22C:113 INTRODUCTION TO SYSTEM SOFTWARE
Syllabus

Instructor: Teodor Rus
Office: 201J MLH, Phone: 335-0742
Class hours: MWF 10:30–11:20am, 51 SH
Office hours: MWF 9:30–10:20:am, 201J MLH

Important dates:
Midterm Exam: Wednesday, 19 October 2011, 10:30am, 51 SH
Final Exam: Thursday 15 December 2011, 7:30am, 51 SH
Assignments: To be set at assignment offering time.
Projects: To be set at project offering time.

Note: This course is given by the College of Liberal Arts and Sciences. This means that class policies on matters such as requirements, grading, and sanctions for academic dishonesty are governed by the College of Liberal Arts and Sciences. Students wishing to add or drop this course after the official deadline must receive the approval of the Dean of the College of Liberal Arts and Sciences. Details of the University policy of cross enrollments may be found at:
`http://www.uiowa.edu/ provost/deos/crossenroll.doc`

Academic Honesty:
The College of Liberal Arts and Sciences expects all students to do their own work, as stated in the CLAS Code of Academic Honesty. Instructors fail any assignment that shows evidence of plagiarism or other forms of cheating, also reporting the student's name to the College. A student reported to College for cheating is placed on disciplinary probation; a student reported twice is suspended or expelled.

Procedures for Students with disability:
I must hear from anyone who has a disability which may require some modification of seating, testing, or other class requirements so that appropriate arrangements may be made.

Please see me after class or during my office hours.

1 Purpose

The purpose of the course "22C:113 Introduction to System Software" is to introduce the student to the collection of programs and documents which constitute the system software of a computer platform. This introduction will allow the student to acknowledge the main objectives, problems, and programming techniques faced by a system programmer designing and implementing system software. Therefore, in this offering of 22C:113 the emphasis will be on concepts and foundations.

Emphasizing concepts and foundations we offer to the student a conceptual framework in which the system software is developed and used rather than an enumeration of programs which belong to the system software running on a given computer platform. This conceptual framework will show the logical relationship between the software components of any software system.

2 Approach

To achieve its goal a top-down approach will be used in this class. The essentials for succeeding with this approach consists of: (a) apply a systematic analysis of the components of the system software; (b) deduce the structuring which allows system software components to be specified in terms of other system software components; (c) study the mechanisms of interaction between system software components; (d) explain the mechanism of integration of system software components. This will be achieved by studying the behavior of the system software as a functional whole called *software system*.

3 Topics

The topics of the course are split into three parts: methodology, programming support environment, and execution support environment.

3.1 The Methodology

Part 1 of the course is devoted to the presentation of a system methodology, appropriate for discussing system software. The fundamental object of this methodology is the concept of a system. This concept is discussed both as a mathematical construction (illustrated by algebraic and logical systems) and as an ad hoc construction (illustrated by ad hoc systems chosen from various fields of interest in computer science). A special role will be devoted to the hardware organized as a system called the hardware system, and to the transition systems used to explain the behavior

of the hardware system. The two concepts, system software and software system, frequently used in computer science, are carefully specified and differentiated. A systematic approach for designing software systems is defined. The concept of a software system is then taken as a building block for the structuring of the system software. The system software is then organized as a software system that manages computer resources and provides services to computer users. A vertical and a horizontal structuring relationship of the programs and documents defining the system software is then defined.

By vertical structuring components of the system software are layered on a hierarchy of levels. A level of this hierarchy is defined in terms of the components provided by the levels already defined. The hardware system is taken as the first level of this hierarchy. The interface relationship between the components of the system software vertical hierarchy is then established.

Each level of the system software vertical hierarchy is discussed as a horizontal structure. The elements of this horizontal structure are specific software components of the system software organized as software systems. A systematic approach is used in order to formally define the software system components that constitute a level of the system software hierarchy.

The specific problems posed by the interaction between the software system components of a horizontal level of the system software hierarchy are discussed and illustrated. The problems raised by the reliability, efficiency, convenience, and evolution of a system software are introduced and illustrated. Two major levels of the vertical hierarchy, the program execution support environment and the program development (programming) support environment are given a special attention.

3.2 Programming Support Environment

The Programming Support Environment (PSE) [1] of the system software is discussed as the collection of tools offered by a computer platform to computer users to help them use the computer to develop programs that solve their problems.

The main mechanism offered by the PSE to computer users that helps them develop programs to solve their problems is introduced as the *translator*. A translator is a program that inputs programs written in a source programming language and maps them into programs written in a target programming language, while preserving the computation source language programs represent. Depending on the source and the target languages of a translator and the mechanism of target program generation and execution, we distinguish between Compilers, Assemblers, Linker/Loaders, and Interpreters. A mathematical methodology for translator specification and implementation will be sketched and will be illustrated.

[1] PSE is also the acronym for Problem Solving Environment

The target language will constantly be the *machine language* of the processor running on the computer platform. The source language will be an abstraction of the target language. The evolution of programming languages will be examined as layers of machine language abstractions from the assembly language to various high-level languages. A special attention will be given to the concept of a control language seen as the interface between a user and the operating system supporting the tools provided in the PSE. The following components of a PSE will be illustrated:

1. Dedicated program packages (such as mathematical and statistical). This illustrates the service-oriented versus programming-oriented approach of computer-based problem solving process.

2. The assembler specification and implementation. Java Virtual Machine (JVM) will be used as a hardware abstraction and a machine language for the JVM will be developed. The student will learn how to write JVM assembly language programs and how to designs a JVM interpretor that maps JVM programs into programs of a given hardware, thus implementing Java slogan "write once and execute always". This illustrates the common approach for system program development.

3. The Loader/Linker specification and implementation. With Loader/Linker the student is introduced to the problems posed by program execution. JVM Class-Loaders will be examined. The concepts of machine language and machine language program will be discussed.

3.3 Execution Support Environment

The Execution Support Environment (ESE) is discussed as a software system that manage computer resources of the computer platform and the processes running on the computer platform. ESE is illustrated by the operating system. The components of the operating system itself are layered on the levels of a hierarchy. The mechanism of a system call (system function call) will be discussed as a tool for implementing this hierarchy relation. The following layers of an operating system will be discussed:

1. Interrupt System.

2. Process Management System.

3. Memory Management System.

4. Input/Output Management System.

5. Information Management System (File System).

4 Homework

The course will be accompanied by assignments and projects. A system programming language needs to be used in this respect. C-language and Java are the programming languages suggested as tools for system software program development. The Linux machines existent in various Labs of Computer Science Department controlled by a version of the Unix operating system will be used in this offering of 22C:113. Each student will receive an account on these machines in the first day of class. Using this account students can develop their programs directly on the Linux machines or they can port them from other machines to the Linux machines available in the department.

4.1 Assignments

The instructor will keep the student's interest daily focusing on the matters discussed in class by means of the assignments. An important part of the assignment work is *student contribution to class evolution.* This consists of student participation to the class development and will be measured by random check on student class attendance. In addition, we will use written assignments that will contain questions and problems to be solved by students as their homework. These questions and problems are efficient learning tools if they push the student to read and to search library materials for solving them. Therefore, they will complete the students instruction, teaching them to use the library, to choose readings, and to solve specific problems in software system. We will try to keep a rhythm of one assignment every other two weeks of classes.

4.2 Projects

The purpose of the projects in this class is to teach students practical skills for designing and implementing system software. Projects will illustrate specific problems posed by software design and implementation. Therefore, the projects of this class concern programs which are actually needed in the system software of any computer platform. These programs should be implemented such that they become building blocks in a component-based software construction. Three projects will be implemented in this class. The evaluation of the projects will focus on problem specification, solution implementation and documentation, and on the use of the software thus implemented and documented.

5 Textbooks

This offering of 22C:113, Introduction to System Software will be taught by a dual-approach. During class presentation we will discuss concepts using instructor's lecture notes based on an updated version of the book T. Rus and D. Rus, *System Methodology for Software*, World Scientific 1996. The lecture notes will be available on the Website of the class at `http://www.cs.uiowa.edu/~rus/`. Concept illustration will be accomplished by the assignments and projects selected from the book D. Hoover, *System Software with C and Unix*, Addison-Wesley 2009, available at IMU Bookstore. Problems raised by programming support environment will be illustrated by the design and implementation of Java Virtual Machine. The book chosen as the textbook for the Java Virtual Machine, is "Programming for the Java Virtual Machine", by Joshua Engel, available at IMU Bookstore. Problems raised by process execution environment will be illustrated by C-language simulation of various components of an operating system.

6 Grading Procedure

The assessment of students in this class will be determined by their scores obtained in assignments, projects, one midterm exam, and one final exam. The midterm is an in-class exam, to be scheduled later. This exam will check the knowledge acquired by the students from the first day of class until the date of the exam. The final exam is comprehensive and mandatory.

The student's final result will be computed as follows:

- Assignment results are 15% of the final result;
- Project results are 15% of the final result;
- Midterm result is 35% of the final result;
- The final exam result is 35% of the final result;

The grades are determined as follows:

1. An **A** is obtained if the final result satisfies the restrictions:
   $90 < finalresult \leq 100$;
2. A **B** is obtained if the final result satisfies the restrictions:
   $70 < finalresult \leq 90$;
3. A **C** is obtained if the final result satisfies the restrictions:
   $50 < finalresult \leq 70$;

4. A **D** is obtained if the final result satisfies the restrictions:
   $30 < finalresult \leq 50$;

5. An **F** is obtained if the final result satisfies the restrictions:
   $0 \leq finalresult \leq 30$;

These ranges are not absolute. However, the lower bounds will not be raised any higher. We will use + and - attached to the letter score.

Good Luck!