

Introduction to C and Unix/Linux

Computer Science 232

Spring semester 2008

David W. Erbach
125D ETCS Hall

online course information at ...
kitani.ipfw.edu/~erbach/

481-6867
erbachD@ipfw.edu

prerequisite: CS 161 – Introduction to Computer Science II

textbook: *C and Unix, Tools for Software Design*
Martin L. Barrett & Clifford H. Wagner
John Wiley & Sons
ISBN 0-471-30927-3 (paperback)

When the Department's CS degree was first accredited a few years ago, we were criticized for not providing students much chance to experience other operating systems than Windows and other languages than Java. This course is part of our effort to broaden your view.

C and Unix have long been closely associated. Unix was the first operating system written in a high-level language instead of assembler, and that language was C. This had the effect of making the operating system largely independent of the computer on which it was running, an idea which seems ordinary now, but was radical at the time. Initially, Bell Labs distributed the operating system freely, because the Bell System wasn't a computer company. Their best-known partner then was UC Berkeley. When the Bell System was split up, Bell tried to commercialize Unix. Bell's version and its descendants became known as "System V" and Berkeley's as "BSD." Each has many variants. Then Linus Thorvalds came along, and wrote an alternative kernel which played well with the Unix toolset. This came to be known as "Linux." Linux has itself split into many versions. Among the best known are Redhat, Debian, and Ubuntu.

In this course, we'll try to give you a good familiarity with the Unix/Linux toolset, and particularly the C language and GNU compilers. If we have time, we'll do a little Python. Side themes include learning how to work with a CLI-based server, and how to design and develop software on a cooperative and slightly larger scale than is possible in introductory courses.

To be more specific, the primary course objectives are:

- To gain fluency in the Linux/Unix operating system environment and standard shell tools.
- To gain fluency in the C programming language.
- To gain fluency in a shell scripting language.

Since CS161 is a prerequisite, I will assume you know the generalities of how to program already, and especially how to look up information you may not know. That will permit us to move relatively

quickly through the standard syntactic issues (program structure, how loops and branches work, etc.), and concentrate on issues which characterize C. These include the frequent use of pointers and indirect references, the libraries, and C's ability to interact with the operating system and shells.

On the Linux/unix side, we will assume less, since you may be less familiar with the unix context than with Windows. After an introduction to the Unix environment, we will begin with the basic commands and their options and move steadily through to more sophisticated tools for analysis and scripting, like `grep` and `sed`.

Here are the grading guidelines for the course.

grading:	homework assignments	100 points	
	semester group project	100 points	
	in-class discussions	20 points	
	2 x mid-term examinations	80 points	(each)
	final examination	120 points	
	total	500 points	

My regular office hours are MWF 9:00 - 10:00. Generally, I am easy to find, except perhaps on Thursdays, which I shall try to keep for my own professional development.

ABET Program Learning Outcomes

The following learning outcomes are defined by ABET for computer science programs.

- (a) An ability to apply knowledge of computing and mathematics appropriate to the discipline
- (b) An ability to analyze a problem, and identify and define the computing requirements appropriate to its solution
- (c) An ability to design, implement, and evaluate a computer-based system, process, component, or program to meet desired needs
- (d) An ability to function effectively on teams to accomplish a common goal
- (e) An understanding of professional, ethical, legal, security and social issues and responsibilities
- (f) An ability to communicate effectively with a range of audiences
- (g) An ability to analyze the local and global impact of computing on individuals, organizations, and society
- (h) Recognition of the need for and an ability to engage in continuing professional development
- (i) An ability to use current techniques, skills, and tools necessary for computing practice.
- (j) An ability to apply mathematical foundations, algorithmic principles, and computer science theory in the modeling and design of computer-based systems in a way that demonstrates comprehension of the tradeoffs involved in design choices.
- (k) An ability to apply design and development principles in the construction of software systems of varying complexity.

The formal objectives of this course listed above address these outcomes as follows:

- 1. (a) (i)
- 2. (a) (i)
- 3. (a) (c) (i)

References

C and Unix are both long-established, and Linux has an increasing fan base. If you are interested in specifics, kitani is running the most recent stable server version of ubuntu.

Apart from the textbook, there are many suitable books and on-line resources you can use for reference. As always, the best idea is to go to the library or any big bookstore and browse until you find something which suits your style and level of knowledge. That said, here are a few books I like:

Ubuntu and Linux culture

Just for Fun: the Story of an Accidental Revolutionary, Linux Torvalds with David Diamond, P (paperback version about \$15 new and \$3 used on Amazon)

This light-hearted book is the closest anyone has been able to come to getting Linus himself to talk about the history and earliest development of Linux some 15 years ago. (He quotes his mother asking how he would ever “meet a nice girl” if he stays at home programming all the time. But he did anyway.)

www.ubuntu.com

This is the main support portal of the ubuntu community. Ubuntu is a Debian derivative, and is entirely supported by volunteers. It is considered to be one of the friendliest and most open and helpful Linux communities.

C and Unix

The C Programming Language, 2nd ed., Brian W. Kernighan & Dennis M. Richie, Prentice-Hall, ISBN 0-13-110362-8 (paperback version)

This slim book is the classic by the people at Bell Labs who designed C in the first place. It's not always easy to read. (Its reputation is that everything is said once and only once, like a mathematics book.) But when you have read the book thoroughly enough to understand the example code, you can feel confident that you know C well.

The C Companion, Allen I. Holub, Prentice-Hall, ISBN 0-13-109786

This book shows at a lower level how C constructs actually work, to help you understand what goes on during the compilation process. C being what it is, that knowledge can be very handy in trying to understand and debug subtle problems.

Understanding Unix, 2nd ed. 1994, Stan Kelly-Bootle, Sybex, ISBN 0-7821-1499-7.

Our text sometimes skimps a little on the Unix side. This is a good supplement by one of the famous names of Unix (How could a name like “Kelly-Bootle” not be famous, you ask?) Though it is a larger book than the others on this list, it is my favorite introductory book about Unix tools.

Programming technique

These two books don't have direct technical information necessary in CS 232. But they are the kinds of information you should be familiar with as you transform yourself into a professional.

The Practice of Programming, Brian W. Kernighan & Rob Pike, Addison-Wesley, ISBN 0-201-61586

From football to music to programming, there are amateurs and then there are the professionals. If your goal is to become a programming pro, this book, which emphasizes the themes of "simplicity, clarity, generality," is a readable and agreeable way to move along in your development.

Programming Pearls, Jon Bentley, Addison-Wesley, ISBN 0-201-65788-0

This is another classic about how to program well, placed in typical contexts, from issues of provable correctness, to standard tasks like updating tree data structures.

Tools and related issues

Learning the vi Editor, 5th ed., L. Lamb, O'Reilly & Associates, Inc., 1990, ISBN 0-937175-67-6

This is the best general guide to the `vi` editor, one of the standard C editing environments. Linux normally invokes the extension `vim`. It's a particularly good source of advanced technique, like how to customize the appearance and behavior of `vi`, depending on what you are doing; perhaps document preparation in one directory, and source code development in another.

`emacs` is the other standard environment. You should pick one of `vi` and `emacs` and learn it well.

Python

Yes, it was named after Monty ...

www.python.com

In Python's case, the guru is Guido van Rossum. The language runs in most environments and is increasingly widely used for its ease of use, extensibility, and strong open-source support community. See, for example, some of the "success stories" listed at the site, including people like Firaxis, developers of *Civ IV*. Of course, there is also lots of technical help to be had here.

en.wikipedia.org/wiki/Python_language

This is python's wikipedia site. It has a lot of background about the language and how to use it.

Programming Python, Mark Lutz, O'Reilly & Associates, ISBN 1-56592-197-6

This is not a slim book. Insead, it's a very gentle introduction to Python, which in its 800 pages, has time to get quite a ways. Near the end it says, "And even more cool stuff. It's impossible to mention all the exciting work being done ..."