

---

## CS246 C and Unix Programming

---

- Today's goals
  - History of C
  - Basic types
  - `printf`
  - Arithmetic operations, types and casting
  - Intro to linux

---

CS2461Lec02

---

- Section 1

## The C Language

---

- Currently one of the most commonly-used programming languages
- “High-level assembly”
- Small, terse but powerful
- Very portable: compiler exists for virtually every processor
- Produces efficient code
- It is at once loved and hated


---

CS2462Lec02

---

## History of C

---

- Developed during 1969-73 in the bell labs
- C is a by product of **Unix**
- C is mostly credited to  Dennis Ritchie
- Evolved from B, which evolved from BCPL

---


CS2463Lec02

---

## History of C

---

- Original machine (DEC PDP-11) was very small
  - 24k bytes of memory,
  - 12k used for operating systems
- When I say small, I mean memory size, not actual size.




---

CS2464Lec02

---

## Why is C Dangerous

---

- C's small, unambitious feature set is an advantage and disadvantage
- The price of C's flexibility
- C does not, in general, try to protect a programmer from his/her mistakes
- The International Obfuscated C Code Contest's (<http://www.ioccc.org/>) 1995 [winning entry](#)

---

CS2465Lec02

---

- Section 2

## Programming Process

---

- Source code must carry extension `.c`
- But may be named with any valid Unix file name
  - Example: `01-helloworld.c`

Lec #	description	C program
} Example program filename convention in this course		

---

CS2466Lec02

### Example

```

/* helloworld.c, Dianna Xu
   Displays a message */
#include <stdio.h>

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

helloworld.c

---

CS246 7 Lec02


### Hello World in C

```

#include <stdio.h>

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

Preprocessor used to share information among source files  
Similar to Java's import




---

CS246 8 Lec02

### Hello World in C

```

#include <stdio.h>

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

Program mostly a collection of functions  
"main" function special: the entry point  
"int" qualifier indicates function returns an integer  
I/O performed by a library function

---

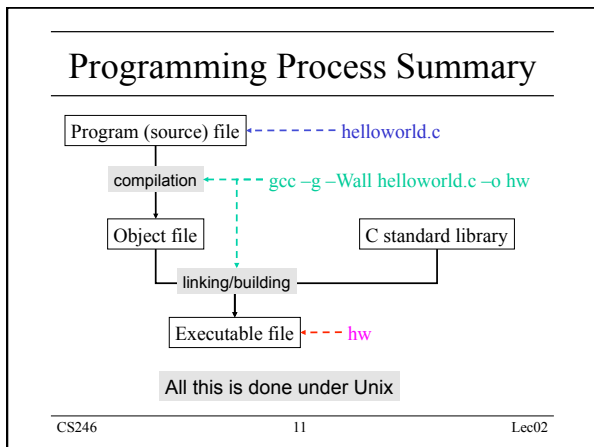
CS246 9 Lec02

### The Compiler

- gcc (Gnu C Compiler)
- gcc -g -Wall helloworld.c -o hw
- gcc flags
  - -g (produce debugging info for gdb)
  - -Wall (print warnings for all events)
  - -o filename (name output file with filename, default is a.out)

---

CS246 10 Lec02



### C Program Style

- Case sensitive
- Ignores blanks
- Comments
  1. Ignored between /\* and \*/
  2. Comments are integral to good programming!
- All local variables must be declared in the beginning of a function !!!

---

CS246 12 Lec02

- Section 3

## Data Types

- Integer
  - C keyword: **int, short, long**
  - Range: typically 32-bit (±2 billion), 16-bit, 64-bit
- Floating-point number
  - C keyword: **float, double** In general, use double
  - Range: 32-bit (± 10<sup>38</sup>), 64-bit
  - Examples: **0.67f, 123.45f, 1.2E-6f, 0.67, 123.45, 1.2E-6**

---

CS246 13 Lec02

## Variables and Basic Operations

- Declaration (identify variables and type)
 

```
int x;
int y, z;
```
- Assignment (value setting)
 

```
x = 1;
y = value-returning-expression;
```
- Reference (value retrieval)
 

```
y = x * 2;
```

---

CS246 14 Lec02

## Constants

- Integer
  - **const** int year = 2002;
- Floating point number
  - **const** double pi = 3.14159265;
- Constants are variables whose initial value can not be changed.
- Comparable to **static final**

---

CS246 15 Lec02

- Section 4

## Output Functions

- Output characters
 

```
printf("Text message\n");
```

\n for new line
- Output an integer
 

```
int x = 100;
printf("Value = %d\n", x);
```

Output: Value = 100

---

CS246 16 Lec02

## Variations

- Output a floating-point number
 

```
double y = 1.23;
printf("Value = %f\n", y);
```
- Output multiple numbers
 

```
int x = 100;
double y = 1.23;
printf("x = %d, y = %f\n", x, y);
```

15 digits below decimal (excluding trailing 0's)

Output: x = 100, y = 1.230000

---

CS246 17 Lec02

## printf Summary

```
printf(" [red box] ", [blue box] );
```

- Text containing special symbols
  - **%d** for an integer
  - **%f** for a floating-point number
  - **\n** for a newline
- List of variables (or expressions)
  - In the order corresponding to the % sequence

---

CS246 18 Lec02

### Display Problem

- Problem
  - Precision of **double**: 15 digits
  - Precision of **%f**: 6 digits below decimal
  - Cannot show all the significant digits
- Solution
  - More flexible display format possible with **printf**

---

CS24619Lec02

### % Specification

- **%i** **int, char** (to show value)
- **%d** same as above (**d** for decimal)
- **%f** **double** (floating-point)
- **%e** **double** (exponential, e.g., **1.5e3**)

---

CS24620Lec02

### Formatting

- Precision **%.#f**
- Width **##f, ##d**
  - Note: Entire width Replace # with digit(s)
- Zero-padding **%0#d**
- Left-justification **%-#d**
- Various combinations of the above

---

CS24621Lec02

### Formatting Example (1)

```

%f with 1.23456789 >1.234568<
%.10f with 1.23456789 >1.2345678900<
%.2f with 1.23456789 >1.23<

%d with 12345 >12345<
%10d with 12345 >      12345<
%2d with 12345 >12345<

%f with 1.23456789 >1.234568<
%.8.2f with 1.23456789 >      1.23<
```

---

CS24622Lec02

### Formatting Example (2)

```

%d:%d with 1 and 5 >1:5<
%02d:%02d with 1 and 5 >01:05<

%10d with 12345 >      12345<
%-10d with 12345 >12345      <

```

11-formatting.c

---

CS24623Lec02

- Section 5

### Arithmetic Operators

- Unary: **+**, **-** (signs)
- Binary: **+**, **-**, **\*** (multiplication), **/** (division), **%** (modulus, int remainder)
- Parentheses: **(** and **)** must always match.
  - Good: **(x)**, **(x - (y - 1)) % 2**
  - Bad: **(x, )x(**

---

CS24624Lec02

## Types and Casting

- Choose types carefully
- An arithmetic operation requires that the two values are of the same type
- For an expression that involves two different types, the compiler will **cast** the smaller type to the larger type
- Example:  $4 * 1.5 = 6.0$

CS246 25 Lec02

## Mixing Data Types

- **int** values only  $\Rightarrow$  **int**
  - $4 / 2 \Rightarrow 2$
  - $3 / 2 \Rightarrow 1$
  - **int**  $x = 3, y = 2;$   
 $x / y \Rightarrow 1$
- Involving a **double** value  $\Rightarrow$  **double**
  - $3.0 / 2 \Rightarrow 1.5$

CS246 26 Lec02

## Assignment of Values

- **int**  $x;$ 
  - $x = 1;$
  - $x = 1.5; /* x is 1 */$  warning
- **double**  $y;$ 
  - $y = 1; /* y is 1.0 */$
  - $y = 1.5;$
  - $y = 3 / 2; /* y is 1.0 */$

int evaluation; warning

CS246 27 Lec02

## Example

```

int i, j, k, l;
double f;

i = 3;
j = 2;
k = i / j;
printf("k = %d\n", k);

f = 1.5;
l = f;
printf("l = %d\n", l); /* warning */ /* truncated */
    
```

mixingtypes.c

CS246 28 Lec02

Section 6

## Essentials of Linux

- Kernel
  - Allocates time and memory, handles file operations and system calls.
- Shell
  - An interface between the user and the kernel.
- File System
  - Groups all files together in a hierarchical tree structure.

CS246 29 Lec02

## Summary

- Learn how to use Unix, emacs and gcc
  - Lab
  - Expect that ‘weird’ things happen; Don’t panic
  - Remember that you can “man” all C functions
- Learn the difference and similarity btw Java/Python and C syntax
- Programming style is important.
  - Speed up your programming and debugging process

CS246 30 Lec02