Unix and Undergraduate Teaching

Dr Carlo Kopp
SCSSE, Monash University
carlo@mail.csse.monash.edu.au

July 5, 2002

Abstract

Academia has been a traditional training ground for the Unix centric programming environment. A large proportion of the Unix literate community acquired their early skills set while at a tertiary institution. With an increasing proportion of newly enrolled students having a Windows-centric background, introducing students to Unix and providing basic skills presents interesting challenges. This paper will review some of the challenges observed at Monash CSSE and discuss techniques and strategies adopted to ease the transition of students to a Unix centric environment.

1 Introduction

Unix user and basic administration skills are an essential component of the required skills package of a university Computer Sciences graduate. While many industry players have repeatedly predicted the demise of Unix based environments in recent years, the converse appears to be the case. In networking environments, development environments and large server environments, Unix remains the operating system of choice. With the emergence over the last decade of unencumbered Unix clones, be it of the Linux or *BSD varieties, supported by an immense pool of GPL applications, the Unix environment is likely to remain a core feature of computing environments for the foreseeable future.

Historically university Computer Science departments and schools have been a key source of early Unix skills training. Undergraduate students frequently acquired their earliest programming skills at university, on a variety of Unix platforms. Students acquired proficiency in the use of shells, compilers, debuggers, libraries and the X11 GUI environment.

If we look back a decade or two, the student population entering the university envirnment would have typically had minimal apriori computing skills. For all practical purposes they would more than often fit the 'tabula rasa' model - with no prior experience they could be taught from the ground up with an appropriate skills set, building progressively upon earlier subjects until they developed a good understanding of the environment.

Over the last decade we have seen some very significant changes, all of which are a consequence of 'commodification' in the computing world. Developed post-industrial nations, of which Australia is a good example, are characterised by a very high level of private computer ownership in households. With the cost of a very respectable desktop machine typically sitting between AU$1000.00 and AU$3000.00, most households from which university undergraduates are recruited will own one or more desktop computers.

Concurrently, commodification of hardware has seen large scale proliferation of desktop systems into secondary education establishments - most high schools will have at least one or more computing labs for students to learn on.

The consequence of commodification is that the profile of the typical first year university undergraduate has changed - the student may have several years of experience using and often administering a Windows/Intel system, or less frequently an Apple/PowerPC system - and very infrequently a Linux system. Such undergraduate students will enter university from an environment where they were previously regarded to be highly computer literate if not expert, and will perceive their own skills to be of a high standard, and highly valuable. More than often such students will take pride in their Windows/Intel or Apple/PowerPC skills set and actively promote the 'virtues' of both of these systems to their less skilled peers.

What this means in practical terms is that many students will be reluctant to learn another operating system and programming environment, as from their perspective it might mean relinquishing their status as 'experts' in their peer group and then having to compete on a truly level playing field.

2 Why Retain *nix as a Teaching Platform?

An argument which is frequently seen in academia is whether Unix and its derivatives should be retained as the primary platform for teaching programming and system administration skills. The counter-Unix case is often presented in terms of 'more than 90

This argument has not been particularly successful, especially in the more established and mature university Computer Science departments and schools. There are a multiplicity of good reasons why Unix and its derivatives should and indeed usually are retained as the primary teaching platform:

Cost (1): the advent of Linux and *BSD permits the deployment of large numbers of on campus Unix derivative systems at very low cost using cheap commodity hardware.

Cost (2): the advent of Linux and *BSD provides students with very cheap systems for working at home, with large packages of very competitive GPL tools, especially for software development. Dual booting permits these installations to coexist on home computers which are shared by the student with other family emembers.

Portability: an ongoing issue with many Microsoft tools is poor portability of source files between different releases of the same platform. As result, unless the on campus systems and students' home systems are of exactly the same configuration, this produces incessant difficulties with moving work from the comapus home and vice versa.

Performance: university labs and students' home machines are frequently 'hand me downs' which are short on memory, CPU clock speed and disk capacity. If a student has access to a machine with only 16 or 32 MB of memory, a Linux or *BSD installation will allow much more work to be done with a given set of computing resources.

Functionality: almost all Linux and *BSD releases are provided with a very rich package of GPL tools, especially development tools, on the CD-ROM or web accessible ISO images. Therefore the overheads in time (and dollar outlay) to provide a powerful development environment with C, C++, Java, Perl, LaTeX and other languages are minimised.

Source Code: GPL operating systems and tools are provided with source code. This is especially valuable for teaching environments as it provides opportunities for student projects in which existing GPL software can be modified, enhanced or indeed developed from scratch. Students get the opportunity to see tools as software with lines of code rather than black box entities provided by vendors.

Depth: While GPL operating systems now provide powerful GUI desktop environments, they retain access to the underlying environment. This permits debugging techniques which are not practical in binary only proprietary GUI environments. In turn this provides students with useful practical administration and debugging skills.

These are compelling pragmatic reasons for the retention of Unix derivative operating systems as teaching platforms. This is, however, not the only reason why Unix derivatives should remain the central teaching platform.

The second major consideration is the placement of graduates once they complete their University courses. For the forseeable future, Unix derivatives will continue to dominate as platforms for RDBMS systems, large multiuser commercial systems, web servers, web caches, NFS servers and engineering/scientific number crunching platforms. Graduates with a skills package which is deficient in Unix skills are implicitly at a competitive disadvantage against their peers who have strong Unix skills, when seeking employment in many key sectors of the industry. Unless the graduate is specifically aiming for employment as a supporter of or developer on a desktop system, such as Windows, the graduate is competing at a distinct disadvantage. Only a minority of employers are likely to make the substantial investment required to push a graduate up the Unix learning curve to a level of useful proficiency.

The changing workforce environment where a stronger emphasis is placed by many employers on self-training and prior experience held by employees creates a strong imperative for universities to equip their graduates with durable skills sets. Unix skills, like fundamental theory in the discipline, fall very much into this category.

3 Challenges in Teaching *nix

Teaching a student population, a large portion of whom have been previously indoctrinated to use proprietary desktop operating systems, presents some very interesting challenges for Computer Science educators.

Key issues may be summarised thus:

1. Overcoming fear of the unknown, and prior indoctrination with the belief that *nix is intractably difficult to use and administer.
2. Overcoming active resistance by a small minority who are convinced that *nix skills are of no value in the workforce.
3. Facilitating students who have an interest in running Linux or *BSD on their home systems.
4. Weaning students away from the exclusive use of point/click and menu interfaces to shell oriented user interfaces.

The best strategy for overcoming fear of Unix, in the author's experience, has been the gradual introduction to Unix environments. Rather than assault the student with a massive deluge of new information to be absorbed, a problem in its own right for new arrivals in the university system, students are provided with Unix exposure in non-programming subjects - the Unix shell and X11 environment are used as a platform for running other tools, such as document preparation toolsets. Typically two semesters worth of *nix exposure seems to be adequate to dispel most of the anxiety which undergraduates often experience with *nix. A very useful measure is the provision of a 'primer' website explaining why *nix skills are valuable, with links to relevant websites.

Active resistance against *nix is seen from time to time with a small minority of students. Typically such students will lobby heads of departments, associate deans or subdeans with petitions to exclude *nix from their subject syllabus, if they do not get any sympathy from the academic teaching the subject. Perhaps the only recipe for dealing with such individuals is patience, since reasoned argument about their future prospects in industry without *nix skills seldom strikes a chord with such an audience[1].

Facilitating access to *nix for student home system installations might appear to be trivial, but in practice it is often a challenge to organise. Many students are not accustomed to working independently and even the task of locating http://www.linux.org/ and acquiring an installation CD-ROM can present them with difficulties. Three strategies have proven to be useful for dealing with this problem. The first is the provision of a webpage with links to sites such as http://www.linux.org/, http://www.kde.org/, http://www.freebsd.org/, http://www.netbsd.org/ and other sites of interest. Another strategy is handing out free of charge Linux installation CD-ROMs - where the opportunity exists a box full of CD-ROMs at the main administration desk for the school or department is always helpful. A third approach exists where the students may have organised a 'Computer Club' or similar enthusiasts' grouping or body - a annual feature at Monash SCSSE is the 'Linux Install Night' where Student Club members volunteer time to install Linux for their less experienced peers.

Weaning students away from the exclusive use of GUIs can also present interesting problems. The strength of the X11 desktop and xterm is that it provides a relatively seamless transition between the two paradigms. Nevertheless, the author has had repeated instances of students using the *vim* (*vi* clone) text editor menus in preference to the use of standard command set syntax, even many weeks after introduction to the new environment.

While the problem of teaching GUI centric students to use shells can be formidable, other problems have also been observed. The most notable is a propensity for the more adventurous students to compete in terms of who can set up and configure the most esoteric desktop environment or indeed configuration thereof. Some students invest more effort into playing with KDE or Gnome than they invest in their planned laboratory tasks, to the detriment of the end product and the students' grades.

4 Conclusions

The commodification of computing hardware and large scale penetration of proprietray desktops into community households has been a mixed blessing for computing educators. While students arriving at universities may have prior experience, much if not all of that experience will be confined to proprietary desktop environments.

The challenges in transitioning to *nix those students who have been previously exposed only to proprietary desktop systems are not insurmountable, but the task does require a systematic effort and some careful thought.

A key aspect of any such effort is gradual, incremental exposure to the *nix environment, to dispel anxieties resulting from a total lack of prior experience, and from aggressive proprietory vendor marketing.

[1] Some instances the author has dealt with would appear to be motivated by factors such as the student wishing to increase his peer group standing by visibly rebelling against authority, or a simple intent to minimise workload by avoiding new learning tasks.

A second important aspect is facilitating easy access to Linux or *BSD installation CD-ROMs. Guidance by more experienced students who have existing household Linux or *BSD installations has proven to be invaluable in the task of increasing the number of student Linux or *BSD installations.

Industry can further facilitate this process by actively supporting university Computer Science schools and departments by making personnel available as guest lecturers on industry issues and the value of Unix skills, and by the provision of free Linux, *BSD or Unix installation CD-ROMs.

Unix and its derivatives remain a core technology for the industry and therefore Unix centric skills will remain an important component of a Computer Science or Software Engineering graduates' skills set. It is therefore important that this skills set is actively supported by both universities and the industry.

5 The Author

Born in Perth, Western Australia, Carlo Kopp graduated with first class honours in Electrical Engineering in 1984, from the University of Western Australia. In 1996 he completed an MSc in Computer Science and in 2000 a PhD in the same discipline, at Monash University in Melbourne.

Carlo has over a decade of diverse industry experience, including the design of high speed communications equipment, optical fibre receivers and transmitters, communications equipment including embedded code, Unix/SPARC computer workstation motherboards, Sbus graphics adaptors and chassis. More recently, he has consulted in Unix systems programming, performance engineering and system administration. His most notable industry project was the design of the first SPARC workstation motherboards to be commercially manufactured in Australia, in 1993.

Actively publishing in Australia's then leading Unix industry journal, AUUR/Systems/Systems Developer, between 1994 and 2002, Carlo specialised in writing technology reviews and fundamentals tutorials. He now lectures in Computer Architecture and Real Time Systems Design at Monash University. Web page: http://www.csse.monash.edu.au/ carlo/ .