

CS 23: UNIX SYSTEMS PROGRAMMING

WILLIAM EMMANUEL S. YU

ABSTRACT. The emergence of Open source Unix-like operating systems such as Linux and BSDs present an attractive option to system integrators due to their low cost and relative stability on production environments as compared to traditional Unix and Windows platforms.

There is a growing need for computer science graduates who are able to deploy, configure and develop on such systems. The course will contain some basic Unix system fundamentals such as installation and scripting to familiarize the student with the environment. It will contain lessons on ANSI C and Unix System and Network programming as well as X/Motif Programming. The course aims to expose students to Unix systems development in order to give our graduates a wider field of expertise.

1. INTRODUCTION

The emergence of Linux as a commercially viable alternative to traditional Unix and Windows operating systems has provided the computing community a cost-effective, stable and flexible environment for systems development. Linux, being similar to the traditional Unices, is an ideal platform for teaching systems programming.

Linux is freely available. There is no need to the school or the students to have to purchase the software needed to teach this course. A student need not purchase expensive software and books. All the required reference material and documentation can be found in the Linux distribution.

Linux is an open source operating system. The student is able to look into the actual source code. It is ideal for learning the basics and intricacies of operating systems.

Linux is a standards compliant operating system. It ANSI and IEEE-POSIX compliant. It uses the GNU C libraries which is becoming a standard across a good number of Unix-based systems. Software development learned while developing Linux applications can be easily ported to other Unix-based platforms.

A good number of traditional Unix vendors are standardizing on Linux. Most of them, such as SCO, SUN, IBM and others, ensure binary compatibility with Linux applications. A good number of major software projects are also developed on Linux. To name a few, they are Apache(the world's most popular webserver), Squid(the most popular proxy server to Unix environments), Bind(a popular DNS server) and many others. These and many other reasons make Linux an ideal operating system for teaching software development and systems programming.

2. GRADING SYSTEM

The course will be a project-based course. It will have no major written requirement. The students grades' will be divided into two components. The first component will be composed of the five major requirements for each section of the course which will compose seventy(70) percent of the final grade. The final integrative project will comprise the remaining thirty(30) percent. The instructor will have the prerogative to give bonus projects and quizzes.

Letter Grade	Numerical Equivalent
A	93%
B+	87%
B	81%
C+	75%
C	69%
D	60%
F	below 60%

3. COURSE COMPOSITION

3.1. Unix Basics.

Duration: 3 weeks

Goal: To be enable the student to install and configure a Linux system. The student must also be able to familiarize his/herself with basic Unix commands and tools.

Major Requirement: Shell Script

Coverage:

- * System Basics
- * Basic Unix Filesystem
- * Using Linux/Unix
- * Networking Tools
- * Basic Shell Scripting (BASH)

3.2. C Programming.

Duration: 3 weeks

Goal: To enable the student to learn programming ANSI C in the Unix Environment. The student must also be able to familiarize his/herself with the available Unix tools for C programming.

Major Requirement: ANSI C Program Distribution

Coverage:

- * Unix Tools for C Programming
- * ANSI C Fundamentals
- * Dynamic Memory in C

3.3. Unix Systems Programming.

Duration: 3 weeks

Goal: To enable the student to use his/her C programming skills in systems programming in the Unix environment. The student must be able to familiarize himself with the available Unix system calls.

Coverage:

- * File I/O and Filesystem Calls
- * Standard I/O
- * System Data Files and Information
- * Unix Process Environment
- * Process Control
- * Standard Relationships
- * Signals

3.4. Unix Network Programming.

Duration: 3 weeks

Goal: To enable the student to use his C programming skills in networking programming in the Unix environment. The must be able to familiarize himself with basic networking concepts and be able to create simple client-server applications.

Major Requirement: Unix Client-Server Program

Coverage:

- * Network Primer
- * Daemon Processes
- * Interprocess Communication (IPC)
- * BSD Sockets

3.5. Unix Graphical User Interface Toolkits.

Duration: 2 weeks

Goal: To enable the student to familiarize himself with X Windows programming by using a GUI toolkit to aid in his/her interface development.

Major Requirement: Integrative Project

Coverage:

- ★ X Windows Systems
- ★ Programming with the X/Motif Toolkit

4. ADVANCED/ACCELERATED TRACK

The students will be given an option to take an advanced track through this course. This is intended for students who already have competent skill in systems and network programming. This is also intended for students who would rather perform research oriented work. During the first week of classes, the students will be asked to submit a project proposal for the said software project. The minimum requirements for the project are:

- (1) The application must have worldly significance or scientific applications for research and development.
- (2) The software must have a client-server architecture and must be accessible over a TCP/IP network.
- (3) The server software must catch the appropriate signals.
 - ★ SIGTERM, SIGQUIT must exit the server gracefully
 - ★ SIGHUP must either restart the server or reload its configuration file
- (4) The client software must have a graphical user interface that runs on an X windows system. Allowable graphics toolkits are QT, GTK and motif.
- (5) Multiple source files must be used. There should be at least a separate server and client source files.
- (6) The software must come with the appropriate build system. The instructor need only to type 'make' to compile the source code and 'make test' to run a test program.
- (7) All source code files must be properly commented.
- (8) The software must be packaged in a typical software distribution with matching makefiles, man pages, documentation, source files and others.

The proposals are subject to the approval of the instructor. If the instructor does not approve of the project another one must be selected or the regular track be taken. In the case of the advanced track, grading is solely based on the result of this project.

5. CONSULATION HOURS

By Appointment F207
 email address: wyu@ateneo.edu

DEPARTMENT OF INFORMATION SYSTEMS AND COMPUTER SCIENCE, ATENEO DE MANILA UNIVERSITY, LOYOLA HEIGHTS,
 QUEZON CITY, 1108 PHILIPPINES
E-mail address: wyu@ateneo.edu