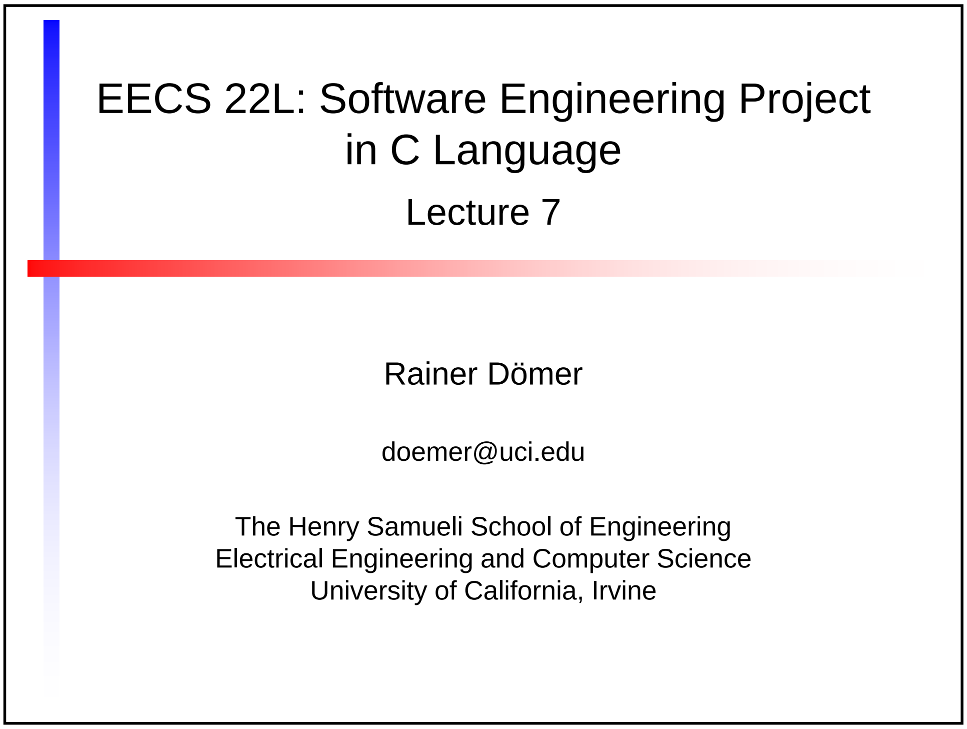


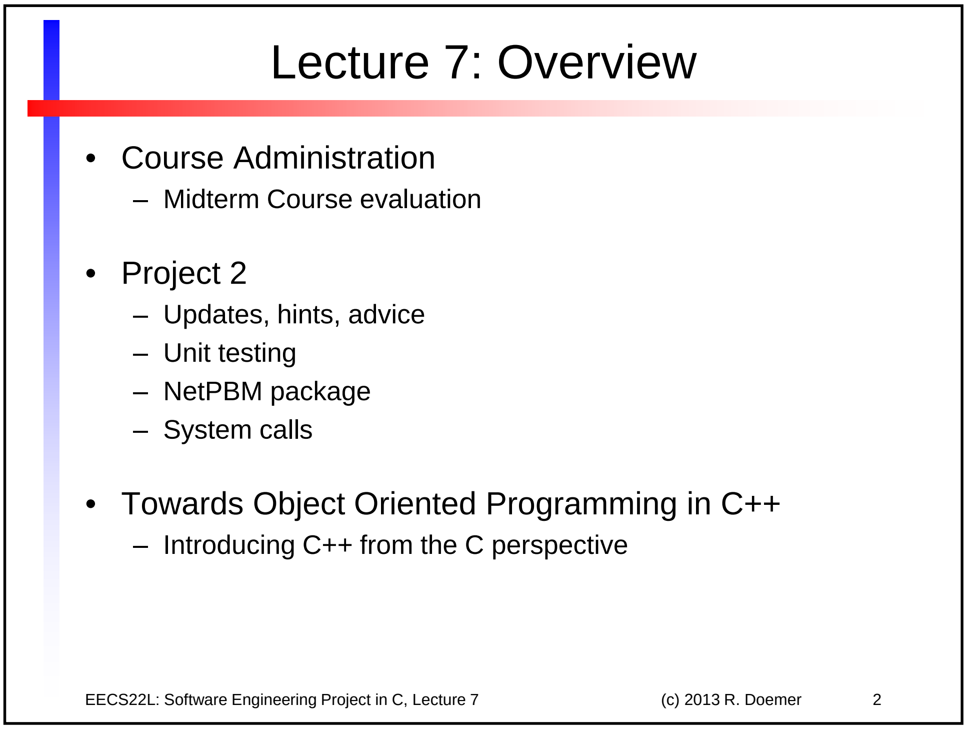
EECS 22L: Software Engineering Project in C Language

Lecture 7

Rainer Dömer

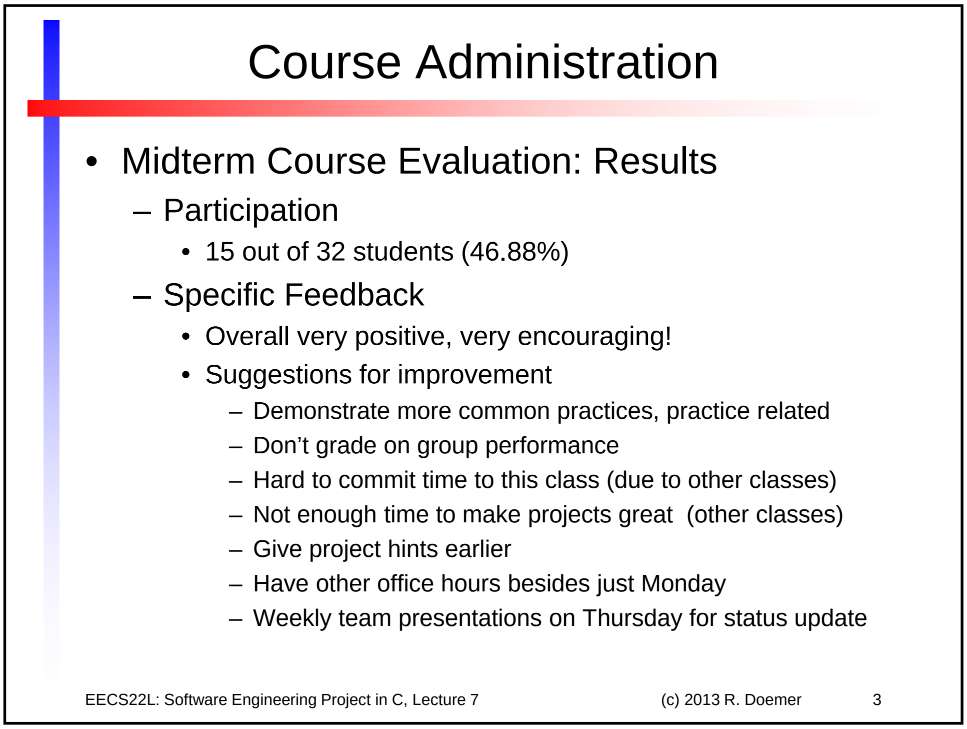
doemer@uci.edu

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



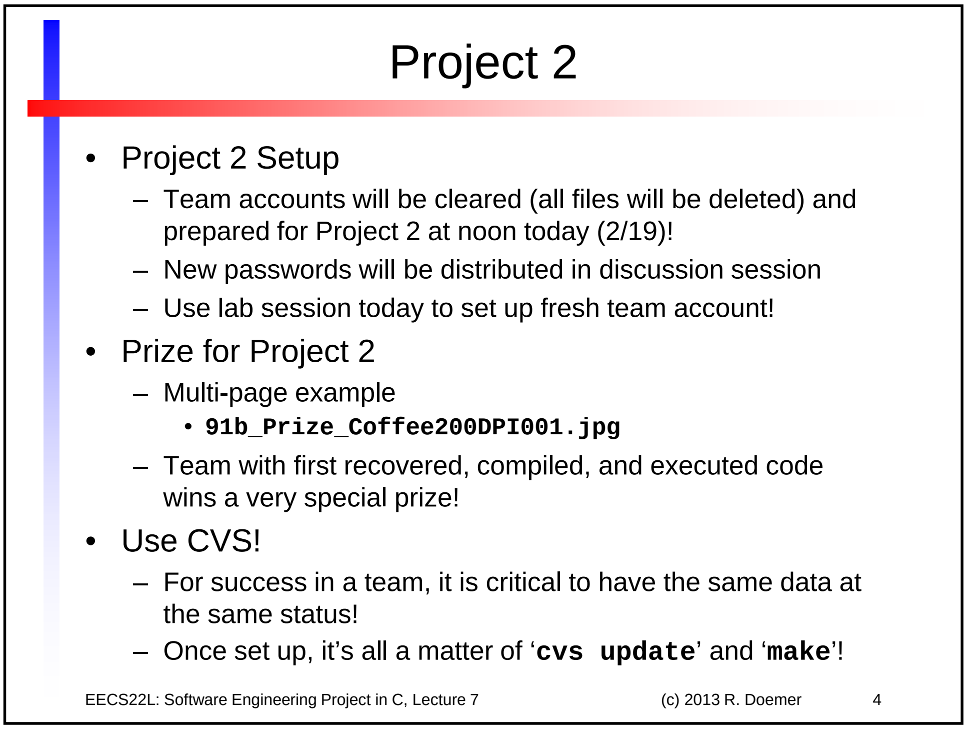
Lecture 7: Overview

- Course Administration
  - Midterm Course evaluation
- Project 2
  - Updates, hints, advice
  - Unit testing
  - NetPBM package
  - System calls
- Towards Object Oriented Programming in C++
  - Introducing C++ from the C perspective



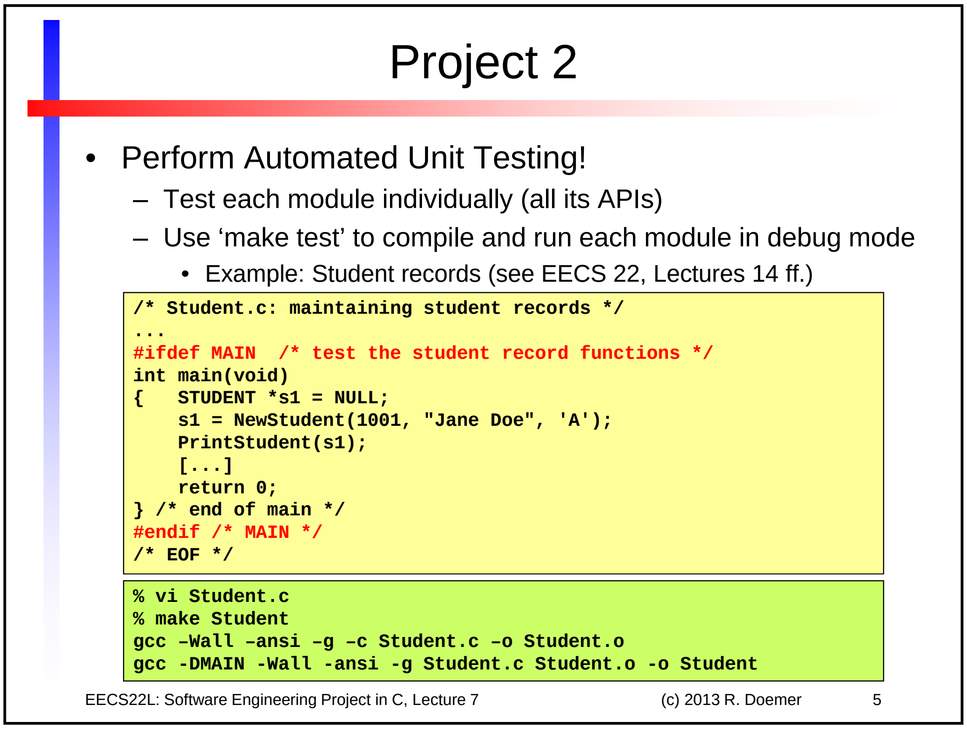
Course Administration

- Midterm Course Evaluation: Results
  - Participation
    - 15 out of 32 students (46.88%)
  - Specific Feedback
    - Overall very positive, very encouraging!
    - Suggestions for improvement
      - Demonstrate more common practices, practice related
      - Don’t grade on group performance
      - Hard to commit time to this class (due to other classes)
      - Not enough time to make projects great (other classes)
      - Give project hints earlier
      - Have other office hours besides just Monday
      - Weekly team presentations on Thursday for status update



Project 2

- Project 2 Setup
  - Team accounts will be cleared (all files will be deleted) and prepared for Project 2 at noon today (2/19)!
  - New passwords will be distributed in discussion session
  - Use lab session today to set up fresh team account!
- Prize for Project 2
  - Multi-page example
    - **`91b_Prize_Coffee200DPI001.jpg`**
  - Team with first recovered, compiled, and executed code wins a very special prize!
- Use CVS!
  - For success in a team, it is critical to have the same data at the same status!
  - Once set up, it’s all a matter of ‘**`cvs update`**’ and ‘**`make`**’!

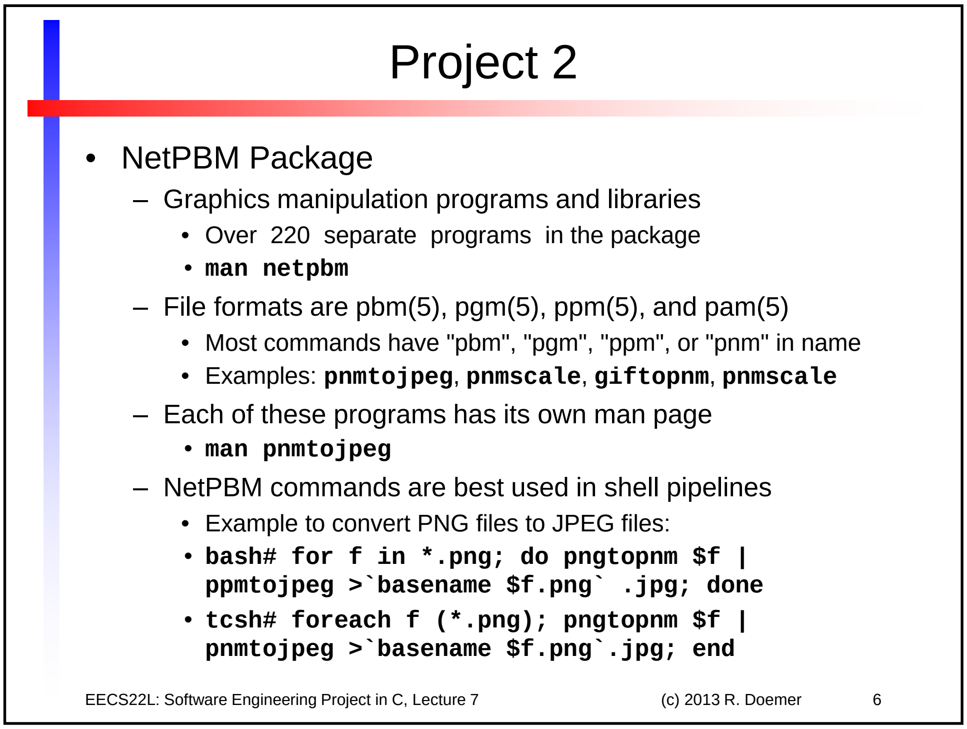


Project 2

- Perform Automated Unit Testing!
  - Test each module individually (all its APIs)
  - Use ‘make test’ to compile and run each module in debug mode
    - Example: Student records (see EECS 22, Lectures 14 ff.)

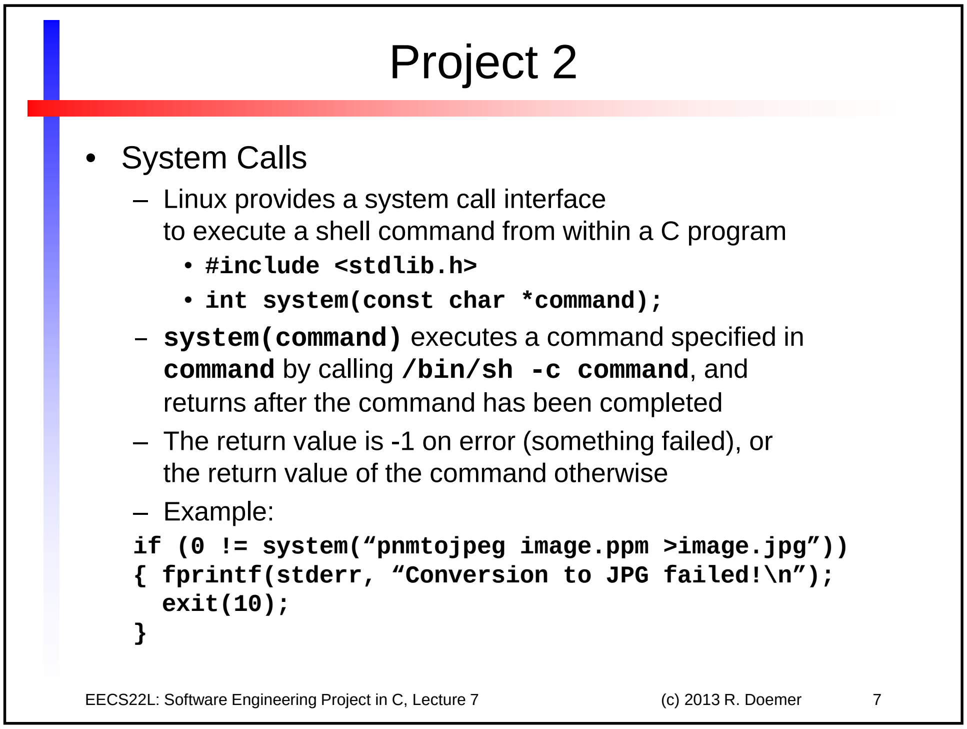
```
/* Student.c: maintaining student records */
...
#ifdef MAIN  /* test the student record functions */
int main(void)
{   STUDENT *s1 = NULL;
    s1 = NewStudent(1001, "Jane Doe", 'A');
    PrintStudent(s1);
    [...]
    return 0;
} /* end of main */
#endif /* MAIN */
/* EOF */
```

```
% vi Student.c
% make Student
gcc –Wall –ansi –g –c Student.c –o Student.o
gcc -DMAIN -Wall -ansi -g Student.c Student.o -o Student
```



Project 2

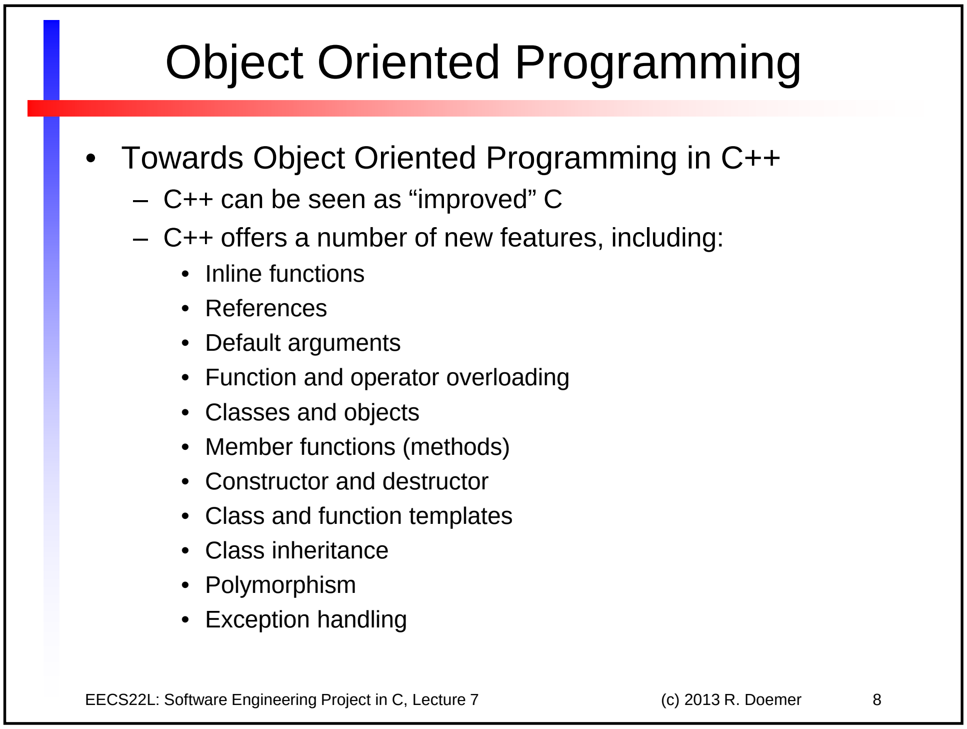
- NetPBM Package
  - Graphics manipulation programs and libraries
    - Over 220 separate programs in the package
    - **`man netpbm`**
  - File formats are pbm(5), pgm(5), ppm(5), and pam(5)
    - Most commands have "pbm", "pgm", "ppm", or "pnm" in name
    - Examples: **`pnmtojpeg`**, **`pnmscale`**, **`giftopnm`**, **`pnmscale`**
  - Each of these programs has its own man page
    - **`man pnmtojpeg`**
  - NetPBM commands are best used in shell pipelines
    - Example to convert PNG files to JPEG files:
    - `bash# for f in *.png; do pngtopnm $f | ppmtojpeg >`basename $f.png` .jpg; done`
    - `tcsh# foreach f (*.png); pngtopnm $f | pnmtojpeg >`basename $f.png`.jpg; end`



Project 2

- System Calls
  - Linux provides a system call interface to execute a shell command from within a C program
    - **`#include <stdlib.h>`**
    - **`int system(const char *command);`**
  - **`system(command)`** executes a command specified in **`command`** by calling **`/bin/sh -c command`**, and returns after the command has been completed
  - The return value is -1 on error (something failed), or the return value of the command otherwise
  - Example:

```
if (0 != system(“pnmtojpeg image.ppm >image.jpg”))
{ fprintf(stderr, “Conversion to JPG failed!\n”);
  exit(10);
}
```



Object Oriented Programming

- Towards Object Oriented Programming in C++
  - C++ can be seen as “improved” C
  - C++ offers a number of new features, including:
    - Inline functions
    - References
    - Default arguments
    - Function and operator overloading
    - Classes and objects
    - Member functions (methods)
    - Constructor and destructor
    - Class and function templates
    - Class inheritance
    - Polymorphism
    - Exception handling