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
  - New project teams
- Project 2
  - Introduction
  - Focus: GUI, CVS, testing
  - Application specification
  - Technical advise

Course Administration

- Midterm Course Evaluation: Results
  - Participation
    - 38 out of 65 students (58.46%)
  - Specific Feedback
    - Mostly very positive and encouraging
      - Overall grade: 32 “A”, 5 “A-”, 1 “B”
    - Some specific comments
      - Lectures on CVS and SDL “not very useful”, “not interesting”
      - Slides on CVS and SDL “were a great resource”
      - More advise and discussion in lectures (“similar to lecture 2”)
      - “Projects with smaller teams”, size limits creativity and pace
      - “Best EECS teacher, ever!”
      - “You look cool and I like your voice but that's about it.”

Course Administration

- New Teams
  - New project 2 teams have been formed
    - Most preferences have been met
    - Very few exceptions (cyclic conflicts)
  - 10 teams, 6 to 8 members each
    - See individual team emails
- New Team Accounts
  - All team accounts have been cleared (all files deleted)
  - New passwords will be distributed in discussion session
    - Use lab sessions this week to set up fresh team account

Project 2

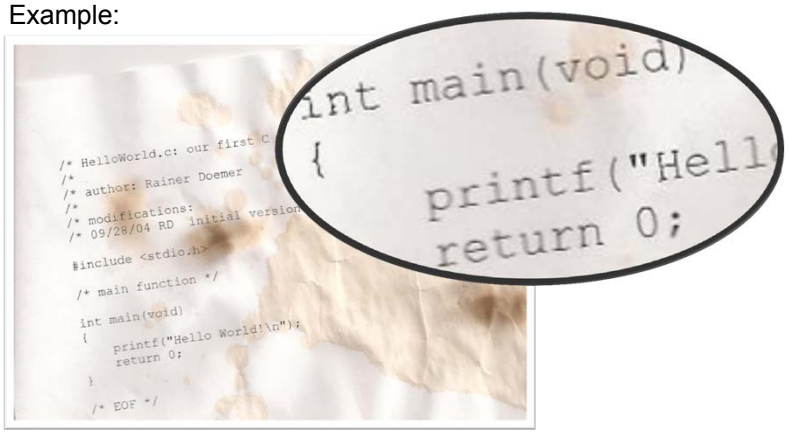
- Introduction
  - OCR: Optical Character Recognition
  - Story:

    Mr. Nob Ackup runs a small software company that implements small-scale projects for its clients. One day, a thunder storm strikes. Lightning creates a sudden power surge followed by an outage. After power is restored, Mr. Nob Ackup finds his computer and backup systems damaged and can only partially recover the source code of his projects from old tapes. Luckily, his software engineers find stacks of papers with printed hard copies of most missing source files that can be scanned into digital images.
  - Task:

    Design a software package for Mr. Nob Ackup's company so that he can efficiently convert the scanned images of printed source code back into compilable files.

Project 2

- Introduction
  - OCR: Optical Character Recognition
  - Example:



Project 2

- Introduction
  - OCR: Optical Character Recognition

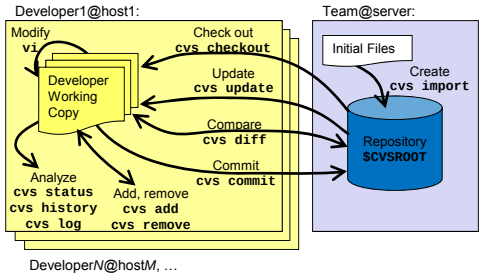
➢ Focus
  - Graphical User Interface (GUI)
    - mandatory
  - Version control with CVS
    - mandatory
  - Testing
    - Unit test
    - System test

Project 2 Focus: GUI

- Graphical User Interface (GUI)
  - Differences to command-line interface
    - Input through events
    - Output through 2D pixel interface
    - Event-based main control loop
  - Available libraries
    - SDL: Simple DirectMedia Layer
      - **`http://www.libsdl.org/`**
    - GTK: GNOME or GIMP Toolkit
      - **`http://www.gtk.org/`**
  - ➢ Refer to Lecture 4 and online documentation

Project 2 Focus: CVS

- Version Control using CVS
  - Source code management
    - For successful team work, it is critical to have the same data at the same state
    - Once set up, it's all a matter of '**`cvs update`**' and '**`make clean all`**'



  - ➢ Refer to Lecture 3 and online documentation

Project 2 Focus: Testing

- Perform Automated Unit and System Test
  - Test each module individually (with specific test data)
  - Use **`make test`** to run each module in debug mode

  - ➢ Top-level **`Makefile`** should provide targets to run tests for each module and the system
  - Unit tests
    - **`% make test_gui`**
    - **`% make test_preproc`**
    - **`% make test_ocr`**
    - **`% make test_postproc`**
  - System test
    - **`% make test`**

Project 2 Focus: Testing

- Perform Automated Unit and System Test
  - Test each module individually (with specific test data)
  - Use **`make test`** to 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
```


Application Specification

- OCR: Program Specification
  - Basic and *advanced* functions: (Requirements and *goals*)
    1. Graphical user interface (GUI) for user interaction
    2. Loading a scanned image into the program and displaying it
    3. Image preprocessing
    4. OCR
    5. Text postprocessing
    6. Text file output
    7. Automated unit test for all major program modules

Application Specification

- OCR: Program Specification
  - Basic and *advanced* functions: (Requirements and *goals*)
    1. Graphical user interface (GUI) for user interaction
    2. Loading a scanned image into the program and displaying it
       a) image in JPEG format
       b) multiple input image formats, such as PPM, PBM, or BMP
       c) combine multiple scanned pages into a single output text file
    3. Image preprocessing
    4. OCR
    5. Text postprocessing
    6. Text file output
    7. Automated unit test for all major program modules

Application Specification

- OCR: Program Specification
  - Basic and *advanced* functions: (Requirements and *goals*)
    1. Graphical user interface (GUI) for user interaction
    2. Loading a scanned image into the program and displaying it
    3. Image preprocessing
       a) black and white conversion
       b) stain and wrinkle removal
       c) rotation (to level a slanted scan)
       d) cropping (to select sections suitable for OCR)
    4. OCR
    5. Text postprocessing
    6. Text file output
    7. Automated unit test for all major program modules

Application Specification

- OCR: Program Specification
  - Basic and *advanced* functions: (Requirements and *goals*)
    1. Graphical user interface (GUI) for user interaction
    2. Loading a scanned image into the program and displaying it
    3. Image preprocessing
    4. OCR
       a) identifying and cropping single characters from the image
       b) comparing cropped characters with reference images from a font data base
       c) storing recognized characters in a text file data structure
       d) Support other fonts ("Courier New"*, "Helvetica", …)*
       *e) Support other styles (*mono-space*, italic, bold, …)*
       *f) Support other sizes (*12 point at 300 DPI*, 10pt at 200 DPI)*
    5. Text postprocessing
    6. Text file output
    7. Automated unit test for all major program modules

Application Specification

- OCR: Program Specification
  - Basic and *advanced* functions: (Requirements and *goals*)
    1. Graphical user interface (GUI) for user interaction
    2. Loading a scanned image into the program and displaying it
    3. Image preprocessing
    4. OCR
    5. Text postprocessing
       a) display the recognized text
       b) allow a text editor to be used on the recognized text
       c) Dictionary support:
       difficult to distinguish similar characters (1 vs. l, 0 vs. O, C vs. G, …)
       a dictionary (e.g. ANSI C keywords) may be used to fix such "typos"
    6. Text file output
    7. Automated unit test for all major program modules

Application Specification

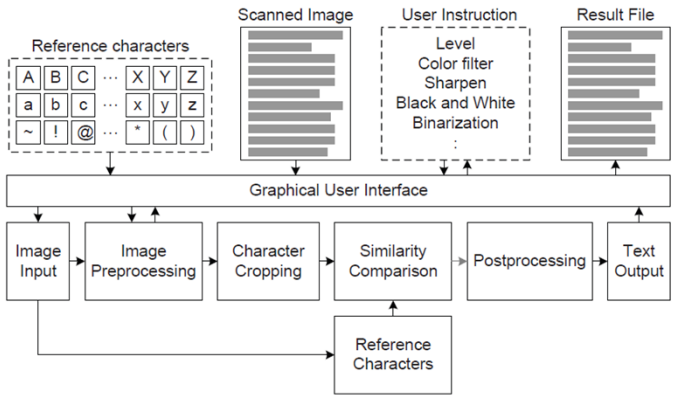
- OCR: Program Specification
  - Basic and *advanced* functions: (Requirements and *goals*)
    1. Graphical user interface (GUI) for user interaction
    2. Loading a scanned image into the program and displaying it
    3. Image preprocessing
    4. OCR
    5. Text postprocessing
    6. Text file output
       a) store the recognized text in a text file
       b) store the recognized text in other formats
       c) run the C compiler on the recognized source code
    7. Automated unit test for all major program modules

Application Specification

- OCR: Program Specification
  - Basic and *advanced* functions: (Requirements and *goals*)
    1. Graphical user interface (GUI) for user interaction
    2. Loading a scanned image into the program and displaying it
    3. Image preprocessing
    4. OCR
    5. Text postprocessing
    6. Text file output
    7. Automated unit test for all major program modules
       a) Unit test
       b) System test

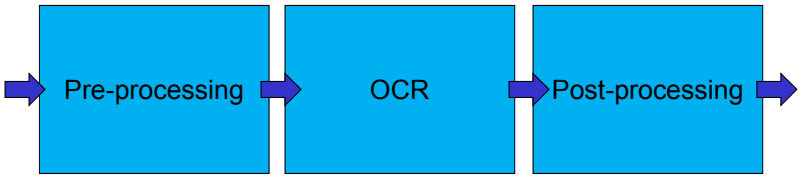
Technical Advise

- Overview of an OCR Program
  - Major system components



Technical Advise

- Overview of an OCR Program
  - Pipelining structure



Technical Advise

- Overview of an OCR Program
  - Pipelining structure with loops