UoSLinux – A Linux LiveCD distribution for use in Higher Education

Robert L Warrender, John Tindle & Ian Naylor
School of Computing and Technology
University of Sunderland
Sunderland, SR6 0DD

Email:{robert.warrender@sunderland.ac.uk}
{john.tindle@sunderland.ac.uk}
{iank@pitech.plus.com}

ABSTRACT

It is commonplace within most University computing schools to find different computer platforms coexisting peacefully with each other. UNIX as well as Linux workstations, PC's and Apple Macs all have their place within an educational curriculum. Students generally find themselves using at least two major platforms at different times during a typical undergraduate programme.

In distance learning, however, such practice would be considered extravagant with most students only having access to one platform, more often than not a PC running a version of the Windows Operating System. Lack of access to required hardware can lead to compatibility issues between courses run on-campus and their equivalent courses run off-campus. There are also issues relating to illegal use of software. While every effort is made to ensure the legality of software used on-campus, even a simple request that students submit their work in Word format can be interpreted as condoning software piracy in countries where legal software is expensive and where 'bootleg' copies are easily available.

This paper describes a project to help address these issues. We look in detail at a project concerned with the building of UoSLinux for use within certain programmes at the University, both on-Campus as well as off-Campus. This so-called LiveCD is based on the Knoppix/Debbian distribution.

Keywords

Linux, LiveCD, distance learning, software piracy

1. INTRODUCTION

While Linux has been around for some time, in the context of system installation, it cannot be regarded as the most user-friendly operating system in the world. True it generates a great deal of information to the user during the install process but with most of the output being of a highly technical nature, its significance often bypasses all but the most experienced of users. Most novice users do not ever have to load an operating system – this is normally bundled with the machine and pre-setup for the particular hardware arrangement. In the event of a major fault, most users would reach for their 'recovery CD' rather than get themselves dirty trying to install a new operating system.

For those of us who have braved the ordeals of installing a new version of Windows operating system, what is particularly striking is the almost complete lack of detail of what is happening during the installation process. Indeed for most of the time your mind is distracted by a series of still screen images thanking you for your purchase and highlighting many of the benefits of your new operating system. Hype you may say, but it does present a calming distraction and feel-good factor when you consider that the operating system is able to take most major decisions (and get them right) about your installation unaided and with no safety net.

Of course most operating systems prefer to be the centre of attraction i.e. the only operating system loaded on that computer. Decisions and consequences become of lesser significance when you can comfortably instruct the installation process to format your hard drive in the file system of its choice. However, when you are in a situation where you need to keep your "normal" operating system usable but wish to try out other systems, then clearly there are major concerns experienced during the installation of a new operating system. It is an unfortunate fact of life that casual users who would like to 'dip' into an alternative operating system are the users least equipped to perform such a task.

2. THE NEED FOR UOSLINUX

It was in this context we investigated the use and distribution of the LiveCD. LiveCDs allow users to run Linux software on almost any machine without

disrupting or changing any data, partitions etc on their host machine. Indeed as the name suggests, the system uses the CD as its hard drive without needing to store any information on the real hard drive.

Alternatives to the LiveCD were explored such as VMWare and Virtual PC. These are very useful alternatives and ones we also use within a higher education environment. They allow the concept of "guest" operating systems to exist on a host system without destroying the underlying disk file system, appearing only as a large file to the host system. Both types of systems have advantages and disadvantages, however one key difference is that LiveCDs such as Knoppix are available on free download, whereas VMWare and Virtual PC are commercial products with related licensing issues.

One of the major problems we faced with the LiveCD was that it came with a fixed set list of programs (many games related, many editors, alternative browsers, even alternative desktops). What we lacked was a means of customizing the product to a point where it could be used within the curriculum. VMWare and Virtual PC here has a distinct advantage in that both the guest operating system and applications can be distributed as files ready to attach to the Virtual Manager. What was needed was a way of customizing a LiveCD distribution to allow us to run whatever application mix we considered appropriate for its use.

As the overall target for the custom LiveCD (subsequently to be called UoSLinux) would be for use by students from the University of Sunderland (both at home or on Distance Learning courses), it seemed appropriate that the main work be undertaken as a final year project. Much of the text in this paper has therefore been taken from that project.

3. DEVELOPMENT ENVIRONMENT

Knoppix uses on-the-fly compression to allow up to 2 gigabytes of useable software to be installed into a 700-megabyte image. This image is stored in the 'KNOPPIX' directory of the CD and is itself called simply KNOPPIX with no file extension.

In an interview with Knopper, Alexander Antoniades [1], discovered that the Knoppix OS itself is compressed using the gzip algorithms. From tests he had carried out, Knopper noted that there were higher compression ratios available but gzip was a good trade off between speed and compression. The KNOPPIX file system is created using a compressed loop back (cloop) driver, which is mounted at boot.

It was expected that the UoSLinux development would need to take place on a hard drive (rather than on the fly). To ensure sufficient space was available for development purposes, it was felt that at least 4 gigabytes of hard drive space would be needed. This would allow for the CD image to be replicated locally, decompressed and then recompressed ready to be written back to CD for testing, with ample leeway for adding and removing programmes as needed. As Knoppix offers approximately 2 gigabytes of uncompressed programmes and the CD itself is 700 megabytes in size, 4 gigabytes of hard drive space was considered ample. Development was undertaken on an 80-gigabyte hard drive with a fresh install of Mandrake 9.2 Download Edition complete with all Mandrake bug fixes and patches installed. This left enough space to create a partition of a suitable size to be used purely for development purposes.

As Knoppix runs from a ram disk, memory was a concern. The official Knoppix home page suggested that the minimum requirements were:

"20 MB of RAM for text mode, at least 96 MB for graphics mode with KDE (at least 128 MB of RAM is recommended to use the various office products)" [2].

It was expected that development would be much more memory intensive than using the 'normal' office tools available. The PC used for development had 512 megabytes of ram installed, and this was considered to be sufficient. No development was attempted on computers with less installed memory.

Linux and Unix both offer the user a number of shells to work in. A shell is a command line interface for Unix or Linux. While a number of shells are available to the user, the Bourne Again Shell (bash) is the most widely used in Linux [3].

The Korn Shell (ksh) was developed by David G. Korn at AT&T Bell laboratories, primarily to take the features of bash and another common Unix shell, the C shell, and build on both with additional features. Ksh is a complete high level programming language aimed at application developers and is considered ideal for prototyping work. Approximately 80% of the respondents to an AT&T Bell survey of Unix users regularly use ksh. The Korn Shell builds on the functionality of the Bourne Shell and almost any script written for the Bourne Shell will work in ksh [4].

On campus, the University makes use of a number of Sun workstations running Sun Solaris OS. As well as running applications such as Java and Oracle, these Unix facilities are also used on one specific module (COM264), for which ksh scripting is a major component. Indeed this was one of the major motivations for developing the UoSLinux project.

As ksh is a proprietary programme, owned by AT&T Bell labs, Linux users have to search for free alternatives. One such offering is known as the Public Domain Korn Shell (pdksh). This clone of the original Korn Shell deviates from the original, as

noted by the current developer, Michael Rendell [5] Debian make a pre-packaged version of pdksh available and was chosen as the version to be used in the production of UoSLinux. As noted by Robert Luberda, the person responsible for the Debian version of pdksh, this is a mostly complete clone of the original ksh [6].

The directory structure created consisted of a main directory and two sub-directories. The main directory was named KNOPPIX and, within this directory, a master and source directory were created. A KNOPPIX directory was then created in both subdirectories. This gave a directory structure as shown below:

Root - Root partition

KNOPPIX - Main development directory

Master - Local copy of CD contents

KNOPPIX - Directory copied from CD

Source - Directory for uncompressed data

KNOPPIX - Uncompressed files from ram disk

This layout was designed to create a central repository for all work connected with the development of UoSLinux. Roberts et al [7] point out the value of centralized repositories for managing and sharing information in a common format that can be more easily managed and queried. Wolin and Lauer discussed the use of a central repository for code management, describing the concept of centralized storage as “essential” [8]. While their work can be considered informal, it is no less relevant.

4. BUILDING THE DISTRIBUTION

The preparation for and initial development of UoSLinux was undertaken using a CLI. This allowed some experimenting and work towards a final system, developing the system alone without the overhead of a GUI. A shell script can often be more easily and quickly modified than a GUI program. The ability to quickly modify the program, as well as allowing for more control, fit well with the RAD ideal of incremental changes implemented quickly.

The use of switches should be common place to anyone who has used the CLI to perform tasks, regardless of the OS. Even GUI based programs can make use of switches to enhance or manage their functionality. The Microsoft Windows Explorer file management tool is typical of this, with a set of switches available to enhance or control the commands’ functionality.

Unlike Windows, Unix and Linux switches and commands are both context sensitive. This is an important distinction as, for example, the command to list directory contents (ls) can be used with a number of different switches. The command ‘man ls’ displays a typical example of this. There are a total of 54 switches available to the command. If the user were to type ‘ls –x’, they would be given a listing of files and directories sorted by rows instead of the default of columns. If they were to use a capital x as in ‘ls –X’, files would be sorted alphabetically by file extension. Both show the same data, it is the sorting order that changes. This ability to customize can, as mentioned earlier, allow for fine control of the output from the majority of Linux commands, but must be managed carefully.

In common with Unix, the majority of Linux commands have manual (man) pages. At the simplest level, the syntax for accessing these is ‘man <function>’, where function is the command for which you need help. For example typing ‘man man’ at the command line gives a brief synopsis of the man command itself and the available switches for customizing output.

The initial copying of the Knoppix CD had to be made in two stages. As well as copying the contents of the CD itself, the decompressed data also had to be copied from the ram disk. The 700 megabytes off the CD plus the 2 gigabytes approximately of uncompressed data could take a while to copy. As an example of this, on the computer used for development, copying the files needed for remastering took approximately 21 minutes. The command to copy data in Linux is cp. The man page for cp offers a number of switches to be used. The most relevant for UoSLinux were –p, -R and –v. Respectively, these preserve file attributes, copy data recursively and do so verbosely. The use of these switches ensured not only that all data was copied exactly ‘as was’ but that the user was kept informed in line with the tenet ‘Visibility of system status’.

5. FINE TUNING THE RESULTS

The system makes it fairly easy to remaster UoSLinux and add/remove software as needed. That said, it could always be easier and the addition of a comprehensive menu to the doUoSLinux.sh script would be fairly easy. In a similar way, breaking the doCustom.sh script down into a menu driven structure or at least a modular set of components would also be fairly easy to do.

Particular attention needs to be paid to the process of removing installed programs as the process can be made more ‘friendly’ to the user, with string handling being handled through the script so as to minimize the work for the user.

All editing is done through the vi text editor. While this can be a valuable learning experience for students (vi is part of the module COM264), it can also be intimidating to be forced into using it and other options such as ‘pico’ or ‘joe’ need to be explored for a more friendly user environment.

Another one of the design requirements for the project was to include a version of the network simulation software 'ns'. Unfortunately there is a tool already installed in Knoppix called ns and this creates conflicts. That said, ns installs and validates itself well enough and seems to be perfectly useable once the various path scripts are setup. However this has not been fully tested at this stage.

Internationalization still needs to be addressed, UoSLinux currently defaults to American English. This is fine when using a GUI but needs to be addressed for shell scripting. It is to make this change at boot but it would be preferable if users never even had to contend with changing system settings before they even used the OS.

None of these problems alone are major but, added together become an irritating array of problems. These will be addressed in the near future.

6. CONCLUSIONS

The overall project demonstrates the feasibility of the distribution although some difficulties do remain. Knoppix (and thus UoSLinux as a derivative work) seems to struggle to recognize and work with USB devices reliably. This problem is not unique to LiveCD versions of Linux but is one that, in general, needs to be addressed within the Linux community if the OS is to achieve the levels of acceptance aspired to. From experience both Mandrake 9.1 and 9.2 struggle to achieve reliable USB support as well. While not yet tested, the newer Linux kernel 2.6 is rumoured to address a lot of the problems with USB devices.

In terms of its acceptance into Higher Education, we plan to trial its use in the "on-Campus" version of COM264 next academic year. Depending on the lessons learned from that experience and further development of UoSLinux as a distribution, we will trial this at some of our Distance Learning groups where we see the major advantage and useful student experience.

In the meantime, we plan to post the current version of UoSLinux onto its own distribution site at the following URL:

www.UoSLinux.sunderland.ac.uk

Both students and other institutions will be invited to get involved with the project and hopefully take this to a much more advanced and useful stage.

7. REFERENCES

[1] Antoniades, Alexander (2002), Interview with Klaus Knopper of Knoppix, www.osnews.com/story.php?news_id=2305& accessed July 2004.

[2] Knopper, Klaus (2004), Minimum System Requirements, http://www.knopper.net/knoppix-info/index-en.html accessed July 2004.

[3] Schenk, Thomas et al (2000), "Red Hat Linux System Administration Unleashed", Sams Publishing, ISBN 0 672 31755 9

[4] Korn, David (2000), "Korn Shell Overview", www.kornshell.com/info/ accessed July 2004.

[5] Rendell, Michael, Pdksh – The Public Domain Korn Shell, www.cs.mun.ca/~michael /pdksh/

[6] Luberda, Robert, pdksk, Debian Website, http://packages.debian.org/stable/shells/pdksh accessed July 2004.

[7] Roberts, Richard J et al, (2001), "Information Access: Building a "GenBank" of the Published Literature", Science, Vol 291 p 2318-2319

[8] Wolin, Elliot and Lauer, Rochelle (1992), "Thoughts on Code Management", Yale University