://www.perl.com .