EECS 22L: Software Engineering Project in C Language

Lecture 1

Rainer Dömer

doemer@uci.edu

The Henry Samueli School of Engineering
Electrical Engineering and Computer Science
University of California, Irvine

Lecture 1: Overview

- Course Administration
  - New courses in Computer Engineering
  - Course outline and overview
  - Projects and deliverables
  - Team work!
  - Grading policy and exams
  - Web page and programming setup
- Introduction to Software Engineering
  - General software engineering
  - Software development process

Course Administration

- New Programming Courses in Computer Engineering
  - EECS 22, *"Advanced C Programming"*
  - EECS 22L, *"Software Engineering Project in C"*
- Effective in 2012/13 Catalogue of Classes
  - Physics 51A, 52A has been replaced with EECS 22, 22L
  - Ongoing students can opt to choose the new plan of study
    - Automatic approval of 22/22L to fulfill the 52A/51A requirement
- EECS 22
  - First time offered in Fall 2011 (Instructor R. Dömer)
  - Recent offering in Fall 2012 (Instructor R. Dömer)
- EECS 22L
  - First time offered in Winter 2012 (Instructor P. Chou)
  - Now offered in Winter 2013 (Instructor R. Dömer)

Course Administration

- Changes to Computer Engineering Program
  - Delete Physics 51A and 52A from Math and Basic Science
  - Add EECS 22 and EECS 22L to Core Courses
- Sample Program of Study

FALL	WINTER	SPRING
Freshman		
Mathematics 2A	Mathematics 2B	Mathematics 2D
EECS12	Physics 7C/7LC	Physics 7D, 7LD
General Education	General Education	EECS20
General Education		General Education
Sophomore		
Mathematics 2J	Mathematics 3D	Mathematics 6D
Physics 7E~~, 52A~~	Physics ~~51A,~~ 52B	EECS40
EECS22	**EECS22L**	EECS70B, 70LB
EECS31	EECS31L	General Education
	EECS70A	
Junior		

EECS 22: Advanced C Programming

- *"All you want to know about C Programming"*
  - Review and reinforce basic C programming concepts
  - Study advanced features in detail
  - Put concepts and tools to their best use
- Features
  - Dynamic data structures using **`malloc()`**, **`free()`**
  - Keywords **`static`**, **`register`**, **`auto`**, **`extern`**, **`volatile`**, …
  - Advanced data types, variable-length arguments, …
  - Libraries, Makefile, …
- Tools
  - C preprocessor, compiler, and linker
  - Debugger 'gdb' and 'ddd'
  - Dynamic memory allocation checker 'valgrind'

EECS 22L: Software Eng. Project in C

- *"Developing real C Programs in a Team"*
  - Hands-on experience with larger software projects
  - Introduction to software engineering
    - Specification, documentation, implementation, testing
  - Team work
- Features
  - Design efficient data structures, APIs
  - Utilize programming modules, build libraries
  - Develop and optimize contemporary software applications
- Tools
  - Scripting 'make'
  - Version control 'cvs'
  - Testing and debugging with 'gdb', 'gprof', 'valgrind', …

EECS 22L: Software Eng. Project in C

- Catalogue Data
  - **EECS 22L Software Engineering Project in C Language (Credit Units: 3) W.**
  - Hands-on experience with the ANSI-C programming language.
  - Medium-sized programming projects, team work.
  - Software specification, documentation, implementation, testing.
  - Definition of data structures and application programming interface.
  - Creation of program modules, linking with external libraries.
  - Rule-based compilation, version control.
  - Prerequisites: EECS 22
  - (Design Units: 3)

EECS 22L: Software Eng. Project in C

- Course Outline
  - Software engineering topics, including specification, documentation, implementation, testing, debugging, project planning, organization, maintenance, version control, organization of source files, header files, modules
  - Compilation flow, Makefile, shell scripting
  - Definition of data structures and application programming interface
  - External libraries, system programming, POSIX API, interrupts
  - Introduction to C++ language, syntax and semantics, references, inline functions, default arguments, classes, members, and methods, object creation and deletion (constructors, destructors)

Course Administration

- Course Overview (tentative!)

Week	Lecture topics	Project tasks	
1	Software engineering flow	Project 1	Application specification
2	Software architecture, version control		Software architecture specification
3	Modules, libraries, documentation		Documentation, implementation
4	Software testing		Implementation, testing, debugging
5	Software packaging, installation		Delivery, installation, deployment
6	Project planning, organization, maintenance	Project 2	Application specification
7	Data structures and APIs		Software architecture specification
8	System programming, shell scripting		Documentation, implementation
9	Introduction to C++		Implementation, testing, debugging
10	Course wrap up		Delivery, installation, deployment

Course Administration

- Projects and Deliverables

Project	Task	Points	Deliverable	Due
Project 1: Chess Game	Application specification	100	Chess_User_Spec.pdf	Jan 14, 12pm (noon)
	Software specification	100	Chess_SW_Spec.pdf	Jan 21, 12pm (noon)
	Software alpha version	100	Chess_Alpha.tar.gz	Jan 28, 12pm (noon)
	Testing plan, status	100	Chess_Testing.pdf	Feb 04, 12pm (noon)
	Software release	100 (+20)	Chess.tar.gz Chess_Source.tar.gz	Feb 11, 12pm (noon)
Project 2: TBD	Application specification	100		Feb 18, 12pm (noon)
	…	…	…	…
	Software release	100 (+20)		Mar 18, 12pm (noon)

Course Administration

- Project Teams
  - Projects will be performed by student teams
    - Project 1: Teams of 4-5 students
    - Project 2: Teams of 8-10 students
  - *Team work* is an essential aspect of this class!
    - ➢ Every student needs to contribute to the team effort!
    - ➢ Tasks may be assigned to individual team members, but all members share the responsibility for deliverables.
  - Team resources
    - Dedicated online discussion forum
    - Dedicated team account on the server
  - Collaboration
    - Share code, data, and documents (within teams only!)
  - Competition
    - Teams compete for the customer!

Course Administration

- Grading Policy
  - Programming projects 50% (team effort)
  - Midterm examination 15% (individual effort)
  - Final examination 35% (individual effort)
- Effort Assessment
  - Team: Project deliverables
  - Individual student: Exams, plus feedback from peers, TAs
- Exams
  - Midterm exam Project 1 contribution (week 6)
  - Final exam Project 2 contribution (finals week)
    - ➢ Short oral presentations by individual students at the computer
    - ➢ Explain original contribution to the team, original deliverable (source code and/or documentation), and answer any questions

Course Administration

- Class Schedule
  - EECS 22L *"Meets for 1 hour of lecture, 1 hour of discussion and 3 hours of laboratory each week for 10 weeks"* (quote from EECS 22L course outline)
  - However, current schedule of classes lists 3 hours of lecture
  - ➢ Use Tuesday slot for actual lectures
  - ➢ Use Thursday slot for project topics at hand (TBD, optional)
  - Example: Week 1
    - Tuesday: Lecture 1: Introduction
    - Thursday: "Customer Interview"
      - "Negotiate" the specification and features for project 1
  - Example: Week 2
    - Tuesday: Lecture 2: Version Control
    - Thursday: "Introduction to GUI programming"

Course Administration

- Course web pages online at **`http://eee.uci.edu/13w/18020/`**
  - Instructor information
  - Course description and contents
  - Course policies and resources
  - Course and project schedule
  - Course communication
    - Message board (announcements, class discussion, and dedicated team/project discussion)
    - Email (administrative issues)
    - Office hours (instructor and TAs)
- Linux system environment
  - Same as for EECS 22
  - EECS Linux server: **`ladera.eecs.uci.edu`**

Introduction to Software Engineering

- What is Software Engineering?
  - Software engineering is the application of *engineering* to *software*
  - Software engineering can be defined as:
    - *The application of, or*
    - *the study of*
      a systematic, disciplined, quantifiable approach to the development, operation and maintenance of software.
- EECS 22L …
  - … is *not* a complete course on software engineering!
  - … consists of projects that demonstrate the essential tasks and tools of software development in C!

Introduction to Software Engineering

- General Software Engineering Process
  - Project feasibility and planning
  - Requirements analysis, definition, and specification
  - Design of the system and software
    - E.g. using UML (Unified Modeling Language)
  - Implementation
    - Programming (modules, system)
    - Testing against the specification (units, system)
  - Delivery, operation, maintenance
- EECS 22L Software Development Process
  - Software specification and documentation
  - Software architecture design and documentation
  - Implementation, testing, and debugging
  - Release

Introduction to Software Engineering

- EECS 22L does *not* cover
  General Software Engineering Topics
  - General processes of software engineering
  - General feasibility study and requirements engineering
  - General design strategy and documention
    - E.g. UML
  - Usability and reliability studies
  - Legacy systems and evolution of software
  - General project or personnel management
  - Consideration of economic, legal, social and other factors
  - Verification of software
  - …

Software Development Process

- EECS 22L covers
  the essential tasks and tools of software development
  - Using ANSI-C programming language
    - With an outlook into object-oriented design, i.e. C++
  - In Linux environment
    - With typical Linux tools chain,
      e.g. **`gcc`**, **`make`**, **`gdb`**, **`ssh`**, **`cvs`**, **`gtar`**, **`bash`**, …
  - With focus on practical aspects
    - Medium-size projects
    - Programming practice!
    - Team work!!

Software Development Process

- EECS 22L Software Development Process
  1. Application specification
     - User's perspective (aka. client or customer)
     - Documentation
  2. Software architecture design and specification
     - Developer's perspective
     - Documentation
  3. Implementation, testing, and debugging
     - Unit testing
     - System testing
  4. Release
     - Binary program and documentation
     - Source code and documentation

Software Development Process

1. Application Specification
   - Goal: Specify the user experience!
     - What does the user (aka. customer, client) want?
     - What does he need to provide? What does he get?
     - What does the software do? What features does it have?
   - Deliverable: Application Specification Document
     - Input data including options and parameters
       - What? In which format? In which order? From which device? …
     - Processing
       - What? (*not* how!) What happens? What is presented?
     - Output
       - What? In which format? In which order? To which device? …
   - ➢ Some features may be intentionally left "unspecified"
   - ➢ Specification can be the starting point of the final documentation!