2004-600

An Online UNIX System Programming Course for Computer Engineering Students

Anthony M. Richardson

University of Evansville

Abstract

This paper describes a UNIX system programming course that is currently offered at the University of Evansville. The course is taken by students from both the Computer Engineering and Computer Science programs. The paper first provides motivation for offering such a course in the Computer Engineering curriculum. Several methods for offering such a course on a campus where access to UNIX computer systems is limited (or even non-existent) are then discussed. Course topics are listed and all course assignments are described. A complete set of lecture notes that can be used for either classroom or on-line instruction is available.

1 Introduction

UNIX continues to increase in popularity and market share. The faculty of the Computer Engineering department at the University of Evansville (UE) felt that an elective course should be developed that would give UE students practical experience in UNIX system programming. The focus of the course would be on file system operations, process creation, interprocess communication, and socket programming. In addition shell programming, Perl programming, and graphical programming using a modern UNIX graphical toolkit should also be covered. Since the majority of our students have little UNIX experience it is also necessary to cover UNIX history, philosophy, and development tools. The course would complement an existing required course in Windows system programming.

Although the computer science laboratory at UE contains computers that will dual-boot Microsoft Windows XP and Linux, requiring students to do all work in the lab was felt to be far too restrictive. One of the goals in developing the course was to provide means by which students could do all course assignments remotely. The majority of students taking the course would either own or have access to an Intel compatible computer with some version of the Windows operating system installed. Although having students install some version of UNIX to complete course assignments is an option, it was not felt that this should be a requirement. Installing a second operating system and setting a machine up to dual-boot between operating

systems is not easily accomplished without expert assistance. So several options that allow students to complete assignments either on or off campus were developed. These are described in a later section.

There are several UNIX like operating systems available. We choose to base the course on Linux primarily because we already had a Linux server and computer lab on campus. The fact that the faculty had experience with Linux and that Linux is freely and readily available were also factors in basing the course on Linux. It also has the same application program interface (API) to services as traditional UNIX systems.

A complete set of web-based lectures was developed so that the course could be taken on-line. The course syllabus, the lectures, and the assignments are available from the ACM Special Interest Group on Computer Science Education (SIGCSE) web site[1].

2 Course Structure

The class met three times a week for fifty minutes over a 15 week term. During each lecture period there would be a 30-40 minute multimedia presentation followed by a 10-20 minute in-class exercise. The lecture slides were uploaded to a web site so that they were available to students taking the course on-line. Final grades were based on the performance on two exams and eight out-of-class assignments.

3 Topics and Assignments

There are a number of topics that the course developers felt should be included. Figure 1 depicts the major topics that are covered. There was not time to go into great depth on any one particular topic, but there was sufficient time to provide a sound introduction to several aspects of UNIX system programming. Figure 2 shows the required programming assignments. A brief description of the course topics and corresponding assignments follows. A day-by-day schedule and more complete descriptions of the assignments can be found through the SIGCSE web site.

3.1 Introduction to UNIX

During three successive lectures we covered UNIX history and features, and described some of the options available for getting access to a UNIX programming environment. We also discussed how to compile and link programs, how to create and manage static libraries and UNIX programming philosophy. Common UNIX commands, special files and devices, file permissions and UNIX file systems were also covered.

3.2 Shell Programming

Two lectures were used to discuss shell (bash) programming. Redirection, pipes, variables, decision and iteration structures, positional parameters, command and arithmetic substitution were covered.

As an assignment students wrote a script that recurses through a directory identifying files that are larger than 100 KB and that have not been accessed in the previous 30 days. The script would then prompt the user to either 1) do nothing, 2) compress the file in place, or 3) compress the file and move it to an archive directory.

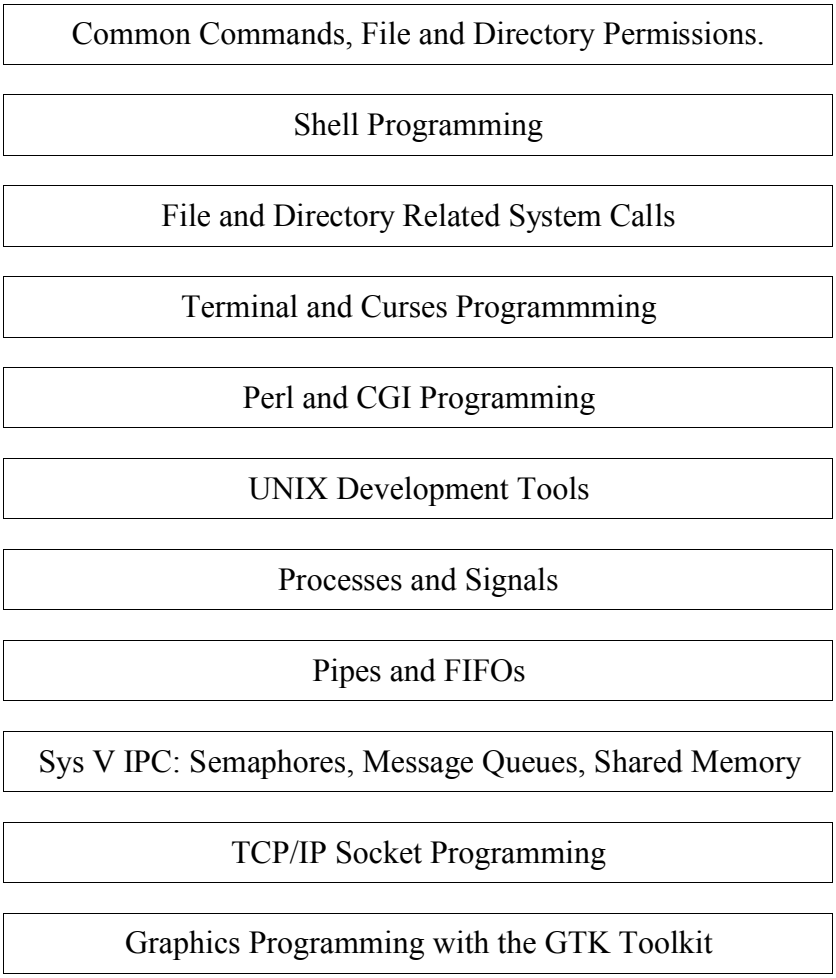

Figure 1: Outline of Course Topics

3.3 Files, Directories, and Terminals

Seven lecture periods were used to discuss several basic system level routines. Initially we discussed files, file descriptors and the associated routines (open, close, read, write, lseek, dup, stat) and also the routines for file maintenance (chmod, chown, link, unlink). We then moved on to a description of the system level routines that are used to traverse directories in the filesystem.

Next, other non-file related system routines were described. This included discussion of program argument processing, environment access, time and date routines, and routines for obtaining user and host information.

Two periods were devoted to coverage of terminal control routines and the curses library. Finally, a single lecture period was used to discuss file locking and the dbm database routines.

For the second programming assignment students wrote equivalents to the standard UNIX *ls* and *cat* programs. A third assignment required students to implement the Mastermind game using the curses library.

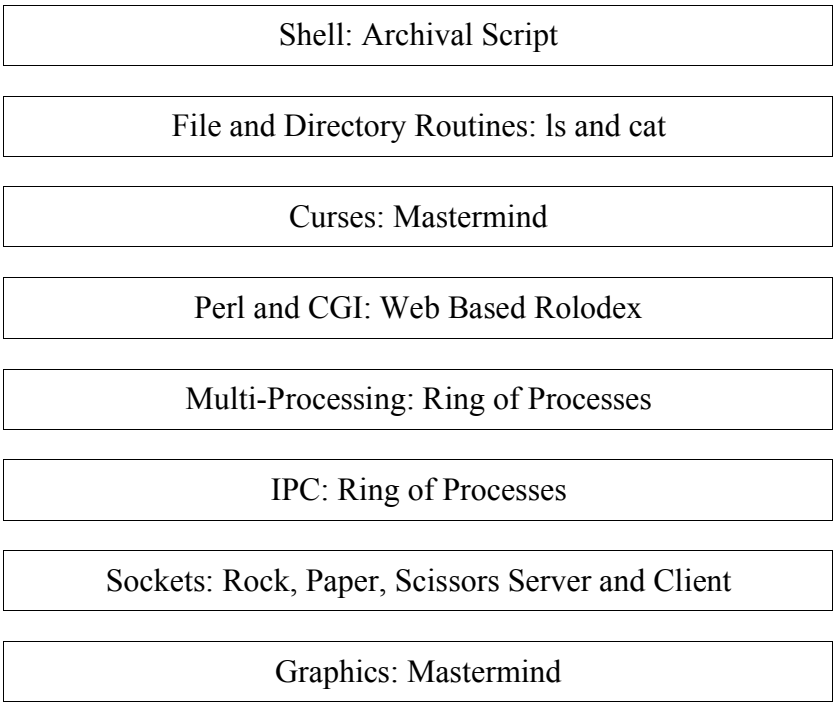

Figure 2: List of Programming Assignments

3.4 Perl, HTML, and CGI Programming

Two lectures were used to discuss the Perl scripting language. Since a lot of CGI (Common Gateway Interface) programs are written in Perl two lectures were used to cover HTML and CGI programming. This gave the students enough background to complete a CGI programming assignment in Perl.

The fourth assignment had the students implement a web based rolodex. The front-end of the rolodex had to be written in Perl while the backend database programs were written in C++ and required that the students use the dbm database routines.

3.5 UNIX Tools for Software Development

In this segment of the course we discussed some of the standard UNIX software development tools: make, source code control (via CVS), and the debugger.

3.6 Processes, Signals, and Pipes

Over the course of four periods we covered the following: (1) how to create processes using the fork and exec routines, (2) signals (software interrupts) and signal handling, and (3) interprocess communication using pipes and FIFOs.

Students were asked to write a program that displayed either the first 40 digits of pi or e depending on the signal received. They also had to write three programs that passed data around in a ring. The first program would read input from the user and pass it to the second program. The second program would convert the text to uppercase and pass it to the third program. The third program would reverse a line of text and send the data back to the first program which would then display it on the screen.

3.7 System V Interprocess Communication

Three lectures were devoted to coverage of System V IPC methods: semaphores, message queues and shared memory. As a programming assignment the students were asked to reimplement the previous ring-of-processes assigment by using either a single message queue or a shared memory segment for communcation.

3.8 Network Programming

Five periods were devoted to network programming. After presenting an overview of TCP/IP networks we moved on to discuss network programming via sockets. Students were shown how to write network clients and servers. Methods by which a server could handle multiple clients were described. The standard routines for name and address format conversion were listed.

The programming assignment was to write a network server and client for the rock, paper, scissors game. Two players, on different computers, would run the client program. The clients would communicate over the network to a server on a third computer. The server would decide the winner in each round and keep game statistics.

3.9 Graphics Programming

In the final portion of the course students were introduced to X windows programming using a high-level toolkit. Although there are several high-level toolkits available I choose to use the GTK toolkit because it is one of the more popular toolkits and it is also discussed in the textbook that was selected for the course.

The corresponding programming assignment was to reimplement Mastermind using the GTK toolkit.

4 Textbook

The textbook for the course was the second edition of *Beginning Linux Programming* by Stones and Matthew and published by Wrox Press[2]. The third edition of this book is now available.

Valuable references for the instructor and students are *Advanced Programming in the UNIX Environment* by Stevens[3] and Volumes 1 and 2 of *UNIX Network Programming* also by Stevens[4,5].

Other suitable choices for textbooks are *UNIX Systems Programming* by Robbins and Robbins[6] and *Interprocess Communications in LINUX* by Gray[7].

5 Required Computer Resources and Configuration

One of the more challenging aspects of the course was setting up a UNIX working environment for the students. The computers in the computer science lab can dual boot Linux and Windows XP, but many of the students wanted to be able to work on programming assignments in other computer labs or at home. The lectures were in a computer instructional lab, but the computers in this room only ran Windows XP. I wanted to be able to demonstrate UNIX programming concepts on the instructor's station and I also wanted to make a UNIX programming environment available to students for in-class exercises. In this section specific recommendations and instructions regarding hardware, software and classroom setup are provided.

5.1 Required Hardware

Students taking the course on-line need a Pentium class computer running either a modern version of Windows or Linux. Students could also use a computer in any of the on-campus labs.

Although not an absolute necessity, for a few of the topics and corresponding assignments, it is convenient to have access to a Linux server. All students in the course should be given accounts on the server. It is not necessary to use a server dedicated to the course. The existing computer science file and printer server is used at UE. This is a Pentium 4 running at 2.66 Ghz with 512 MB of RAM and a 100 GB ATA hard drive. This was a much more powerful machine than was necessary for this course.

5.2 Required Software

Although the various Linux distributions can be installed along side another operating system (a dual-boot configuration) it was felt that the installation procedure would be too difficult for most of the students enrolled in the course. A dual-boot setup was listed as an option for the very daring or for those who had friends with UNIX experience. It was emphasized that students would not be required to install UNIX on their computers in order to successfully complete the course.

Students taking the course on-line were advised to install the Cygwin UNIX emulation tools. Students working on campus had the option of using the computer science lab or of using any Windows based computer connected to the campus network. The Linux server had to be specially configured to enable connections from the campus network.

The Cygwin tools and server configuration are described in further detail in the following sections.

a) The Cygwin UNIX Emulation Tools

The Cygwin tools are ports of the GNU development tools and utilities to Windows. (The GNU tools are UNIX-like, but are free from UNIX license restrictions.) Cygwin also provides a UNIX emulation library that is amazingly complete. UNIX I/O, file and directory, process management, System V IPC, and networking (sockets) routines are available. The emulation library makes it easy to port UNIX applications to Windows. More importantly for this course, it allows students to do UNIX development on a Windows OS. Also included are standard UNIX shells (bash, tcsh, pdksh, zsh) and scripting languages (Perl, Python, Tcl/Tk). In addition to the C and C++ compilers required for most course assignments, compilers and/or interpreters are included for Ada, Java, Pascal, Fortran, Common Lisp, and Scheme. An X server and the standard X window applications are also included.

Students taking the course on-line were encouraged to install the Cygwin tools on their personal computers. The Cygwin tools were also installed on a network file server on the campus network. This allowed students to run the tools from any computer on the campus network and did not require installation of software on any computer except the campus server.

The Cygwin tools are completely free and may be downloaded from the cygwin.com web site[8]. To make installation easier I made a CD-ROM disk containing the cygwin tools available to students. A complete installation of all tools requires approximately 800 MB of disk space.

b) Setting Up The UNIX Server for Remote Access

Course topics and assignments could easily be selected so that all course work could be done using only the Cygwin tools. For the following topics, however it is easier to do development on a UNIX server:

- *System V IPC* - The Cygwin System V IPC emulation requires running a special Windows service. Although this is relatively easy to do, explaining how to do it and assisting students in getting it set up properly takes time away from other course topics.

- *CGI Programming* - The Perl programming assignment requires the students to write a Common Gateway Interface (CGI) program. These programs must be run on a web server. Explaining how to set up a web server on the student's computer and configure it for CGI was beyond the scope of this course.

- *GTK Programming* - Finally, the graphical toolkit selected for the GUI assignment (the GTK toolkit) is not included as a standard component of the Cygwin software. (The GTK toolkit has been ported to Cygwin, but is not yet part of the standard distribution. The GTK port could be downloaded and installed on the network server for students who want to use Cygwin. Alternatively, one could consider using the cross-platform FLTK toolkit which is included in the standard Cygwin distribution.)

These problems were easily solved by giving all students in the course an account on a UNIX server. They can log onto the server remotely to complete all assignments.

All of the assignments except the GTK assignment can be done using only a command-line interface to the server. Students can get a command-line interface (CLI) by logging in via either Telnet or Secure Shell (ssh). A Telnet client is included with the Microsoft Windows OS. A popular (and free) ssh client for Windows is Putty. The Cygwin suite also includes a ssh client. There are several options for working on the server via a GUI. Either Virtual Network Computing (VNC) or the X Display Manager Control Protocol (XDMCP) can be used to run a complete modern desktop environment (GNOME or KDE) on the server while displaying all graphics on the local computer. The VNC software is freely available[9]. The viewer software does not require installation on the local (Windows) computer. The user only needs to run the viewer application, by placing the viewer on a network server, the viewer can be run on any computer connected to the campus network. XDMCP requires X server software on the local computer, buts puts less of a load on the server. An X server is included with the Cygwin tools so, by installing Cygwin on a network server, this method can also be used from any machine on the campus network. Slow network connections or the presence of a firewall can make both of these methods frustrating for off-campus users.

The Cygwin tools include an X server and several window managers. So another way to work in a GUI is to run an X server and window manager on the local machine. Applications can be run on the server with results displayed on the local computer. This was the recommended method for obtaining a GUI connection because it puts the lightest load on the server. Unfortunately, the default set of Cygwin tools do not currently include either the GNOME or KDE desktop environments.

5.3 Classroom Setup

The course was taught in one of our computer instruction classrooms. There is computer for each pair of students. There is also a instructor computer station which is connected to a projection system. The computers all ran Microsoft Windows XP. In-class exercises could be completed using either the network installation of the Cygwin tools or by obtaining either a CLI or GUI connection to the server. No software needed to be installed on the classroom computers.

All lecture presentations were developed on a Linux system using the OpenOffice.org office suite. To display the presentation a Cygwin X server was run under Windows XP on the instructor's station. A network connection to the server was established and OpenOffice.org was

run on the server while displaying all results on the X server on the instructor's computer. Figure 3 shows the OpenOffice.org presentation program (Impress) running on the server, while the result is shown in an X server window that is running on the local Microsoft Windows XP machine. In this example, a connection to the server was created using XDMCP. The example shows the KDE desktop environment running on the server. Normally the X server was run in full-screen mode so that the local windows machine looked exactly like a UNIX system. The presentation software was then run in full-screen mode during the presentation. The figure shows the presentation software and the X server running in windowed mode to show the underlying KDE and Windows desktops.

The OpenOffice.org software allows the user to easily export the lecture as a web-based presentation. The web presentations were then uploaded to a web server for use by students taking the course on-line. Many of the students in the classroom preferred to follow the lecture by looking at the web pages rather than the image from the projection system.

During the lecture presentation it is convenient to also have a terminal window open on the server, so that the instructor can easily jump out of the presentation to demonstrate something at a command prompt.

Figure 3: Instructor Station showing the Cygwin X Server and OpenOffice.org running on Windows XP

6 Summary

A "UNIX System Programming" course was developed in reponse to the increasing need for practicing Computer Engineers to be familiar with the UNIX environment and development tools. With relatively few required resources it is possible to create a convenient UNIX programming environment for students on a campus where few UNIX systems are available. Lists of topics and assignments for such a course are presented. Complete courseware resources are available through the ACM SIGCSE web site[1].

7 References

1. ACM Special Interest Group in Computer Science Education (SIGCSE) Resources Web Site, Complete courseware for the course described here may be found through this site., http://sigcse.org/topics/
2. Stones, Richard and Matthew, Neil, *Beginning Linux Programming*, WROX Press, © 1999
3. Stevens, W. Richard, *Advanced Programming in the UNIX Environment*, Addison-Wesley, © 1992
4. Stevens, W. Richard, *UNIX Network Programming, Volume 1 - Networking APIs: Sockets and XTI, Second Edition*, Prentice-Hall, © 1998
5. Stevens, W. Richard, *UNIX Network Programming, Volume 2 - Interprocess Communications, Second Edition*, Prentice-Hall, © 1999
6. Robbins, Kay and Robbins, Steven, *UNIX Systems Programming*, Prentice-Hall, © 2003
7. Gray, John S., *Interprocess Communcations in Linux*, Prentice-Hall, © 2003
8. Cygwin Web Site, This site contains the Cygwin tools. The Cygwin tools provide a UNIX environment for Windows., http://cygwin.com/
9. Virtual Network Computing (VNC) Web Site, VNC is remote control software that allow you to view and interact with another computer on a network., http://www.realvnc.com/

8 Author Biography

ANTHONY M. RICHARDSON obtained B.S., M.S. and Ph.D degrees in Electrical Engineering from the University of Kentucky, Syracuse University and Duke University respectively. He is an Assistant Professor in the Electrical Engineering department at the University of Evansville.