10 things to do with a dead computer in Bristol
Or
Teaching operating system concepts using open source software

Craig Duffy, CEMS Faculty, Bristol, UWE.
craig.duffy@uwe.ac.uk
Cartoons by Bob Lang, Bristol, UWE.
bob.lang@uwe.ac.uk

Abstract
This paper looks at the use of open source software, specifically GNU LINUX, in the teaching of operating systems concepts. Various approaches to teaching operating systems are examined. Teaching many of the key concepts of operating systems and networks through systems administration courses is explored.

Background

At Bristol UWE Computing department we have been using UNIX in our teaching for over 20 years. UNIX has been used to support language teaching as well as our work on operating systems and networking. Over the last decade or so we have been increasingly teaching systems administration, mainly based on Linux, both as a subject in its own right and as a vehicle for teaching operating system and networking concepts.

I will outline 3, rather extremely characterised, ways of teaching operating systems and networking.

A theoretical approach

A very popular approach is to take a rather abstract view of the operating system and to concentrate on the data structures and algorithms that underlie its operation. The textbook by A. M. Lister, “Fundamentals of Operating Systems”, is a good example of this approach. Lister talks of a ‘paper’ operating system. The difficulties with this approach are that there is very little room for practical sessions and the relevance to current usage is very limited. Unless taught extremely well it can become a rather formal presentation of very dry material. It

---

is favoured by a number of teachers and institutions because of its ‘rigorous’, namely mathematical, approach and it has the happy side effect, for the lecturer, of not being much influenced by technological changes therefore requiring minimal updating and changing of course notes. One adherent once confided to me that it is computer science without the computers! From an institutional point of view it is very advantageous in that it can be taught without involving expensive, troublesome equipment, or the even more troublesome people who manage it. The badly taught versions have the other positive side effect of reducing sleep deprivation amongst the student population!

![Diagram](image)

Teaching Operating Systems in a very theoretical manner

**A commercial, ‘training’, approach**

At the other extreme is to take a commercial, training approach, effectively to ‘teach’ whatever is new in the Microsoft stable. This approach tends to be practical in terms of the installation, maintenance and management of the systems, but due to the commercial nature of the software few real insights in to the lower level operations can be gained. In reality, it is training not teaching. The available teaching materials tend to be very expensive, short lived, and are either very dry sets of instructions or marketing material.
Teaching using commercial material makes it removed from the reality of the implementation.

Although this approach is appropriate for the vocational side of computing courses, it tends to lock students into one approach and make them very vulnerable to changing trends. Whilst the newness of the material and the preparation for entry into the job market is motivating for many students, this hardly compensates for the mechanical nature of the courses. It is akin to teaching architecture by looking, from a distance, at the edifices of very impressive looking buildings without being able to know how they were constructed. Teaching this way, one quickly gets the impression that they are just glossy edifices hiding rather dubious design and workmanship.
The tutors sometimes may speculate on the reality behind the flashy façade

A practical approach

“Linus is on holiday at the moment….”

I heard the above comment made by one of my second year undergraduate networks and operating system students during a lab sessions. A group of students had been trying to get a driver for an obscure frame grabber working to create a web cam for the lab sessions, and were thus quite involved in the latest kernel and library developments. The remark, quite apart from cheering me up, is in some ways insightful of the nature of open source software development, in that it is not seen as totally anonymous, and that students, at all levels can feel, and become, involved. In another context, it would be hard to imagine a student being taught with some commercial software package knowing the holiday arrangements of its project leader! In fact, inactivity becomes less of an option; the nature of open source software can make the non-contributor feel somewhat parasitical.
It should not be assumed, however, that teaching open source is synonymous with teaching Linux, there are other, well-established systems available. But Linux is the most known, has the best documentation and is commercially relevant, the later being an important factor in motivating and sometimes winning over students to a particular approach. A fairly substantial problem for a teacher is the evolving, the unkind would say unstable, nature of Linux, which as a consequence, means that a tutor using Linux has to accept changing course notes on a fairly regular basis. One can use more straightforwardly teaching systems, such as Minix\(^2\), which have excellent course material and are, by nature, pretty stable, but you do lose the enormous motivating factor of involving the students in a real-world, mass development.

\(^2\) A. S. Tanenbaum and Woodhull, ‘Operating Systems, design and implementation’. 2\(^{nd}\) edition, Prentice-Hall, 1997. Actually using Linux as the main teaching vehicle doesn’t preclude using Minix, as it is possible to use some parts of the Minix system, or example the file system, within Linux.
### Teaching Systems Administration

Over the past few years we have been using Linux to teach systems administration, which as well as giving the basics of administrative practice also is a very good way of teaching other aspects of operating systems and networks. Our reasons for teaching systems' administration as a separate subject, as distinct from operating systems and networking, were two fold. Firstly, we had begun to notice, around 12 years ago, that our placement students were being given more and more substantial systems administration tasks as part of their placement duties. Secondly, it seemed to us a good way to approach the teaching of operating system and network design and implementation. We had already seen the benefits of this, many years previously, when the more pro-active, and able, students had helped with the management of our LSI-11 system, but the usage of larger UNIX systems, under the control of the computer centre had made this activity impossible.

Our earlier version of such courses were taught using centralised machines running commercial software such as SCO Unix, Xenix, and HP-UX. These systems were good for teaching the basics of systems administration, however due to their closed nature only limited work could be done on operating systems' design and implementation. Then around 8 years ago we switched to Linux, whose various flavours we have stayed with throughout.

The use of Linux allowed us to re-use and recycle old hardware. The practical sessions are not so much Tahoe but more Steptoe! At the beginning of each year the tutors on the course tutors go around begging, borrowing and, occasionally, stealing equipment. The students then have the task of figuring out exactly what they have been given and have to install Linux on to the systems, which are ultimately networked. The systems quite often require rebuilding,

---

3 NetBSD is definitely worth a tutor considering, in that it is free and open, has some good teaching materials around it, and has been very influential on the development of operating systems.

4 Some people, students for example, may query the view that text books on MINIX, NetBSD and the paper OS’s are affordable, but in comparison with some of the prices charged by Microsoft press, they are real bargains!
reconfiguring, and, occasionally, repairing. The sessions allow the students to work at their own pace, in that for some simply installing, configuring and getting onto the network are major achievements, whilst others will have much grander aspirations for their system.

The students (there are about 180 per year) are organised in to groups of 5 or 6. They then must undertake a series of tasks, from partitioning and installing their machine, through to rebuilding the kernel and installing the machine on the network. The students are then expected to offer services to other machines on the network, such as NFS, print serving, and so on. All the machines must be made secure and all of the work on them logged.

The development of more user-friendly installation and administration front-ends has lead to some thinking about whether their use is acceptable in that they permit students to go onto more interesting areas of systems administration, or whether the students should understand the processes behind such scripts. The use of older equipment tends to prejudice the issue, in that the newer interfaces tend to fail with older equipment, but my predilection is that the students should understand what goes on behind the interfaces, for no other reason that they will have to gain such knowledge when the automatic systems fail.

We have found that the use of both Linux and the systems administration format useful in teaching more traditional areas of the syllabus in slightly different ways. Security can become a very dry, formal subject, taught at a distance from the equipment and practical work. Using the system administration format, security becomes a live issue. The students are all aware that their machine is very vulnerable in a reasonably open laboratory, surrounded by, to all intents and purposes, competitors’ machines. An awareness of security and security issues becomes engrained very early on in the course and the motivation of the students is extremely high.

Likewise, teaching networking protocols is no longer a matter of slowly, snoring ones way up the ISO OSI stack, but a real question of how to get the machine connected. The relationship of hardware, MAC layer to TCP/IP is much more clearly understood when allied to practical experience of netmasks, arp and route tables.

**Conclusion**

Our experiences using open source software, and, specifically Linux, has lead us to feel that it can make important contributions to both the teaching and learning process. The ethos of such movements is of active participation, and active participation normally means lively and rewarding classes. The students are encouraged to see the software development process as one to which they can be actively engaged in as a participant, rather than one to which they are simply subjected to.