

CSE 495.21: Programming in C

Instructor: Weikuan Yu
Email: yuw@cse.ohio-state.edu
Phone: 614-292-8458
Office: DL 744 (Dreese Laboratories)
Office Hour: Friday 2:30–3:30PM

Course Information:

- **Course Page:**
 - <http://www.cse.ohio-state.edu/~yuw/459-21>
- **Prerequisite:**
 - cse321 or equivalent
- **Textbook:**
 - *The C Programming Language*, Kernighan & Ritchie
- **Newsgroup:**
 - cis.course.cis45921,
 - the main forum of class communication
 - Office hour stop-by or appointments

Syllabus:

- **Lectures:**
 - Five lectures covering Ch. 1-8,
 - Ch. 2-3, read on your own
 - Occasional pop quizzes
- **Labs:**
 - Four labs, 20% each
 - Electronic submission
 - run on stdsun.cis.ohio-state.edu
 - lab quality, correctness
 - Additional emphasis on Coding style and Makefile
- **Exams:**
 - None, but all others count

A little history:

- 1970's
 - Unix
 - C, from BCPL (Thompson and Ritchie)
- C programming Language
 - Widely used like the others: Fortran, Pascal
 - Main form of language for system programming
 - Available on any machine with C compiler and library

Some Characteristics:

- **Scope:**
 - Intended:
 - HPC, OS, general programming
 - Typically, long developing cycle
 - very platform dependent
 - Not intended: web, scripting, data processing
- **Features:**
 - Close interaction with system
 - Standard library (user-level only),
 - no other fancy stuff: networking, memory, graphics
 - Extensive use of pointers
 - Procedure (OOB) programming, strictly pass by value

Sample 1:

```
/* Hello World! */
#include <stdio.h>

int main()
{
    printf("Hello World!\n");
    return 0;
}
```

Walk through:

- C program: hello.c
 - emacs, vi, vim, pico, joe ...
 - But text editors only. No word processor
- Preprocessing: hello.s, assembly code
 - cc -S hello.c
- Compilation: hello.o, a binary file
 - cc -c hello.s
- Linking: a.out or hello, an executable file
 - cc hello.o
 - cc -o hello hello.o
- Loading (dynamical linking) and execution: ./hello
 - ./a.out
 - ./hello

Programming (Editing)

- C program: hello.c
 - emacs, vi, vim, pico, joe ...
 - But text editors only. No word processor

Compile:

- Preprocessing: hello.s, assembly code
 - cc -S hello.c
 - Can you take a look at hello.s and compare to hello.c?
- Compilation: hello.o, a binary file
 - cc -c hello.s
 - Do you know how to take a look at hello.o?
- Linking: a.out or hello, an executable file
 - cc hello.o
 - cc -o hello hello.o
 - Want to look at the executable, hello?

Run:

- Loading (dynamical linking) and execute: ./hello
 - ./a.out
 - ./hello
- What you do with java programs?
- What computer does to run your program?

Sample 1 Dissection:

- What it has?
 - A function definition: main
 - A line of comment
 - A line of preprocessor directive
 - An output statement
 - A return clause
- What it does?
 - Ask the computer to say hello to the world.
- What it does not do?
 - It seems not computing!!
 - No real work
 - not taking input
 - does not change any value

Sample 2:

```
#include <stdio.h>
#define MAGIC 10
int main(void)
{
    int i, fact, quotient;
    while (i++ < 3) {
        printf("Guess a factor of MAGIC larger than 1: ");
        scanf("%d", &fact);
        quotient = MAGIC % fact;
        if (0 == quotient)
            printf("You got it!\n");
        else
            printf("Sorry, You missed it!\n");
    }
    return 0;
}
```

Sample 2 Dissection:

- What more it has?
 - Macro definition
 - Variable declaration
 - Operations represented by operators
 - Conditional computation
 - Input Processing
 - Loops: three chances

Syntax Dissection:

- What is syntax?
 - m-w.com:
 - the way in which linguistic elements (as words) are put together to form constituents (as phrases or clauses)
 - How to put your C elements together into a program?
- Variable declaration,
 - Identifier: case, keywords, style
 - datatype specification
 - char, short, int, long, float, double
- Operators:
 - +, -, *, /, %, ++, --, >, <, ==, :, &, |, ...
 - Priority and Precedence
 - Associativity

More Syntax Dissection:

- Preprocessor directive
 - macro
 - header file inclusion
- Loop constructs
 - for, while, do-while, break, continue, label--go;
- Conditional statements
 - If, else, else if, switch, break
- Function definition
 - return type, parameter definition, parameter passing
 - Standard library and function calls
- Take home practice:
 - Walk through the following steps with sample 2
 - preprocessing, compilation, linking, loading/execute
 - Modify and repeat once more if you can