UNIX Scripting and High-level Language Education Using an Emulator

R. W. Skeith
Department of Computer Engineering
University of Arkansas
rws@engr.uark.edu

M. A. Thornton
Department of Computer Science and Engineering
Southern Methodist University
mitch@engr.smu.edu

Abstract

Entering and continuing engineering students need to learn skills in the use of high-level languages and the use of the *UNIX* operating system[1] including the development of shell scripts. In the past, this requirement has been very challenging to educators since it requires access to a laboratory containing (sometimes expensive) computers that are *UNIX*-based workstations. The widespread availability of the *LINUX* operating system[2] helps to alleviate this problem somewhat since the operating system is free and associated high-level language compilers are also freely available through the *GNU* project[3]. Unfortunately, the skills required to successfully install and use *LINUX* often precludes its use as a classroom tool that students can easily maintain. An alternative and free solution based on the use of a *UNIX* emulator that runs under Microsoft® Windows operating systems is described here.

INTRODUCTION

Recently, several *UNIX* emulators that are easily installed and used under the Microsoft® *Operating Systems* (OS) commonly referred to as "Windows" have become available[4,5,6,7]. This paper describes the authors' experiences in using these tools in an undergraduate setting for the purpose of teaching the use of *UNIX* and various high-level languages such as *PERL*, **FORTRAN**, *C* and *C++*. In addition to these programming languages, the tools are also useful for teaching more advanced concepts such as *UNIX* shell scripting; all while existing in the native operating system environment Microsoft® provides which is by far the most predominant installation in PCs found today.

Many entering students in engineering programs are proficient in the use of basic office tools that run on PCs. In reality, a large majority of these students have never been exposed to any operating systems interface that did not include a *Graphical User Interface* (GUI). Consequently, while these students are quite impressive in their skills with a GUI based interface, the presentation of an operating system based on a command-line interface such as the one used in *UNIX* can be quite daunting. A large percentage of industrial CAD tools and other engineering aids are currently available in the *UNIX* environment only. This leads to the dilemma of trying to teach these entering, "computer-literate" students skills in *UNIX*.

The emergence of the *LINUX* operating system solved many problems in this regard. Although great advances in installation programs and other aids in the use of *LINUX* have emerged in the

past few years[8,9,10], it is still true that once the *UNIX* neophyte has installed this OS, she/he is forced to administrate it.

Here, we make the case that it is much easier to install a *UNIX* emulator such as *UWIN* [4] under the comfortable Windows-based operating system than it is to completely abandon the environment that the entering student is used to dealing with in favor of a sometimes non-user-friendly environment such as *LINUX*.

In particular, a summary of some of the experiences of using *UWIN* (Unix for Windows) in the educational environment is given. *UWIN* is a package that provides the necessary software to develop and execute Unix applications on a Windows NT or a Windows 98 system. The *UWIN* system provides a means of teaching a wide range of computer engineering and computer science courses. For example, introduction to *C/C++* programming, use of the *UNIX* operating system, and shell script programming to mention a few of the teaching applications.

In the remainder of this paper, we discuss experiences in the installation of the *UWIN* system as contrasted with *LINUX* for the new *UNIX* system user. Next, we describe how the system can be used to teach shell scripting and other programming languages. The following section gives a brief overview of the various compilers that are available for high-level languages. In particular, it is noted that equivalent compilers based on the "Windows" operating systems have a significant associated cost as compared with the GNU public license. Finally, a summary is given based on the benefits of using this freely available software (for educational purposes) versus populating and maintaining a laboratory with equivalent commercial software.

OVERVIEW OF THE UWIN SYSTEM

One of the primary purposes that stimulated the development of *UWIN* was to provide a *UNIX* environment on a Microsoft platform. The *UWIN* base system which provides this environment is the result of the efforts of David G. Korn of AT&T Laboratories[4,11]. The base toolkit provides more than three-hundred *UNIX* shell tools and utilities. A paper, "UWIN-UNIX for Windows", by David G. Korn which details the development of *UWIN* can be found at www.research.att.com/sw/tools/uwin. Another paper by David G. Korn, "Porting UNIX to Windows NT" can also be found at this site and describes the efforts to build an UNIX interface layer on top of the Windows NT system.

There exist several alternatives to *UWIN*. NuTCracker from DataFocus, Interix from Softway Systems, and cygwin32 from Cygnus. The reason for our selection of *UWIN* versus these other alternatives was the *UNIX* like flavor and the availability. Furthermore, there is no license limit period for *UWIN* educational users, research users, and AT&T users[12]. Commercial users can purchase a license from Global Technologies, www.gtlinc.com or Wipro, Inc., www.wipro.com.

Educational users can download the *UWIN* system from www.research.att.com/sw/tools/uwin. Interested users should also visit the Wipro, Inc. web site as much information and documentation is available at this site. Some of the packaging and development for *UWIN* was accomplished in India by Wipro, Inc. A basic *UWIN* system can be installed in about 35 Megabytes, however, if additional software support is anticipated such as the gnu packages more than double this amount

is required. *UWIN* executes better on a Windows NT system but will also operate on Windows 98.

The *UWIN* system installs rather smoothly and this ease of installation was one of the main reasons we use *UWIN* versus some of the other alternatives. The basic installation file (the uwinbase file) is a self-extracting archive that contains the basic distribution. The *GNU* packages, Perl and other software package can be installed from the root directory that is selected when the uwinbase file was installed. The gnu packages can be downloaded from www.xraylith.wisc.edu/~khan/software/gnuwin32/egcs/ uwin.html. Additionally, many network applications are fully functional and included in the base distribution such as ftp, telnet, etc.

TEACHING EXPERIENCES USING UWIN

The *UWIN* package was used in two undergraduate classes. One of the courses was at the freshman level and it included an introduction to programming using the *C* programming language. The other course was at the senior level which covered shell script programming and *UNIX* tools and utilities. The *UWIN* system was installed in a PC laboratory which both classes had access.

The students in both courses were provided with a CD-ROM that included the *UWIN* system plus additional software support packages and a *readme* file which provided the instructions on installation. Many students installed the system on their personal PCs and continued to use the system after the class was over. Of particular benefit was the ability of the students to have the powerful GNU programming language compilers for *C*, *C++*, **FORTRAN**, *PERL* and other languages for free. In contrast, the commercial Windows-equivalent compilers can be quite costly.

Also, software engineering tools such as version control tools (**sccs**, **cvs** and **rcs**) and the **make** utility which are commonly used in industry and are available and useable under the *UWIN* environment. In addition to the classes, several graduate students have also installed the system on their PCs so that they can work on their programming projects without using a modem to connect to on-campus *UNIX* systems.

One of the more powerful aspects of the *UNIX OS* is the ability to generate powerful commands through the use of the pipe (eg. |) and **stdio** redirection capabilities. The students were exposed to the use of these techniques along with the usage of system commands such as **grep**, **awk**, **sed**, **find**, etc. An outline of a typical 2-hour course at the sophomore level follows.

Catalog Listing (Symbol, Number, Title, Description):
ECE 2xx2. Engineering Applications in UNIX. (2) (Prerequisite: ECE 3714). One hour lecture. Three hours laboratory. A survey of topics using a UNIX based computer system including C++, Perl and various UNIX tools.

1. DETAILED COURSE OUTLINE

(2 hrs.) I. Introduction to the UNIX Operating System
- A. Introduction and Installation
- B. Basic System Commands
- C. Basic Text Editing
- D. The System Shell Interface

(2 hrs.) II. Data Representation and File Systems
- A. Integer Representations
- B. Floating Point
- C. Binary Data Files and I/O
- D. File Types, Directories and Structures

(2 hrs.) III. System Tools
- A. grep, find
- B. Shell Scripting, awk, sed

(2 hrs.) IV. Software Development
- A. Make
- B. rcs/cvs

(1 hr.) V. Mid-Term Exam

(3 hrs.) VI. Perl Scripting
- A. Introduction to Syntax
- B. Using System Calls with Perl
- C. Other Perl Topics
- D. Engineering Applications with Perl

(3 hrs.) VII. C++
- A. Object-Oriented Design
- B. Encapsulation, Overloading

(1 hrs.) VIII. Final Exam

LABORATORY

(6 hrs., 2 labs)	I.	Basic Unix Commands, Editors, Shells
(6 hrs., 2 labs)	II.	Files, Navigating the OS
(6 hrs., 2 labs)	III.	System Tools
(6 hrs., 2 labs)	IV.	Software Developemnt
(9 hrs., 3 labs)	V.	Perl
(9 hrs., 3 labs)	VI.	C++

2. METHOD OF INSTRUCTION

Laboratories	50%
Mid-Term Exam	25%
Final Exam	25%
	100%

AVAILABLE COMPILERS AND TOOLS WITH UWIN

The students in both courses were provided with a CD-ROM which included the *UWIN* system plus additional software support packages and a *readme* file which provided the instructions on installation. A shell script program included on the CD-ROM simplified the task of installing the included software. Compilers available for use included *C*, *C++*, **F77**, and *PERL*. Tutorials on **vi**, *UNIX* and *PERL* were also contained on the CD-ROM.

Both courses required the student to use **vi** as the text editor. The **vi** editor included in the *UWIN* package allows the user to position the cursor in both the edit and the command mode. Although, this may seem to be a minor feature it is a very frustrating experience to have to be in the command mode to move the cursor and this feature allowed students to more easily overcome the learning obstacle for **vi**. The system provides the necessary support so that students were able to build libraries, use **make**, and develop their applications as if the operating system was Unix.

UWIN allows for shell scripts to be written and executed as does *UNIX*. The primary tools **sort**, **grep**, **sed**, and **awk** work quite well. This system is an excellent way to introduce the student to regular expressions while operating in the Windows system so it can be very beneficial as a prerequisite course to compiler design and/or formal languages. In fact, it is possible to teach a compiler design course based on **lex** and **yacc** (or the GNU variants **flex** and **bison**) for the implementation of the front end.

SUMMARY AND CONCLUSIONS

An inexpensive way (in fact, free for educational and research use) to teach *UNIX OS* fundamentals and to obtain compilers for high-level languages was described. This is particularly helpful for students who are only familiar with "Windows" based *OS*es since the emulator tools run directly within this environment. Furthermore, many different programming languages can be taught without purchasing a separate compiler and licenses for each. The rapidly decreasing cost of PC hardware makes this choice attractive as compared to purchasing relatively more expensive *UNIX* based workstations.

References

[1] D. M. RITCHIE and K. THOMPSON, (July 1974), The UNIX Time-Sharing System, *Communications of the ACM*, vol. 17, no. 7, pp.365-375.

[2] M. WELSH and L. KAUFMAN, (1995), **Running LINUX**, O'Reilly and Associates, Sebastopol, California.

[3] GNU WEBSITE, (2001), GNU's Not UNIX! – the GNU Project and the Free Software Foundation, **http://www.gnu.org/**.

[4] D. KORN, (January 1997), Porting UNIX to Windows NT, Proceedings of the USENIX Annual Technical Conference, Anaheim, California.

[5] G. J. NOER, (2001), Cygwin: A Free Win32 Porting Layer for UNIX® Applications, webpage article, Cygnus Solutions, **http://sources.redhat.com/cygwin/usenix-98/cygwin.html**.

[6] DATAFOCUS, INC., (2001), MKS Toolkit Cross-Platform Developer's Guide, Available Electronically, **http://www.datafocus.com/misc/more_info.asp**.

[7] SOFTWAY SYSTEMS, Interix Development Kit, (2001), **http://www.omicron.se/produkter/softway/intdevelop.html**.

[8] D. CANTRELL, L. JOHNSON and C. LUMENS, (2001), Slackware Linux Essentials The Official Guide to Slackware Linux, Available under the GNU Public License, **http://www.slackware.com/book/**.

[9] REDHAT INC., The Redhat Linux for Intel Processors, (2001), Available Electronically, **http://www.redhat.com**.

[10] SUSE INC., SuSE Linux 6.4 for Intel, (2001), Available Electronically, **http//www.suse.com**.

[11] D. KORN, (2001), UWIN- Unix for Windows, **www.research.att.com/sw/tools/uwin.**

[12] UWIN 2.25 OVERVIEW, (2001), **www.research.att.com/sw/tools/uwin.**

[13] D. KORN, (2001), Porting UNIX to Windows NT, **www.research.att.com/sw/tools/uwin**.

Ronald W. Skeith

Ronald W. Skeith received the Ph.D. in Engineering from Arizona State University. He is currently employed as a faculty member at the University of Arkansas where he has served for over 36 years and he is a Professor of computer engineering.

Mitchell A. Thornton

Mitchell A. Thornton received the Ph.D. in computer engineering from Southern Methodist University in August 1995. He has 6 years of industrial experience and 7 years of academic experience. Currently, Mitch is an Associate Professor of computer science and engineering at Southern Methodist University.