601.220 Intermediate Programming

Spring 2023, Day 2 (Jan 25th)

Welcome!

Today's agenda:

- get started with C
- Exercise 2

Announcements

- HW0 is due on Friday, Feb 6th

Goals for today

- By the end of class today, you should
  - Be able to access your ugrad account
  - Have written and executed a C program
  - Accepted the email invitation to join the `jhu-ip` Github organization (and have access to your private repo?)

Let me know if you didn't receive an invitation via email

Day 2 recap questions

1. The command to compile a C program is `gcc <source file> -std=c99 -pedantic -Wall -Wextra`. Use man or Google to find out the meaning of the four flags, i.e. `-std=c99`, `-pedantic`, `-Wall` and `-Wextra`.
2. Briefly describe what a preprocessor, compiler and linker do when transporting C code into executable?
3. What does an undefined behavior mean in programming? Do we need to care about it? Why or why not?
4. What does the modifier const mean?
5. What are the primitive types in C and what are their byte sizes?
6. What is the value of `7 / 2` (a division of two integers) in a C program?

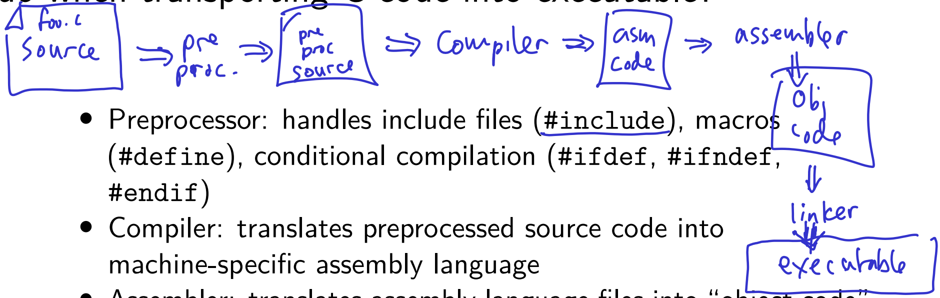
1. The command to compile a C program is gcc <source file> -std=c99 -pedantic -Wall -Wextra. Use man or Google to find out the meaning of the four flags, i.e. -std=c99, -pedantic, -Wall and -Wextra.

c11

- -std=c99: Use the C99 version of the C language
- -pedantic: Strictly adhere to the language specification
- -Wall: enable (almost) all warnings
- -Wextra: enable extra compiler warnings

goal: NO warnings

2. Briefly describe what a preprocessor, compiler and linker do when transporting C code into executable?

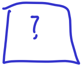


- Preprocessor: handles include files (`#include`), macros (`#define`), conditional compilation (`#ifdef`, `#ifndef`, `#endif`)
- Compiler: translates preprocessed source code into machine-specific assembly language
- Assembler: translates assembly language files into “object code” (machine language)
- Linker: joins object files together into an executable

3. What does an undefined behavior mean in programming? Do we need to care about it? Why or why not?

Example:

```
#include <stdio.h>
int main(void) {
  int x;= 0;
  printf("%d\n", x);
  return 0;
}
```



Undefined behavior means that the behavior of the program can't be predicted. Programs with undefined behavior can't be relied on to do anything useful!

4. What does the modifier `const` mean?

`const` means “read-only”.

E.g.:

```
const float PI = 3.14159;

PI = 3.0; // not allowed, compile error
```

5. What are the primitive types in C and what are their byte sizes?

Data type	Typical size in bytes
`char`	1
`int`	4
`long`	8
`float`	4
`double`	8

Note that C mandates a minimum range of values for each data type, but in practice that range could be larger. For example, `int` is guaranteed to allow a range of at least $-32,768$ to $32,767$ (i.e., 2 bytes), but supports a much larger range on most modern systems.

6. What is the value of `7 / 2` (a division of two integers) in a C program?

```
float f;
int a = ... , b = ... ;
✗ f = a / b;
```

`7 / 2` = `3`. This is because `7` and `2` are both integer (`int`) values, and a division of two integer values is an *integer division* where

- the result is an integer, and
- the fraction is discarded

Another example: `19 / 4` = `4`

The C language

- The first half of the course will focus on programming in C
- It is a *low-level*, “systems” programming language
  - Very close to the machine
  - Directly exposes machine-level concepts like
    - hardware-supported numeric data types
    - memory addresses

Hello world in C

```c
// hello_world.c:

#include <stdio.h>

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

Compiling and running the program:

```
$ gcc hello_world.c -std=c99 -pedantic -Wall -Wextra
$ ./a.out
Hello, world!
```

How to try this out on ugrad?

Use `ssh` (or PuTTY) to log into your ugrad account.

Use `mkdir` to create a directory to put your code in.

Use `nano` to edit the source file. (By Friday you will know how to use a better editor, `emacs`.)

Use `gcc` to compile the source code into an executable.

Run the executable.

Reading input, computation, printing a computed value

```c
// add.c:
#include <stdio.h>

int main(void) {
  int a, b, sum;
  printf("Enter two integers: ");
  scanf("%d", &a);
  scanf("%d", &b);
  sum = a + b;
  printf("Sum is %d\n", sum);
  return 0;
}
```

Compiling and running the program:

```
$ gcc add.c -std=c99 -pedantic -Wall -Wextra
$ echo "2 3" | ./a.out
Enter two integers: Sum is 5
```

Some C numeric data types

Data type	Description
`char`	Character data type, typical range $-128 \ldots 127$
`int`	Integers, typical range $-2^{31} \ldots 2^{31} - 1$
`long`	Integers, typical range $-2^{63} \ldots 2^{63} - 1$
`float`	Floating point (approximate real number), 32 bit
`double`	Floating point, 64 bit

printf and scanf placeholders

Use these in `printf` and `scanf` format strings to designate output values (`printf`) or variables in which to store input values (`scanf`)

Data type	Placeholder
`char`	`%c`
`int`	`%d`
`long`	`%ld`
`float`	`%f`
`double`	`%lf`

In-class activity

Exercise 2: practice programming in C using the Online C compiler.

If you finish and want to continue: try editing, compiling, and running a program using your ugrad account.

Notes

Notes

Notes

Notes

Notes