Informatics in Education, 2003, Vol. 2, No. 1, 15–20
© 2003 *Institute of Mathematics and Informatics, Vilnius*

Open Source Software in Education: a Report of Experience

Hagen HÖPFNER
Otto-von-Guericke-University of Magdeburg
P.O. Box 4120, 39016 Magdeburg, Germany
e-mail: hoepfner@iti.cs.uni-magdeburg.de

Received: March 2003

Abstract. Most students use computers without thinking about how the software and the hardware works internally. So, their occupational outlook depends strictly on the equipment used by a potential employer. Obviously, in case of shoals of applications students which know about the handling of miscellaneous systems are preferred. The concept behind UNIX, BSD or Linux is to provide a set of small but efficient and specialised chiefly tools. More complex problems can be resolved by combining these tools. So, learning how to work with a UNIX-like operating system is learning how to break down a problem into manageable subproblems. But, schools are typically under-funded. That means, that they are unable to bye commercial UNIX-systems. A solution for this dilemma is the usage of Open Source Software like Linux. This paper presents a report of experience which is based on three lectures and practical courses on UNIX/Linux. We will show how to introduce the concepts of Linux as well as how to advance the motivation of students by facilitate individual successes.
Key words: open source software, Linux, UNIX.

1. Introduction and Motivation

Due to the increasing usage of information technology in every industrial sector education has to consider the special requirements caused by the heterogeneity of deployed hard- and software. Not only computer science students but also students of business sciences or natural sciences have to know the functionality of miscellaneous computer systems. They are used to handle standard personal computers running an Microsoft or Apple operating system. The idea of such GUI[1]-driven operating systems is to manage everything in an standardised way. In fact, here a small number of application programs (e.g., a finder) manage a lot of different problems (e.g., finding files, finding computers, finding shared folders). For the end-user on a home computer this approach may be advantageous. But, if someone have to solve problems which require a non-standardised solution, such approaches fail. Here, the UNIX-philosophy can be an aid. UNIX-systems are designed as collections of small but efficient and specialised chiefly tools. In our example that means, that we have different search tools (e.g., *nslookup* or *find*) and each

[1]Graphical User Interface

one can find a special kind of objects. But, the handling of such systems is quite different. Complex problems, which are easily solvable by GUI-based, graphical systems, have to be broken down into smaller subproblems which are manageable by the UNIX-tools.

The greatest problem in order to teach the handling of UNIX-systems for most schools is the price. Nearly all commercial UNIX operating systems (with the exception of SOLARIS) require special, expensive hardware (e.g., HP–UX needs HP-machines, IRIX needs SGI-machines). A solution of this dilemma is the usage of open source software (OSS) like Linux. This operating system is nearly for free and can be used with a standard personal computer in addition to a Microsoft or Apple operating system.

This paper is an report of experience. We present our approach for teaching the usage and the concepts of UNIX-based operating systems with the aid of open source software (especially of Linux). These results are based on the experiences of two lecture courses (90 minutes per week) and practical courses (180 minutes per week) at the University of Applied Science in Brandenburg (Brandenburg, Germany) and a lecture course at the "Verwaltungs- und Wirtschafts-Akademie" (VWA) in Magdeburg (Saxony–Anhalt, Germany).

The remainder of this paper is structured as follows: Beside the presentation of the contents of the lecture courses we will show how to advance the motivation of students by facilitate individual successes in Section 2. In Section 3 we explain the chosen examination. Finally, we summarize the paper and give a conclusion in Section 4.

2. OSS for Teaching UNIX

There are a lot of aspects of UNIX which could be taught. The question was: what aspects are useful and which aspects are teachable regarding given time restrictions and previous knowledge. As mentioned in Section 1 the course was supplemented by practical exercises. So we decided to take such parts which can be tried out. In fact, lecture and practical exercises where not strictly separated accomplished. The resulting agenda is:

1. Introduction (obligatory)
2. First steps (booting, login, logout, shutdown)
3. Terminals, Processes, Utilities
4. Shell-Introduction
5. Shell-Handling text files (excursus: emacs)
6. Shell-Programming
7. X-Windows
8. Administration
9. Software management
10. UNIX/Linux networks

After the obligatory introduction lecture (motivation and history of UNIX) students could take their first steps. They started the computers and saw the output of the boot sequence of SuSE Linux 7.1 (SuSe, 2003). Finally they got the textual login prompt of run level 3. Basing on this first "shock", after the first login and after testing some rudimental commands (*ls*, *cd*, *mkdir*, *rmdir*, . . .) the boot process was elucidated. The lecture was closed by shutting down the computer manually with *shutdown*.

The third part of the lecture course was called "Terminals, Processes, Utilities". It contained some historical stuff about the concepts of terminals and servers, how this concepts can be associated with UNIX, the three kinds of I/O[2] channels (input, output, error), some theoretical considerations on processes and finally the necessary tools and methods for handling terminals, channels and processes. In fact, the students where mostly interested in the last part. Here they could modify the terminal settings (*stty*), observe the process table (*ps*, *top*) and "kill" processes by sending signals (*kill*). The handling (|, >, <, . . .) of channels was the first illustration of the UNIX-philosophy (see Section 1).

The following three parts which are titled "Shell" where the central parts of the course and covered a overview of the capabilities of the different Shells (*sh*, *korn*, *bash*, *tcsh*, . . .) as well as an insight into the relevance of text files for a UNIX system. Students learned that most tasks of the handling of UNIX systems (e.g., administration, configuration, logging) are associated with simple text files. Afterwards, we looked at the programming of shell-scripts. Here we focused on *bash* and its control structures (*if*, *while* etc.). Students wrote small scripts and got the direct success or disappointment.

The typical participant of this course was a Microsoft Windows user and familiar with this operating systems. Therefore, they had already seen at least one graphical user interface. So we could skip an introduction into this subject and went directly to the capabilities of the X-Windows-System (XFree86). The most interesting aspects were the display forwarding and the multitude of different window managers. The students were allowed to try out these capabilities by themselves. Finally we changed the default run level into 5 (boot ends in a graphical login) by editing the text file */etc/inittab*.

By the time all students used the same login. That means, that they could tease one another. The most favoured trick was to open the CD-ROM drive of a foreign computer (remote login with *ssh* combined with *eject*). They could also modify stored data, but they did not do that. So we had a motivation for looking how to administrate (adding, modifying, deleting of groups and users) a UNIX-system. The ending of this part was the programming of a script for adding a great many of new users (with different groups).

Due to the fact, that software management under UNIX and Linux differs from Windows- or Apple-approaches, we decided to take a look at the different ways to install and manage applications unter Linux. After a short overview of available package formats (binary, source code, *rpm*, *deb*, *tar*, ebuilds ...) we focused on the Red Hat Package Manager (RPM) and the compilation of the source code. The practice was the programming of a shell script for installing programs (resolving dependencies, finding necessary packages, etc.).

[2]input/output

The course was finalised with some aspects of the network capabilities of Linux. Because of the given time limitations we had to pick out a couple of interesting aspects. We decided to look at the network file system (nfs) and gave the necessary introduction (routing, ip-adresses, dhcp, etc.). The last step – providing and mounting nfs-folders – was the practical task.

3. Examination

In contrast to the contents of the course, the examination differs regarding the educational institution. At the VWA we where asked for a written examination. Because this is an standard procedure, we will skip this here and fade to the examination at the University of Applied Science of Brandenburg. Here the students had to solve the following configuration and installation tasks in groups of at most three persons:

1. Set up an PC with GENTOO-Linux!
   Gentoo Linux (2003) is a source code based distribution which requires a good knowledge of the internal functionality of Linux. X-Windows was not required.
2. Set up a Linux PC as a gaming station!
   We took the "old" SuSE 7.1 distribution. The students had to update the X-Windows and to install and configure 3d drivers from the source code.
3. Set up a Linux PC as file server for Microsoft Windows clients!
   In fact, the task was to install (from the source code) and configure a SAMBA-server (Samba, 2003).
4. Set up a Linux PC as a multimedia station!
   Current distributions are able to play many media files, but not so the old SuSE 7.1. Therefore, students had to install a media playing software as well as the required codecs from the source code.
5. Enable a Linux PC to run Microsoft Windows applications.
   Here the students could experience with the emulation software WINE (Wine, 2003).

After finalising their projects students had to present their results. The presentation slides where often made with Open Office (OpenOffice.org, 2003) even without a previous constitution of the presentation software.

4. Summary and Conclusions

In this paper we presented a report of experience of our lecture and practical courses on UNIX/Linux. We introduced the necessity of this field of computer teachings by illustrating the differences between handling a Microsoft Windows or Appel system. Furthermore we discussed the contents of the lecture course and pointed out some tasks of the practical course. Finally we described the chosen examination.

Concluding the experience of this courses we can mention that UNIX is not only interesting and beneficial for students of computer science but also for students of other sciences. So far we did not mention that the student in Brandenburg where studying business informatics with main focuss on business and that the students at the VWA took evening classes.

The resume of these three courses is that open source software can be adapted for similar fields of knowledge. We used Linux and much more open source application for teaching the handling of UNIX-like operating systems. The successes of the practical examinations shows us that our approach works fine.

References

Gentoo Linux (2003). `http://www.gentoo.org`
OpenOffice.org (2003). `http://www.openoffice.org`
Samba (2003). `http://www.samba.org`
SuSE The Linux Experts (2003). `http://www.suse.com`
Wine: An Open Source Implementation of the Windows API (2003). `http://www.winehq.org`

H. Höpfner is a staffer of the Institute of Technical and Business Information Systems of the Department of Computer Science of the Otto-von-Guericke University of Magdeburg. His primary research interest are mobile databases and information systems and federated information systems. Furthermore, he wrote a couple of Linux-related articles for the German journals “Linux Magazin” and “Linux User” and for the English journal “Linux Magazine”.

Atvirojo teksto programinė įranga švietime: patirties analizė

Hagen HÖPFNER

"Tebūnie tai paprasta ir kompaktiška programa!", – šis šūkis galioja visoms UNIX šeimos operacinėms sistemoms. Sudėtingos problemos gali būti išspręstos panaudojus paprastas, tačiau efektyvias priemones. Ši tiesa yra aktuali ne vien informatikos studentams. Ji gali būti naudinga ir mokantis vadybos ar gamtos mokslų. Kaip bebūtų, UNIX operacinių sistemų mokymas reikalauja daug brangios programinės įrangos. Brangiausios UNIX operacinės sistemos priklauso nuo specialios aparatūros, kuri, daugeliu atvejų, yra visiškai neprieinama aukštosioms mokykloms ar fakultetams, kuriuose nėra dėstoma informatika. Todėl buvo nutarta reikalingų žinių dirbant su UNIX mokyti naudojantis "Linux" operacine sistema. Šis straipsnis remiasi patirties, įgytos dėstant paskaitų ciklus Brandenburgo taikomųjų mokslų universitete bei Magdeburgo vadybos ir ekonomikos akademijoje, analize. Be abejo, nėra įmanoma per vieną paskaitų ciklą supažindinti su visais UNIX operacinės sistemos aspektais. Dėl to buvo atrinkta tik tai, kas reikalingiausia. Buvo nutarta apžvelgti "Linux" sistemų pajėgumus bei susitelkti ties tuo, kas nesunkiai gali būti išbandyta. Kurse buvo pateiktas įvadas į temą (čia supažindinama ir su UNIX bei "Linux" istorija), nušviesti kai kurie sistemų įkėlimo aspektai, pateikiama esminė informacija apie pagrindinius procesus, terminalus, mokoma, kaip elgtis su kompiuteriais, kuriuose nėra grafinės vartotojo sąsajos, aptarti "XWindows-System" pajėgumai bei dalykai, susiję su kompiuterių tinklais. Studentams buvo pateikiamos nedidelės užduotys. Tokiu būdu išmoktą teoriją jie nesunkiai galėjo pritaikyti praktikoje. Studentų žinioms patikrinti buvo parengta mokomoji projektinė medžiaga. Studentai, dirbdami nedidelėmis grupelėmis, turėjo įdiegti ir sutvarkyti atvirojo teksto "Linux" operacinę sistemą. Tokiu būdu jie galėjo pademonstruoti tai, kaip suprato jiems pateiktą medžiagą.