

CSE 31

Computer Organization

Lecture 1 – Course Info
C Programming

CSE 31: Spring 2019

▶ Lecturer

- Chi Yan “Daniel” Leung
- cleung3@ucmerced.edu
- Office Room: AOA 126
- Office Hours:
 - T/R: 10:00am – noon
 - W: 9:00am – 10:30am
 - By appointment

▶ TA

- Pooya Tavallali ptavallali@ucmerced.edu
- Hung-Yu Tseng htseng6@ucmerced.edu
- Yu-Ting (Julia) Chang ychang39@ucmerced.edu

▶ All email inquiries received before 5pm during school days will be replied within 48 hours

- Please follow the guidelines below for proper email communications
 - https://cms.cerritos.edu/uploads/ifacon/How_to_Email_your_Professor.pdf

Course Overview

▶ CatCourses

- Check regularly for announcements.
- Labs, Projects, and Homework Assignments will be posted and submitted there.
- Grades for assignments will also be found there (secure).

- ▶ 2 Lectures and 1 Lab per week
- ▶ 2 Mid-term exams (Feb. 27 and Apr. 17, in class)
- ▶ Final exam (May 11, 11:30am, classroom)
- ▶ 10 lab assignments
- ▶ 5 – 6 homework assignments
- ▶ 2 projects

Course Objectives

▶ Labs:

- Attendances are mandatory
- Each lab assignment is closed at 11:59pm of the 7th day after it is assigned. You cannot submit any work after it is closed.
- You must demo your lab submission within 14 days after it is assigned in order for it to be graded.
 - **ALL SUBMITTED LABS WITHOUT DEMO WILL NOT BE GRADED.**
 - If you demo it within 7 days, you are allowed to make corrections and re-submit it before the assignment is closed.
 - You will have time to demo your submissions to your TA during lab time of the following week.
- If you expect to submit your work late, you must request for approval from Daniel only **BEFORE** the due date.
 - All late submission requests after the due date will not be considered unless accompanied with proper documentations of excuses.
- Try to debug yourself before asking questions
- Follow the guidelines (see below) to debug and ask your TA for help.
- <http://www.cplusplus.com/forum/articles/28767/>
- <http://www.catb.org/esr/faqs/smart-questions.html>

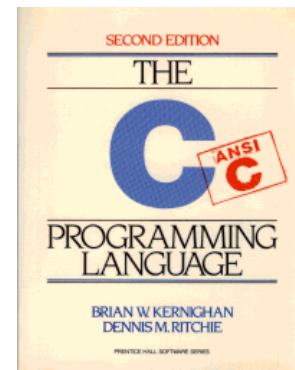
Course Objectives

- ▶ Learning C
 - If you know one, you should be able to learn another programming language largely on your own
 - If you know C++ or Java, it should be easy to pick up their ancestor, C
- ▶ Assembly Language Programming
 - This is a skill you will pick up, as a side effect of understanding the Big Ideas of computer organization
- ▶ Hardware design
 - We'll learn just the basics of hardware design

Course Material

► Text Books:

- Computer Organization and Design from zyBooks
 - Sign up/sign in at [zyBooks.com](https://www.zybooks.com)
- Enter zyBook code: **UCMERCEDCSE031LeungSpring2019**
 - **You must subscribe your own copy. Participation grade will be partly evaluated based on the activities within the subscription account.**
- The C Programming Language, Kernighan and Ritchie (K&R), 2nd edition



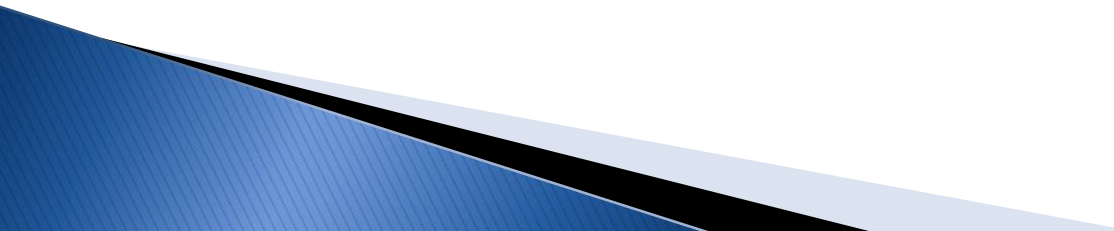
Prerequisites

- ▶ CSE30: Data Structures
- ▶ Math: logarithms, series, boolean logic, matrices, calculus ...
- ▶ Coding: intermediate programming experience (Java, C, C++, ...)
 - Coding in terminals??
- ▶ Curiosity: observe how the world is run by computers, and what problems we face.

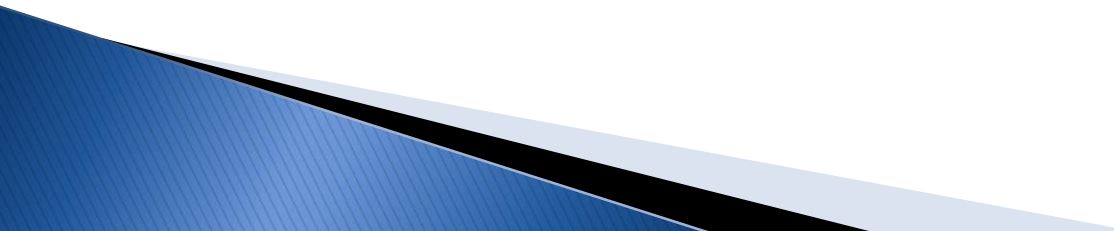
Grading

- ▶ Homework: 15%
- ▶ Projects: 15%
- ▶ Lab assignments: 15%
- ▶ Mid-terms: 30%
- ▶ Final exam (comprehensive): 15%
- ▶ Participation (reading & labs): 10%
- ▶ Grades:
 - 90% of points at least an A
 - 80% at least a B
 - 70% at least a C

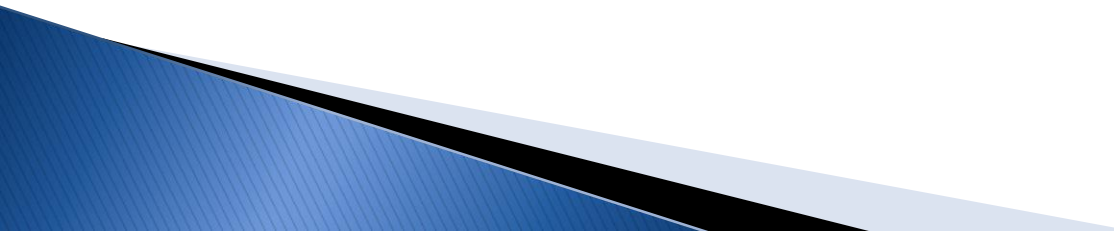
Hints for success

- ▶ Attend lecture
 - ▶ Read the textbook and do the activities
 - ▶ Do & understand the labs and homework **YOURSELF**
 - ▶ Create a portfolio to save all your work
 - ▶ Take notes while reading and in lecture
 - ▶ Ask questions
- 

Policies

- ▶ Don't copy someone else's code
 - ▶ Don't give your code away
 - ▶ Don't outsource your assignments
 - ▶ Don't use electronic devices in exams
 - ▶ Don't use electronic devices during lecture for purposes other than note taking
 - ▶ Turn off speakers/cellphone during class
- 

No Cheating!

- ▶ Communicating information to another student during examination.
 - ▶ Knowingly allowing another student to copy one's work.
 - ▶ Offering another person's work as one's own.
 - ▶ I am serious!
- 

History Lesson on C

- ▶ C developed by Dennis Ritchie at AT&T Bell Labs in the 1970s.
 - Used to maintain UNIX systems
 - C was derived from the B language
 - B was derived from the BCPL (Basic Combined Programming Language)
 - Many commercial applications are still written in C
- ▶ Current standard updates
 - C11: improved Unicode support, cross-platform multi-threading API
 - C99 or C9x remains the common standard

History Lesson on C

▶ References

- <http://en.wikipedia.org/wiki/C99>

▶ Highlights

- Declarations in for loops, like Java
- Java-like `//` comments (to end of line)
- Variable-length non-global arrays
- `<inttypes.h>`: explicit integer types (`intN_t`, `uintN_t`)
- `<stdbool.h>` for boolean logic def's

▶ Current version is C18

Disclaimer

- ▶ **Important:** You will not learn how to fully code in C in these lectures! You'll still need your C reference for this course:
 - K&R is a must-have reference
 - Check online for more sources

Compilation : Overview

C compilers take C and convert it into an **architecture specific** machine code (string of 1s and 0s).

- Unlike Java which converts to **architecture independent** bytecode.
- Unlike most functional programming languages (ex. Scheme) which interpret the code.
- These differ mainly in **when** your program is converted to machine instructions.
- For C, generally a 2 part process of compiling .c files to .o files, then linking the .o files into executables. Assembling is also done (but is hidden, i.e., done automatically, by default)

Compilation : Advantages

- ▶ **Great run-time performance**: generally much faster than Scheme or Java for comparable code (because it optimizes for a given architecture)
- ▶ **OK compilation time**: enhancements in compilation procedure (`Makefiles`) allow only modified files to be recompiled

Compilation : Disadvantages

- ▶ All compiled files (including the executable) are **architecture specific**, depending on both the CPU type and the operating system
- ▶ Executable must be **rebuilt** on each new system.
 - Called “**porting your code**” to a new architecture.
- ▶ The “change→compile→run [repeat]” iteration cycle is slow

C vs. Java™ Overview (1/2)

Java	C
Object-oriented (OOP)	No built-in object abstraction. Data separate from methods.
“Methods”	“Functions”
Class libraries of data structures	C libraries are lower-level
Automatic memory management	Manual memory management

C vs. Java™ Overview (1/2)

Java	C
High memory overhead from class libraries	Low memory overhead
Relatively Slow	Relatively Fast
Arrays initialize to zero	Arrays initialize to garbage
Syntax: <code>/* comment */</code> <code>// comment</code> <code>System.out.print</code>	Syntax: <code>/* comment */</code> <code>// comment</code> <code>printf</code>

You need newer C compilers to allow Java style comments, or just use C99

C Syntax: `main`

- ▶ To get the main function to accept arguments, use this:
`int main (int argc, char *argv[])`
- ▶ What does this mean?
 - **argc** will contain the number of strings on the command line (the executable counts as one, plus one for each argument). Here `argc` is 2:
`./sort myFile`
 - **argv** is a pointer to an array containing the arguments as strings (more on pointers later).
 - Always return a value according to ANSI (American National Standard Institute)

C Syntax: Variable Declarations

- ▶ Very similar to Java, but with a few minor but important differences
- ▶ All variable declarations must go before they are used (at the beginning of the block)*
- ▶ A variable may be initialized in its declaration; if not, it holds garbage!
- ▶ Examples of declarations:
 - correct: `int a = 0, b = 10;`
 - ...
 - Incorrect:* `for (int i = 0; i < 10; i++)`

*C99 overcomes these limitations

C Syntax: True or False?

- ▶ What evaluates to FALSE in C?
 - 0 (integer)
 - NULL (pointer: more on this later)
 - no such thing as a Boolean*
- ▶ What evaluates to TRUE in C?
 - everything else...

Boolean types provided by C99's `stdbool.h`

C syntax : flow control

- ▶ Within a function, remarkably **close to Java** constructs in methods (shows its legacy) in terms of flow control
 - `if-else`
 - `switch`
 - `while` **and** `for`
 - `do-while`

Common C Error

`a = b` VS `a == b`

- ▶ There is a difference between assignment and equality

`a = b` is assignment

`a == b` is an equality test

- ▶ This is one of the most common errors for beginning programmers!

- One solution (when comparing with constant) is to put the var on the right!

If you happen to use `=`, it won't compile.

```
if (3 == a) { ... }
```


All objects have a size

- ▶ The size of their representation
- ▶ The size of static objects is given by sizeof operator (in Bytes)

```
#include <stdio.h>
int main() {
    char c = 'a';
    int x = 34;
    int y[4];
    printf("sizeof(c)=%d\n", sizeof(c) );
    printf("sizeof(char)=%d\n", sizeof(char));
    printf("sizeof(x)=%d\n", sizeof(x) );
    printf("sizeof(int)=%d\n", sizeof(int) );
    printf("sizeof(y)=%d\n", sizeof(y) );
    printf("sizeof(7)=%d\n", sizeof(7) );
}
```

Output:

```
sizeof(c)=1
sizeof(char)=1
sizeof(x)=4
sizeof(int)=4
sizeof(y)=16
sizeof(7)=4
```

Quiz:

```
void main() ; {  
    int *p, x=5, y; // init  
    y = *(p = &x) + 1;  
    int z;  
    flip-sign(p);  
    printf("x=%d, y=%d, p=%d\n", x, y, p);  
}  
flip-sign(int *n) { *n = -(*n) }
```

How many syntax+logic errors in this C99 code?

- | #Errors |
|---------|
| a) 1 |
| b) 2 |
| c) 3 |
| d) 4 |
| e) 5 |

Quiz: Answer

```
void main() ; {  
    int *p, x=5, y; // init  
    y = *(p = &x) + 1;  
    int z;  
    flip-sign(p) ;  
    printf("x=%d, y=%d, p=%d\n", x, y, *p) ;  
}  
flip-sign(int *n) { *n = -(*n) ; }
```

How many syntax+logic
errors in this C99 code?

5...

(signed ptr print is logical err)

#Errors
a) 1
b) 2
c) 3
d) 4
e) 5

Announcement

- ▶ Lab #1 starts next week
 - Due in one week
- ▶ Reading assignment
 - Chapter 4-6 of K&R (C book) to review on C/C++ programming