Today

- Reviewing C programming
  - Basics
  - Compilation
  - Character arrays
  - Pointers

Review

- What is the Unix philosophy?
  - What are the tradeoffs of that philosophy?
- What was your favorite Unix command?
- What is shell scripting?
- What are two ways to execute a shell script?
- What does a Bash script look like?

Review: Unix Design

- Small, simple programs
  - Easier to maintain
  - Single-responsibility principle
- Combine (a few or lots) with pipes
  - Easy to combine with a simple interface |
- Not-so-user-friendly to get started

What Does This Script Do?

We did not talk about everything in here, but you can figure a lot out

- Break into pieces
- Use appropriate terminology
- What do you know?
- What can you figure out?
  - Fill in the gaps

```
#!/bin/bash
# Usage: `basename $0` <whichassign>

TURNINDIR=/csdept/courses/cs330/turnin
LAB=$1

for STUDENT in `ls $TURNINDIR`
do
    if [ -e $TURNINDIR/$STUDENT ]; then

        if [ -d $TURNINDIR/$STUDENT/$LAB ]; then
            echo "--------------$STUDENT----------------"
            ls -l $TURNINDIR/$STUDENT/$LAB
        else
            echo "** $STUDENT has not turned in $LAB"
        fi
    fi
done
```

What Does This Script Do?

shebang

```
#!/bin/bash
# Usage: `basename $0` <whichassign>

TURNINDIR=/csdept/courses/cs330/turnin
LAB=$1

for STUDENT in `ls $TURNINDIR`
do
    if [ -e $TURNINDIR/$STUDENT ]; then
        if [ -d $TURNINDIR/$STUDENT/$LAB ]; then
            echo "--------------$STUDENT----------"
            ls -l $TURNINDIR/$STUDENT/$LAB
        else
            echo "** $STUDENT has not turned in $LAB"
        fi
    fi
done
```

Parameter 0 is the executing program

Parameter 1 is the first command line argument

Iterate through the contents of the turnin directory

Does the student directory exist?

Is the directory for the assignment there?

List the contents of the directory

Display error message

C

CSCI210 Review

- What do you remember about C?
  - What are the programming language characteristics of C?
  - How do you execute programs in 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
- (Most) code should be in functions

C review – Data Types

```c
/*
 * 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 %.3lf\n", x, a, e, d);
}
```

What is the output of this program?

datatypes_and_print.c

C review – Data Types

```c
/*
 * 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 %.3lf\n", x, a, e, d);
}
```

54.123456789 is too much precision for float to handle

datatypes_and_print.c

C review – arithmetic

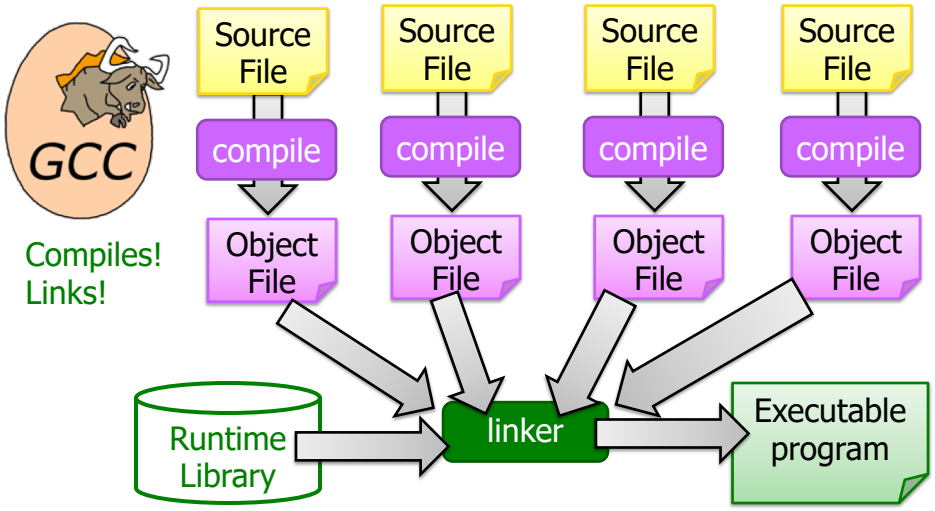
`printf` ignores data not matching format specifier

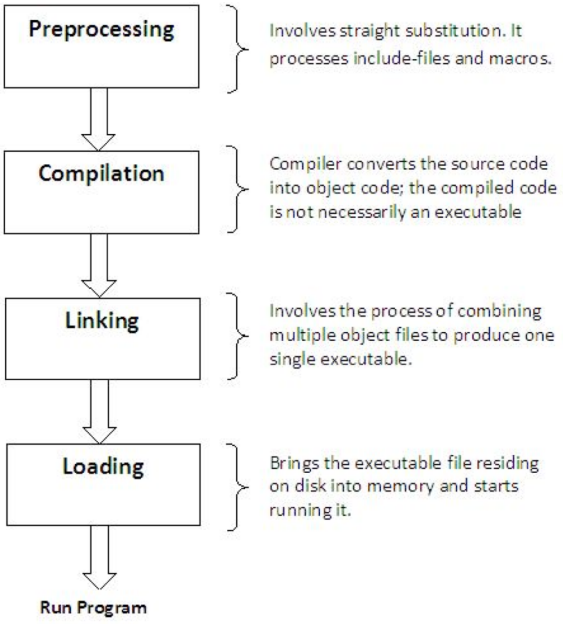
```c
/* A review of the basic arithmetic operators in C.  */

#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);
}
```

arithmetic.c

C COMPILATION OVERVIEW

Compilation Overview





Compiling with gcc

- `gcc –o <executable> <filename>.c`
  - -o means "output file"
- If don't include the –o option, default executable file is `a.out`

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 MAX 10`
  - `#define` : defines a macro `#define SQUARE(x) (x*x)`
  - `#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 arrays, strings, …

THE TRICKY BITS

C Advanced Data Constructs

- **Array**: fixed-size 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.c

Arrays in C are not *safe*

- Array index out of bounds may or may not result in an error:

```
printf("%d\n", a[4]);
```

- Prints 4, in case to right)
  - But, who knows what value is there
- Be ***very*** careful when directly indexing array elements

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

Strings, a.k.a. character arrays

- Example:

```
char a[6];
a[0] = 'H';
a[1] = 'i';
a[2] = '!';
a[3] = '\0';
```

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

Make this explicit; don't rely on the memory being 0'd out. (Why isn't the memory necessarily zero'd out?)

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

 string.c

Strings, a.k.a. character arrays

- Equivalent:
  - '\0'
  - 0
  - NULL
- Can use any of those to mark the end of a string
  - I will tend to use the first

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

Char by Char String Processing

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

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

    s[0] = 'a';
    s[1] = 'b';
    s[2] = 'a';
    s[3] = 'c';
    s[4] = '\0';
    printf("%s\n", s);
    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);
}
```

- What is the output of this program?
- What are more descriptive names for i and j?

charbychar.c

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

Looking Ahead

- Bash assignment due Friday before class
- Friday
  - C
    - More pointers
    - Command-line arguments
    - Structs
  - Make