Today

- C programming

Review

- How was your first Bash scripting experiment?

CSCI210 Review

- What do you remember about C?

C Overview

- **Compiled, statically typed**
- **Data types:** int, char, float, double (short, long, signed, unsigned)
  - What's missing?
- **Arithmetic**: +, -, *, /, %
- **Printf:** # of args determined by format string http://www.cprogramming.com/tutorial/printf-format-strings.html
- **Loops & conditionals:** same as Java
  - But no "for each" loop

C review – 4 data types

```
/*
 * A review of the basic data types in C.
 */

#include <stdio.h>

int main() {
        int x, y;
        char a;
        float f, e;
        double d;

        x = 4;
        y = 7;
        a = 'H';
        f = -3.4;
        d = 54.123456789;
        e = 54.123456789;

        printf("%d %c %f %lf\n", x, a, e, d);
        printf("%d %c %.9f %.9lf\n", x, a, e, d);
}
```


C review – arithmetic

Printf ignores data not matching format specifier

```
/* A review of the basic arithmetic operators in C.  */

#include <stdio.h>

int main() {
        int x, y;
        int r1, r2, r3, r4, r5;

        x = 4;
        y = 7;
        r1 = x + y;
        r2 = x - y;
        r3 = x / y;
        r4 = x * y;
        printf("%d %d %d %d\n", r1, r2, r3, r4);

        r3++;
        r4--;
        r5 = r4 % r1;
        printf("%d %d %d\n", r3, r4, r5);
}
```

C PROGRAM ORGANIZATION & DEVELOPMENT

Using Source and Header files

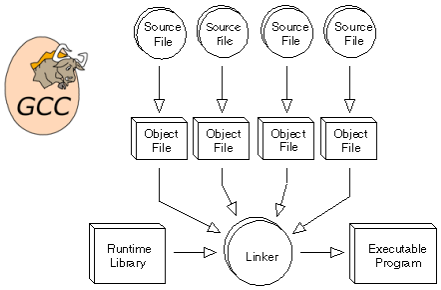
- Place related code within the same module (i.e., file)
- Header files (`*.h`)
  - function prototypes, data types, macros, inline functions and other common declarations
- Do not place source code (i.e., definitions) in the header file with a few exceptions.
  - Inline'd code
  - class definitions
  - const definitions
- *C preprocessor* (cpp) is used to insert common definitions into source files

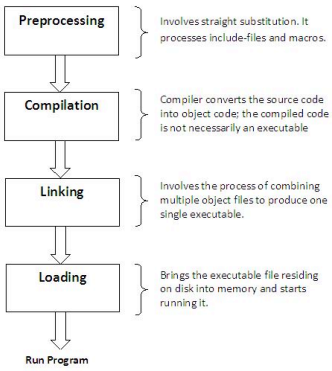
The Preprocessor

- The C preprocessor permits you to define simple ***macros*** that are evaluated and expanded prior to compilation.
- Commands begin with a '#'. Abbreviated list:
  - `#define` : defines a macro
  - `#undef` : removes a macro definition
  - `#include` : insert text from file
  - `#if` : conditional based on value of expression
  - `#ifdef` : conditional based on whether macro defined
  - `#ifndef` : conditional based on whether macro is not defined
  - `#else` : alternative
  - `#elif` : conditional alternative
  - `defined()` : preprocessor function: 1 if name defined, else 0

```
#if defined(__NetBSD__)
```


C COMPILATION OVERVIEW





C arrays, strings, pointers, bits

THE TRICKY BITS

C Advanced Data Constructs

- **Array**: list of same-type values
- **String**: character array of text
- **Pointer**: address of another variable
- **Structure**: group of mixed-type values
  - Precursor to objects
  - Structs are like classes with fields but no methods

Arrays

- Syntax similar to Java:

```
int a[3];
int b[2];
a[0] = 1;
a[1] = -1;
a[2] = 2;
b[0] = 4;
printf("%d %d %d\n", a[0], a[1], a[2]);
```

- But what happens when we execute this statement?
  - `printf("%d\n", a[4]);`

Arrays in C aren't *safe*

- Array index out of bounds may or may not result in an error:
  `printf("%d\n", a[4]);`
- Prints: 4
- Be ***very*** careful when directly indexing array elements

Label	Value
a[0]	1
a[1]	-1
a[2]	2
∅	\0
b[0]	4
b[1]	

Strings, aka character arrays

- Example:
  char a[6];
  a[0] = 'H';
  a[1] = 'i';
  a[2] = '!';

Label	Value
a[0]	H
a[1]	i
a[2]	!
a[3]	\0
a[4]	
a[5]	

- String processing methods will stop when the string delimiter, '\0', is reached

String Library Functions

- **strlen**: returns the number of characters in a string (before the delimiter)
- **strcmp**: returns whether 2 strings:
  - 0: are identical
  - -1: first string smaller
  - 1: second string smaller
- **strcpy**(dest, src): copies from src to dest
- **strcat**(orig, append): append a string to orig
- **sprintf**(a, "",?): store printf output to string a

Char by char string processing

```c
#include <stdio.h>
#include <string.h>

main() {
    int i, j;
    char s[6];

    s[0] = 'a';
    s[1] = 'b';
    s[2] = 'a';
    s[3] = 'c';
    s[4] = 0;
    i = 0;
    j = 0;
    while (s[i] != 0) {
        if (s[i] != 'a') {
            s[j] = s[i];
            j++;
        }
        i++;
    }
    s[j] = 0;
    printf("%s\n", s);
}
```


Next Time

- More C
- You: review C code