CS240: Programming in C

Lecture 1: Class overview.

WELCOME to CS240

240 Team

- Instructor: Cristina Nita-Rotaru
- Special GTA: Gregor
- GTAs: Derek, Luojie, Shuvra
- UTAs: Scott, Xiangyu, Anant, Yudong

Why learn C (1)

- **<u>C is one of the foundations for CS:</u>**
  - Contains/applies principles from programming languages, computer architectures, operating systems, network communication, database, graphical user interface (GUI), graphics, image processing, parallel processing, multi-threads, real-time systems, device drivers, data acquisition, algorithms, numerical analysis, and computer game.

What does this buy you?

- **Understanding:** understand better the interaction between machine and software:
  - "…teaches individuals how computers really work"
  - "…built a foundation you'll be thankful for every 300+ level course "

Why learn C (2)

- **C is the most commonly used programming language in industry.**
  - Next two popular are Java and C++
  - Language of systems programming: low-level control over the OS, networking, crypto operations, email, games, embedded systems have higher performance when written in C

http://www.langpop.com

What does this buy you?

- **<u>Helps you be as prepared as possible for a job:</u>**
  - Most of the employers want candidates to know multiple languages
  - Will prepare you better for a job interview
  - Gives you more opportunities within a company

Why learn C (3)

- **C is the base for almost all popular programming languages.**
- Because of the performance and portability of C, almost all popular cross-platform programming languages and scripting languages, such as C++, Java, Python, Objective-C, Perl, Ruby, PHP, Lua, and Bash, are implemented in C and borrowed syntaxes and functions heavily from C.
- Almost all languages can interface with C and C++ to take advantage of a large volume of existing C/C++ libraries. Many of their toolkits, modules or packages are written using C or C++.

What does this buy you?

- **<u>It will help you learn quickly other languages</u>**
- **<u>It will allow you to interface with many other languages</u>**

A word of caution …

" With great power,

comes great responsibility”

- C gives the user greater power than other languages
  - Fine-grain control over resources
  - Explore the interaction between software and hardware
- C is less forgiven with user’s mistakes

Course information

- Meetings
  - Tu Th 9:30-10:20pm Armstrong 1010
- Professor contact info:
  - Office: LWSN 2142J
  - Email: cnitarot@purdue.edu
  - In person office hours: by appointment
  - All emails should have cs240 in Subject
- TA:
  - Piazza
- Class webpage

http://www.cerias.purdue.edu/~crisn/courses/cs240_Fall_2013

- Communication via piazza

Course outline

- Tentative schedule is available on the class website.
- Lists the plan for lectures, labs, exams, and projects.
- Everything happens in class: lectures, reviews for exams, solving the exams.
- All communication happens on piazza: all the notifications, advice posting of labs/projects happens on piazza.

Grading policy

• Labs	10%
• Projects	40%
• Midterm 1	10%
• Midterm 2	10%
• Final	20%
• Class participation	10%

Labs and Projects

- **Learning how to program is achieved by PROGRAMMING (i.e. DOING the LABS and the PROJECTS)**
- **<u>THERE IS NO SUBSTITUTE FOR YOUR WORK</u>**
    - YOU CAN MISS SUBMITTING 1 LAB
    - MISSING MORE THAN 1 LAB will result in failing the class.
    - YOU HAVE TO SUBMIT ALL PROJECTS, missing to submit one or more projects will result in failing the class

Environment

- Linux
- gcc
- gdb
- make
- vim/emacs, your favorite editor
- VMware image

Due dates

- Both labs and projects are assigned on Tuesdays, posted on piazza after class
- Both labs and projects are due on Mondays at 10 PM
- Labs, 1 week, project 2-3 weeks
- Individual work, no teams

Put your name on Labs and Projects

- NAME and USER NAME

```
/***********************************/
/* CS240 Fall 2013 Project 1    */
/* John Doe                        */
/* john_doe_27                     */
/* This project counts till 10.    */
/***********************************/
```

Autograder

- You will receive a unique id that will allow you to test and submit the code
- More information will be provided in labs
- Do not submit in the last moment, you can submit many times before deadline we consider the latest copy

Coding style

- Labs and projects will be manually inspected
- You will receive a guide with some basic good coding habits we will check
- It is 20% of the grading of each project

Exams

- Closed books, closed notes
- There are no makeups
- Two midterms and 1 final exam
- We will solve solutions in class

Regrading

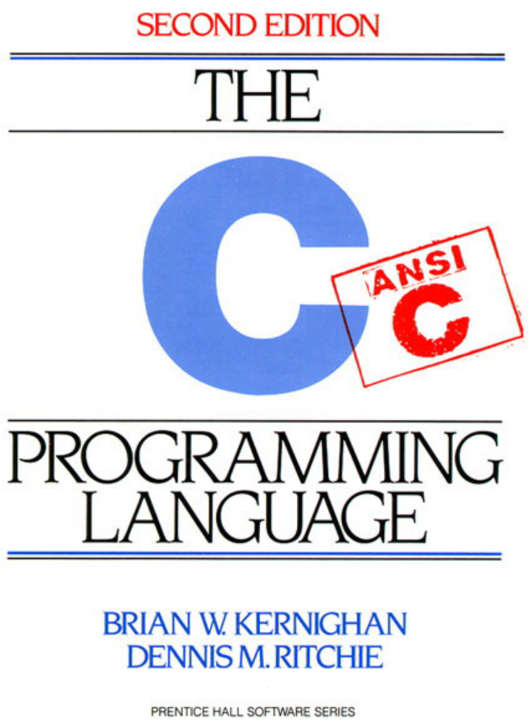
- YOU HAVE 1 WEEK to ASK for REGRADING of a lab, project or midterm
- Send mail to the course staff mail alias (admin240f13@cs) with a precise description of your grading request, and get a ticket
- Midterm/Final: handled by the Instructor
- Project/labs handled by TAs (cc me on all communication with TA)

Attendance

- **CLASS AND LABS attendance REQUIRED !!!**
- **Attendance taken in labs**
- Slides will be made available online before lecture
- YOU ARE STRONGLY RECCOMENDED TO TAKE NOTES
- CODE ALL THE EXAMPLES AND SOLVE THE PRACTICE EXERCISES

Reference material

- The C Programming Language, Brian W. Kernighan and Dennis M. Ritchie, 2nd Edition
- Lecture slides posted online



Academy integrity

- Class policy

http://homes.cerias.purdue.edu/~spaf/cpolicy.html

- Never have a copy of someone else's program in your possession and never give your program to someone else.
- **NO CHEATHING WILL BE TOLERATED.**

  ANY CHEATING WILL AUTOMATICALLY RESULT in F grade

Weather/Emergency

- In the event of a major campus emergency, course requirements, deadlines and grading percentages are subject to changes that may be necessitated by a revised semester calendar or other circumstances beyond the instructor's control.
- Please read:

http://www.itap.purdue.edu/tlt/faculty/QuickRefGuide.pdf

How to study

- Read the book, reference material, slides, man pages
- Code each example from class, don't just read it, code it
- Do the labs and projects
- Do the practice exercises from lectures
- Start small, then add functionality
- Make mistakes and observe output
- Make sure you always understand why it did not work and why the solution works

How to ask on Piazza

- Read the book, slides, notes
- Describe the problem clearly, using the right terms
- Add code in attached files
- Add output from compiler
- Add any other relevant information
- All the info has to be from running the program in the VM used for the class or the machine used in the lab

One last word …

- No meetings will be accepted with the TA or instructor the day projects or labs are due, or the day of exam
- Start early
- Plan carefully
- Submit your code often
- Don’t post solutions on piazza
- Don’t cheat

PIAZZA ACCOUNTS

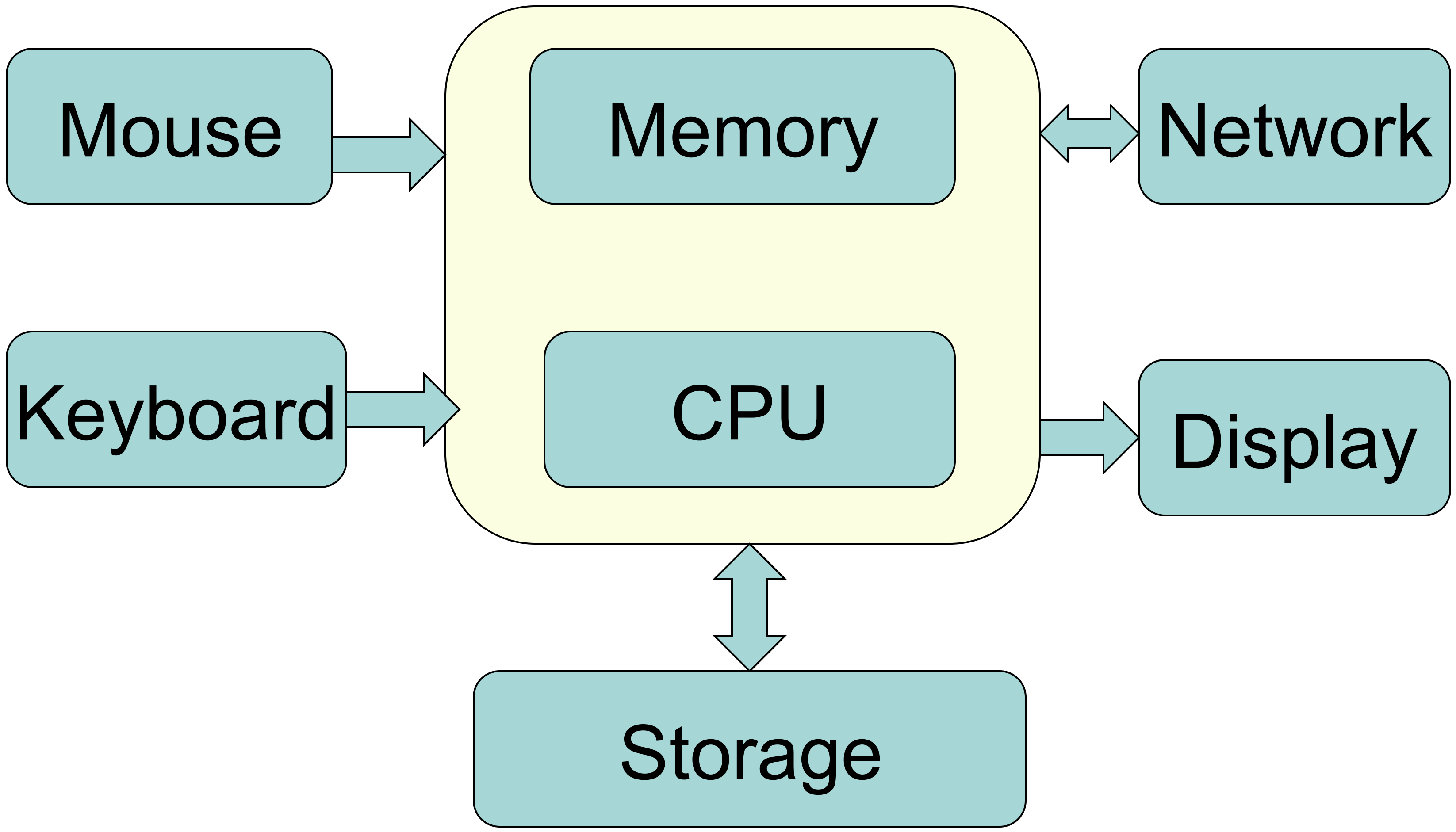
- If you have not received a piazza notification
- Email: dschatzl@purdue.edu
- Also cc me cnitarot@purdue.edu

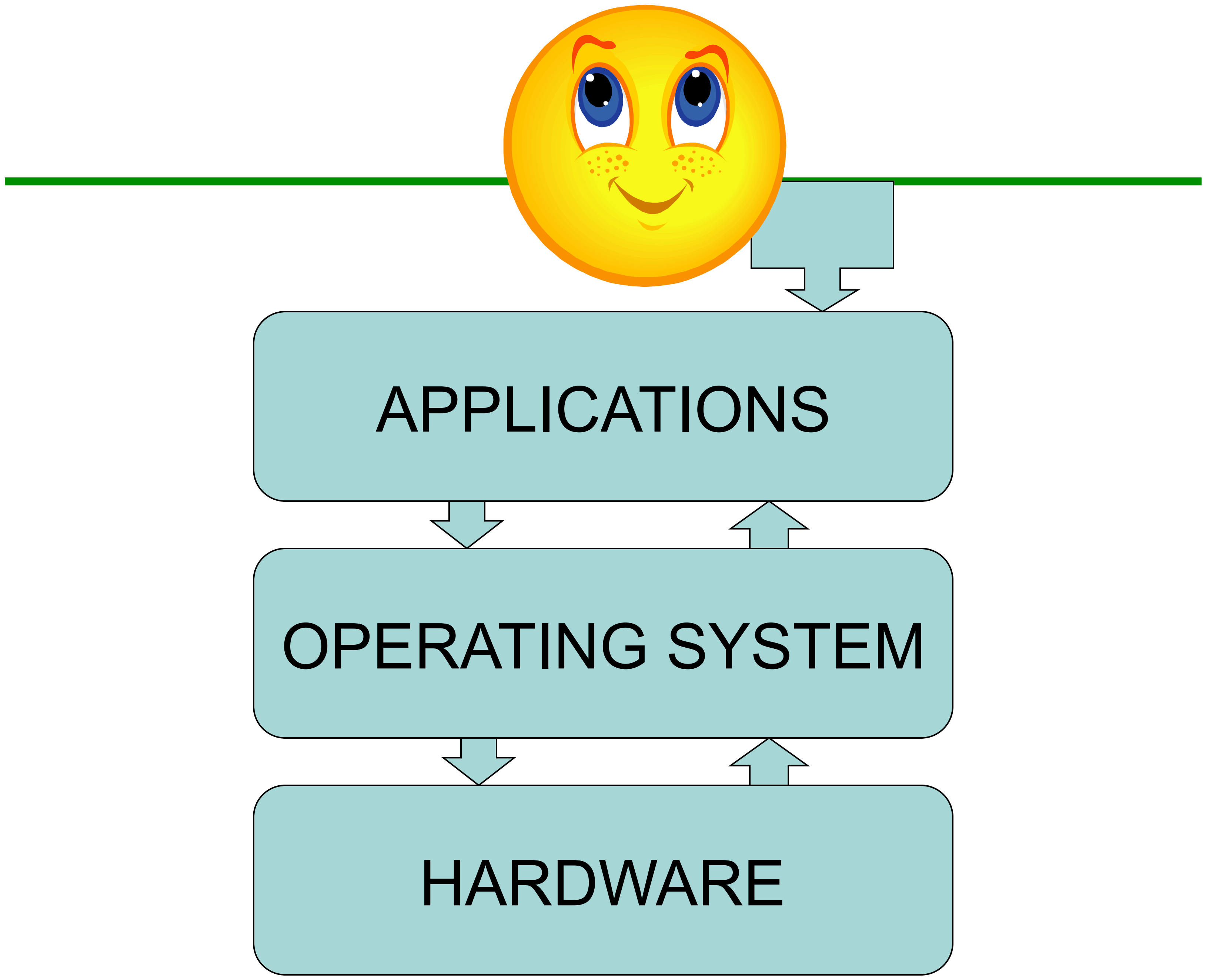
QUESTIONS?

Terminology

- What's a computer?
- What is hardware/software
- What's an algorithm ?
- What's a program?
- What's an operating system?
- What's a programming language ?
  - Machine language
  - Assembly language
  - High-level language

Computer architecture




APPLICATIONS
OPERATING SYSTEM
HARDWARE

OS Job

- Management of the processes and their access to resources
  - Memory
  - CPU access
  - I/O
  - Network
  - Other devices
- Interaction with the user
  - Graphic interface
  - Other devices

Algorithm/Program

- **Algorithm**: procedure for solving a problem in finite steps
- **Program**: set of instructions to the CPU, stored in memory, read and executed by the CPU

Machine and assembly language

- **Machine language** : binary information, specific to a CPU
  - How a CPU interprets data: e.g. how are memory addresses represented, how is an instruction coded, etc
  - This is the **binary or executable code**
- **Assembly language**: easier to write for people, using symbols, requires an assembler
  - Still need to think in terms of low level CPU steps
  - Still hardware-specific

High-level language

- Closer to human language
- Needs a compiler to convert it to machine language
- One can write programs in many high-level languages for the same CPU
- More portable
- Examples: C, C++, C#, Objective C, Java, SmallTalk, also Cobol, Basic, Pascal …

Readings for next lecture

K&R Chapter 1: A tutorial introduction