Tele 2015: Computer Science Concepts

C Programming, Unix, C++, and the X Window System
Spring 1993(93-2)

Michael B. Spring
Department of Information Science
University of Pittsburgh
spring+@pitt.edu
Introduction

Telecommunication 2015 introduces C and Unix in the context of telecommunications and information system problem solving. The C language and the Unix operating system have been growing in importance since their development in the early 1970’s. Recently, Unix has served as the basis for much of the thinking about the Portable Operating System Interface -- POSIX. Similarly, C has become the base language for C++ and has been the first language for the development of several important libraries such as the X Window System. Because of the opportunities presented, more and more people have been moving to C and Unix for project development over the last several years. This course will offer an overview of C and Unix, and, as time permits, will introduce the X Window System and C++ extensions to C.

The importance of C and Unix

The development of many different kinds of systems is taking place more and more on workstations. These workstations run the Unix operating system and programming is done in the C language. In particular, network based interactive programs are being designed to run on these workstations taking advantage of the ability to control high resolution bit mapped screens connected to a distributed network of processors and servers. This course will focus on assisting you in the development of the basic skills needed in these types of environments. The programming work in this course can be done on PC's, workstations, or mainframes. The Unix work will require that you work on the SUN workstations located on the eighth floor of the LIS building, or on the universities unixd platform. However, this work can be done from a remote PC logged in as a terminal. The one exception to this will be the final program that demonstrates simple X Window System programming. This program will have to be executed from an X Window System Server -- in our case, a Sun workstation.

As you are probably aware, our computer environment is changing. The programs written for 24-hour bank machines are good examples of network based interactive programs. They are considered good interactive system programs because almost anyone can use them without reference to additional materials. The program is self explanatory. The quality of the programs as examples of network based software is also high. They maintain reliable communications links to mainframe systems that allow them to process up-to-date information about your accounts and activities.

In terms of software, we are less and less concerned with processing algorithms and more and more concerned with processing tools and human-computer interfaces. Take for example the fact that in text processing we have evolved from line editors and printers, through screen editors, to page layout and document processing software and laser printers. You must not only know the sorting, searching, pattern matching algorithms, you must also know what screen colors will be best, how to form icons, what command words the user will understand, what likely mistakes the user will make, etc.

We are also more concerned today than in the past with standards. For example, significant progress is being made toward a Portable Operating System Interface (POSIX). That interface is modeled to a significant degree on Unix. At the same time, networking standards such as TCP/IP and ISO’s Open Systems
Interconnection are finally reaching significant maturity and levels of saturation. Every day, a new standard is being introduced which will impact how you program. You will need to learn and use the standards in your programming.

Conceptually, there is ongoing pressure to move to higher and higher levels of meaning in the analysis of information and communication. This is perhaps best illustrated by the use of CASE and CAD/CAM/CIM tools. While programming always involves working at the lexical levels, it should be your goal to raise your programming standards to include syntactic as well as semantic aspects of information processing.

**Instructional Design of the Course**

As in any graduate course, in 2015, students have responsibility to be proactive in their learning. The instructor’s role is less directive and more one of stimulating and guiding learning. If you think back to the things you remember from other courses you have taken, you will probably find that the things you remember best are the things you had to work hard to learn. It is simply true that the things you remember are the things you work hard to learn. For that reason, the instructor and students share the responsibility to make the course "work".

Generally speaking, programming languages and specific operating systems are not taught at the graduate level. Because so many students enter the program with little or no knowledge of C and Unix, an exception to this rule has been made to provide students with an opportunity to come to grips with the C language and the Unix operating system in the context of important information science and telecommunications programming concepts.

Keep in mind that this course takes an accelerated look at C and Unix. Indeed, every effort will be made to cover not only C and Unix, but a little about the X Window System and C++. Students will need therefore to stay abreast of the requirements in the course so as not to get lost.

You are cautioned that falling behind in the course will greatly reduce your ability to understand the lecture content, making it still more difficult for you to keep up with the course.

C and Unix are each rich and complex. Both work best when the user has a sound theoretical understanding and long years of experience using them. The power of the tools comes as much from having used them to do something as it does from understanding the theory of what’s possible. This is equally true for the X Window System.

**A Few Words about Assignments and Requirements**

It is your responsibility to come to class prepared to ask questions and explore problems related to the lecture topic. If you are not prepared, the concepts discussed and the nuances of the language will not be seen. You will be at a significant disadvantage if you come to class with simply a clear head!
The reading assignments for each class are to be done before class. The lecture will reflect on these. The lectures will not provide notes to review after class to make the readings more intelligible.

The programming assignments will be as meaningful as possible within the constraints of a course. In the real world, software systems are extremely complex and consist of thousands to hundreds of thousands of lines of code. You should anticipate many hours of work on some of your assignments to make them work. If you take on a real problem, the code you develop will be extensive.

The final programming assignment, using the X Window System, is the only "real" assignment that will be possible in this class. To make it possible for you to work on this more complex assignments, you may work on it in a teams. This means you will need to coordinate your efforts. Grading will reflect group effort. That is, more will be expected of projects completed in groups. Because of the energy required to coordinate groups, it is recognized that the quality of a product of a group effort will be somewhat less than a multiple of the number of people involved. At the same time, it should be more complete and sophisticated than an effort completed by an individual. If you work in a group, it is very important that you contribute in a timely fashion and be cooperative about meetings.

Students working in groups who "take a vacation" and then try take credit for the work of your teammates will not be tolerated. They will receive a zero for the assignment.
Goals, Requirements, and Grading

Course Goals for IS 2015

The goals of the course are as follows:
- To learn and apply the C language to programming of communications and interactive systems.
- To learn the basics of the Unix operating system with attention to the array of tools generally available in Unix.
- To learn some simple algorithms for such things as data compression, error checking, screen display, event control, device drivers, etc.
- To learn the basic structure of windowing systems with particular attention to the rudimentary commands in the X Window System.
- To understand the nature of object oriented design and programming as manifest in the evolution of C++.

Course Requirements

This course is a skills course. To be successful, you must exercise a concerted effort in the course. For this reason, the grading scheme is constructed to encourage you to keep on the schedule. It is as follows.
- Assignments submitted before the due date will be graded on a basis of 110% of the indicated value. Thus for a 10 point project, the maximum award will be 11 points if submitted the week before the due date.
- Assignments submitted after the date they are due will be graded on the basis of a 10% maximum point award reduction per week. Thus an assignment worth 10 points and due in week 3 will be graded on the basis of a maximum award of 9 points if received in week 4, 8 points in week 5, etc.
- The instructor will adjust the schedule as needed if the lectures fall off the announced schedule to allow time for you to work on the various assignments after the related issues have been covered in class.

Students who find themselves in situations where they will not be able to work on schedule for business or personal reasons should petition the instructor in advance and in writing providing: 1) a reason for the special schedule and, 2) a copy of the proposed schedule. This personal schedule will become binding on the student with the same grading provisions as the general schedule for the class.

The course requirements include programming, use of Unix tools, and mini-quizzes. They are all described below.

Programming Assignments

There are a total of five programming assignments. The hard copy of the source code for each assignment will be submitted to the instructor along with a disk for
PC based code and a electronic mail copy of Unix based programs. The programs are:

- (0-15) A keyboard filter that reads characters from the keyboard and keeps track of what was typed. The screen will be divided into two halves -- top and bottom. Across the top, you will show the printable characters types by the user. Across the bottom, you will show all letters converted to upper case, all digits as their word equivalent, all function keys as their function key name, and all control characters as their ascii control character name -- e.g. control Z is known as EOF.

- (0-15 points) A driver program that control a screen in none of the following ways:
  - a turboc program to draw on a pc graphics screen using a mouse. The colors available, the width of the stroke, and the ability to quit should all be mouse selectable.
  - a program to interpret a text file containing Hex 81h(bold), 82h(reverse video), 83h(underline), 84h(center), 85h(flush right) and 80h(end special case) and display it on a standard VT100 screen one page at a time with key control for forward page, back page, and end.

- (0-15) A file display/dump program that uses low level block I/O with array processing. The program will take any file and display it one of two ways: in hex, 16 characters per line, with the "ascii" as well as the hex representation; in byte display mode, displaying control characters as the bold form of the character and those above hex 81 as reverse video of either the ASCII character set or the extended ascii character set.

- (0-15 points) A program that reads a file and constructs an array of all the digrams in the file, sorts and finds the 128 most frequently occurring digrams, and uses that list to encode the file compressing the 128 most frequently occurring digrams as the 128 extended(high order bit on) ascii codes. The frequently occurring digram list, in order, should be prepended to the compressed file as the first 256 characters.

- (0-40 points) Write a small interactive program using the X Window System to provide access to information in a way similar to that provided by inf on icarus. Your program should be windowed and will have at least two display areas. The project, particularly if it incorporates the optional window may be done in groups of 2-3 students. The areas for the system include:
  - The Command Area provides services to the user. At a minimum it must include a quit button and a menu that provides options for the display window listing the files the user has for obtaining information about. When an option is pointed at, that information is displayed. It might also include such functions as print screen, copy screen to buffer, shell mail program etc.
  - The Display Area provides a space in which the menu option selected is displayed. It should also include controls for paging forward and backward in the file.
  - (Optional) A Message Area could display message of the day messages on a recurring basis. The messages will come from the file "specialmessage.file" The file will contains a series of one line messages (max 80 characters). Each line begins with a two digit number that specifies the length of time to display the message in seconds.
Unix Tool Assignments:

There are five assignments related to Unix tools.

- (0-10 points) Determine the subjects of RFC 821 and 822 on the directory info/rfc on the machine ftp.pitt.edu, which is open to anonymous ftp. Access the same structure by telneting to unixd, and accessing directory /afs/pitt/public/info/rfc and determine what RFC contains the specification of the Transmission Control Protocol.

- (0-10 points) Provide a count BY FILE of the number of occurrences of the word "help" in all man pages on your standard Icarus man page directory or directories. The search must be case insensitive and the occurrence of help embedded in a word should not be counted. Show a printout of all your work as it occurs. (HINT use 'tee')

- (0-15 points) Obtain the files bullwnk, donald.duck, and station, which are gif images requiring binary transfer, from the anonymous ftp server wuarchive.wustl.edu. Tar the images into a single file, uuencode it, csplit it into 25K files, and mail them to the course account.

- (0-15 points) Write a shell script to remove identified backup files from your home directory structure -- including subdirectories, and to prompt you for the disposition of any files that have not been accessed in more than a year.

- (0-20 points) Prepare a three to five minute presentation, preferably via video tape, that shows one or two useful applications of one of the Unix commands from part 3, chapter 11 of Kochan and Woods. The goal of the presentation is to demonstrate how one of these commands can be used effectively to make better use of the Unix system. This project, particularly if done on video, may be done in groups of 2 or three students.

Exams

There will be 5 quizzes in the course. Each quiz will be worth 10 points. Some quizzes will consist of multiple choice, true/false, and short fill-in questions. Others will consist of a programming problem to be resolved in a small number of lines of code. Coding quizzes will be open book. Question quizzes will be closed book. Each quiz will be 15 minutes in length and will be scheduled for the beginning of class. Students are required to take only three quizzes. The last two may be used as extra credit.

Grading

Based on points earned out of total points possible:
- **A** 180 - 200pts
- **B** 150 - 179pts
- **C** 120 - 149pts
- **F** 0 - 120pts
Required and Supplementary Materials

Required and Recommended Materials

There are two required texts for the course:


In addition to this text, students may wish to purchase


Supplementary Materials

If you like to work on PC’s, you may want to consider some reference book on C, QuickC or TurboC. You will find a variety of different books in the Pitt bookstore. Pick up one that reads well to you. One that helps you better understand C. The Instructor’s personal preference is Schildt’s *Turbo C/C++ The Complete Reference*, Osborne McGraw-Hill.

Other books that might be of interest for various assignments include:

- Scheffler, Gettys and Newman, *X Window System*
- Sobell, *A Practical Guide to Unix System V*
- Young, *The X Window System, Programming Applications with Xt*
- Martin and Odell, *Object Oriented Analysis and Design*
- Spring and Dubin, *Hands on Postscript*
Course Schedule and Lecture Topics

Course Schedule

Classes will meet from 6:00 to 8:50pm in Room 533 LIS building on Monday of each week.

<table>
<thead>
<tr>
<th>Lecture</th>
<th>Date/Quiz</th>
<th>Assignments</th>
<th>Topic</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1/11</td>
<td>Review of Syllabus; Intro to C and Unix</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1/18</td>
<td>Martin Luther King Day</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1/18</td>
<td>Martin Luther King Day</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>2/1</td>
<td>Basic Syntax of C; Overview of Unix</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2/8</td>
<td>Flow Control; Basic Unix Commands</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>2/15</td>
<td>Data Types and Character Processing</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>2/22</td>
<td>;Regular Expression and File Manipulation</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>3/1</td>
<td>;Shell Scripts, Functions, and Aliases</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>3/15</td>
<td>Spring Break!!!!</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>3/22</td>
<td>Types, Bit Operations, and Pointers</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3/29</td>
<td>Arrays and Array Pointers</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>4/5</td>
<td>The X window System Libraries</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>4/12</td>
<td>Advanced Text Processing</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>4/19</td>
<td>C++</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>4/26</td>
<td>Student Presentations</td>
<td></td>
</tr>
</tbody>
</table>

Lecture Topics

At the time of this writing, it is not quite clear the speed with which we will be able to cover all the topics in the course. For that reason, the topics that follow are presented in terms of weeks. It is probable that we will fall off this schedule in some ways. The instructor will notify you of those changes as they occur.
Lecture Topic Outlines

Lecture 1: Introduction to C and Unix

Topics

- Course Overview and Requirements
- Rationale for the Course
  - The importance of C
    - The use of libraries
    - The emergence of C++
    - High-level low-level
  - The importance of Unix
    - Multi-platform OS
    - Relationship to POSIX
    - Tool based approach
- Review of the course syllabus
  - Course philosophy
  - Course requirements
  - Course
  - Introduction of the class, instructor, and graduate student assistant
  - Description of the PC & SUN programming environments
    - TurboC/C++
    - dbxtool

Readings

Kochan and Wood, Unix Desktop Guide to Tools, Chapter 11, SKIM Commands.
Kelley and Pohl, C by Dissection, Chapter 15
Kelley and Pohl, C by Dissection, Chapter 14 (Optional)

Lecture 2: Lexical Elements & Operators; Overview Of Unix

Topics

- Comments, keywords, identifiers, and constants
- String Constants
- Operators and Punctuators
  - Increment and decrement operators
  - Assignment Operators
- Style
  - Common Programming Errors
System Considerations

- Overview of Unix
- The History of Unix
- The Unix System Environment
  - The File System
  - The Process Control System
  - The Shell
- Issuing Commands
- Relationship of the System, the Shell, and Programs
- Shell Variables and Metacharacters
- File Modes

Readings

Kochan and Wood, Unix Desktop Guide to Tools, Chapter 1
Kelley and Pohl, C by Dissection, Chapters 1,2
Kelley and Pohl, C by Dissection, Skim Chapter 5

Lecture 3: Flow Of Control In C; Simple Unix Commands

Topics

- C
  - Relational, Equality, and Logical Operators
  - Compound & Empty Statements
  - The if and the if-else Statements
    - Nested Flow of Control
  - The while, for, and do Statements
  - The switch Statement
  - The ’,’ and conditional Operators
  - The goto, break, and continue Statements
  - Common Programming Errors
    - = and ==
    - logical operation errors and parentheses

- Simple Unix Commands
  - Directory Commands
  - File Management Commands
  - System Commands
  - Information Retrievers

Readings

Kochan and Wood, Unix Desktop Guide to Tools, Chapter 3
Kelley and Pohl, C by Dissection, Chapter 3
Lecture 4: Functions, Structured Programming and Input/output And Files

Topics

- Functions
  - Function definition and invocation
  - The return Statement
  - Function Prototypes
  - Invocation and Call-By-Value
- File I/O
  - The Output Function print()
  - The Input Function scanf()
  - The Functions sprintf() and sscanf()
  - The Functions fprintf() and fscanf()
  - Accessing Files
  - Using Temporary Files and Graceful Functions
  - Accessing a file randomly
  - Common Programming Errors

Readings

Kelley and Pohl, C by Dissection, Chapters 4, 14

Lecture 5: The Fundamental Data Types And Character Processing

Topics

- Declarations and Expressions
- The Data Type char
- The Data Type int
- The Integral Types short, long, and unsigned
- The Floating Types
- The Sizeof Operator
- Mathematical Functions
- Conversions and Casts
- The Data Type char
- The Use of getchar() and putchar()
- Character conversion macros
- Character arithmetic
- Common Programming Errors -- casting and implicit conversion
- System Considerations -- libraries

Readings

Kelley and Pohl, C by Dissection, Chapters 5, 6
Lecture 6: Regular Expressions and Text File Manipulation

Topics

- Regular Expressions
  - Where Are Regular Expressions Used?
  - Regular Expression Characters
  - Regular Expression Examples
  - Regular Expressions and Commands
- Examining Text Files
  - "Look But Don’t Touch" Commands
  - The grep Family

Readings

Kochan and Wood, Unix Desktop Guide to Tools, Chapters 2, 4

Lecture 7: Shell Scripts, Functions and Aliases

Topics

- Writing Shell Scripts
  - Running a Shell Script
  - Constitution of a Shell Script
  - Other Shell Constructs for Scripts
  - Korn Shell Arithmetic
  - Commands for Scripts
  - Pipes and Shell Scripts
  - Scripts with awk and/or sed
  - Making Scripts Friendlier
- Writing Aliases and Functions

Readings

Kochan and Wood, Unix Desktop Guide to Tools, Chapters 8, 10

Lecture 8: Enumeration, Typedef, Bitwise Operators, Functions, Pointers, And Storage Classes

Topics

- Typing and Bitwise Operators
  - Enumeration Types
  - The Use of typedef
  - Bitwise Operators and Expressions
  - Masks
• Pointers
  • Pointer Declaration and Assignment
  • Addressing and Dereferencing
  • Void Pointers
  • Call-by-Reference
• Additional Declaration Rules
  • Scope Rules
  • Storage Classes
  • Default Initialization
  • Function Declarations and Definitions
  • Style -- pointer declaration
  • Common Programming Errors -- pointer manipulation
• Miscellaneous File Commands
  • File Comparison
  • File Encoding, Archives, and Compression
  • File Information: file and strings
  • Summary

Readings
Kochan and Wood, Unix Desktop Guide to Tools, Chapter 9
Kelley and Pohl, C by Dissection, Chapters 7, 8

Lecture 9: Arrays, Strings, And Array Pointers

Topics
• Declaration and Initialization of Simple Arrays
• Arrays and pointers
• Pointer Arithmetic
• Passing Arrays to Functions
• Multidimensional Arrays
• Dynamic Memory Allocation
• The End-of-String Sentinel \0
• Initialization of Strings
• Using Pointers to Process a String
• main() and int argc, char ** argv
• String Handling Functions in the Standard Library

Readings
Kelley and Pohl, C by Dissection, Chapters 9, 10
Lecture 10: Recursion, Structures And Linked Lists

Topics

- Recursive problem solving
- The divide-and-conquer methodology
- Common Programming Errors -- infinite loops and system limits
- System Considerations -- memory space
- Declaring Structures
- Accessing a Member
- Structures and Functions, and Assignment
- Initialization of Structures
- The Use of typedef for structures
- Self-Referential Structures
- Linear Linked Lists
- List Operations
- Counting and Lookup
- Insertion and Deletion

Readings

Kelley and Pohl, C by Dissection, Chapters 12, 13

Lecture 11: The X Window System

Topics

- History and Overview of the X Window System
- Architecture of X
- Functions, Xtrinsics, and Widgets
- Display Control
- Event loops
- Opening a window
- Acting on an event

Lecture 12: Advanced Text Processing Commands

Topics

- Formatting and Working with Fields
  - Formatting Commands
  - Field-Oriented Commands
- Changing Characters and Strings in Files
Changing Characters with tr
The sed Command
Summary

The awk Language
Using awk
Summary

Readings

Kochan and Wood, *Unix Desktop Guide to Tools*, Chapters 5, 6, 7

Readings

Lecture 13: C++

Topics

- Non-object oriented extensions
  - Comments
  - Stream I/O
  - Scope resolution operator
  - Function prototypes
  - Function code inlining
  - Function arguments
- Object Oriented Features
  - Classes
  - Constructor and Destructor functions
  - Friend functions and operator overloading
  - Inheritance

Readings

Pohl, *C++ for C Programmers*, Chapter(s) All