CS 222: Overview of C

Chris Kauffman

Week 1-2

Logistics

Office Hours

- Tue/Thue 3:00-4:00 pm
- Before class
- By appointment: let me know if you can't make it before class
- Anyone hosed?

HW 1

- Due next week Tuesday by 11:59 on Blackboard
- Tour in a moment
- Try them over the weekend
- Any questions now?

Reading

- Schedule Here
- Zyante 1 & 2 this week
- Zyante 3-6 next week

Query

How many of you are taking a summer A Session course (ends in a couple weeks)?

Card-worthy Review: Setup, Shell, Compiler

- How do you move from one directory/folder to another in the unix shell?
- Where and how do you write a C program?
- How do you compile the program you wrote?
- How do you run the program once compiled?
- Can you run the program without compiling it?
- Can you read the compiled program?

Card-worthy Review: C Comments and Variables

- What are two ways to write comments in C?
- What's a really useful commenting technique?
- Describe a numeric type that C uses and what it calls such numbers
- Describe the another numeric type; what's the difference?
- What's another kind of variable C uses?
- What type name represents "nothing"?

Tour of HW 1

http://www.cs.gmu.edu/~kauffman/cs222/hw1.html

- 4 problems
  1. Debug
  2. Real/Integer Division
  3. Coin Counting
  4. I/O and non-trivial calculation
- Exercises your ability to do...
  - Basic I/O
  - Linking against math library
  - Variables/expressions
  - Algorithmic thinking
- Important
  - Name your directory right
  - Include `ID.txt`
  - Use the provided test script:

Goals

- Assignment and expressions
- Basic I/O
- Practice

Every Programming Language

Look for the following

- ☒ Comments
- □ Statements/Expressions
- ☒ Variable Types
- ⊟ Assignment
- ⊟ Basic Input/Output
- □ Function Declarations
- □ Conditionals (if-else)
- □ Iteration (loops)
- □ Aggregate data (arrays, structs, objects, etc)
- □ Library System

First Textbook Program: salary.c

```c
/* From Zyante Programming C Ch 2.15 w/ modifications
   Calculate age in days based on input, assumes 365 day years

   compile:     gcc age.c
   run on mac:  a.out
   run on win:  a.exe
*/

#include <stdio.h>

int main(void) {
   int userAgeYears = 0;
   printf("Enter your age in years: \n");
   scanf("%d", &userAgeYears);

   // Declare anywhere
   int userAgeDays = userAgeYears * 365;
   printf("You are %d days old.\n", userAgeDays);
   return 0;
}
```

Declare then Use

Must *declare* variables before using and give them a *type*

Right

```
int main(){
  int x = 4;
}
```

```
int main(){
  int x;
  x = 4;
}
```

Wrong

```
int main(){
  x = 4;
}
```

```
int main(){
  x = 4;
  int x;
}
```

Historical Note

Old C

```
int main(){
  int x, y;
  double d;

  x = 4;
  y = x + 2;
  d = 12.34;
}
```

New C (C99/C11)

```
int main(){
  int x;
  x = 4;
  int y = x + 2;

  double d;
  d = 12.34;
}
```

A Lesson

All languages change

- New words enter English (e.g. *truthiness*, *selfie*)

New ideas enter PLs

- C is changing
- Very slowly compared to other PLs

Gotchya: not every compiler translates C to machine language the same way

- May not support the latest lingo (C11)
- Use our environment so that you are compatible

Statements/Expressions - Do Something

- Assignment is very common use ‘equals sign’

  `x = 5;`
- End with a semicolon: ;
- Most frequent error is forgetting ;

Follow the integer arithmetic below

```
int main(){
  int x, int y = 5;
  x = y * 2 + 1;
  x = (y * 2) + 1;
  x = y * (2 - 1);
  x = x * x + y - 1;
  x = y / 2;
  x = y % 2;                            /* ??? */
}
```

Real Arithmetic

Follow the real number arithmetic below

```
int main(){
  double x, double y = 5.0;
  x = y * 2 + 1;
  x = (y * 2) + 1;
  x = y * (2 - 1);
  x = x * x + y - 1;
  x = y / 2;
  x = y % 2; // !!!
}
```

Numeric conversions

- C will automatically convert between `int` and `double`
- Context matters a lot though: all integers means integer division (no fractions)
- Problem 2 of HW 1 deals with this
- Example code: `w01-2-code/number_conversions.c`

More on Variables Types

Tons of variable types in C: Wikipedia

- `int`, `double`, `char` are relevant for this class
  - *Repetition is important in education*
  - How much memory does each one take?
- Other types vary these sizes (`long`, `float`, `short`, etc.)
- Actually a `bool` with `true`/`false` (C99, do `#include <stdbool.h>`)
- `size_t` memory consumption (more later)

Common C operators

Will cover each of these as we progress

Arithmetic + - * / %

Comparison = > < < >= !=

Boolean && ||

- Next week with Conditionals

Memory & and *

Bit Ops ^ | &

Compound += -= *= /= . . .

Conditional ? :

Input/Output

Beginning C

- Terminal `printf` and `scanf`
- Text Files (later) `fprintf` and `fscanf` with `fopen` and `fclose`
- Maybe Binary I/O with `fwrite` and `fread`

printf

Simple String messages

```
printf("Hello world\n");
printf("Line 1\nLine 2\nLine 3\n");
```

Formatted Output

Substitute variable values into format string at certain locations

%d	integer	%lf	double
%c	character	%s	string

```
printf("An integer %d\n",123);
printf("A real %f\n",    0.456);
printf("A string %s\n",  "sweet");
// Multiple outputs in single statement
printf("An integer %d   A real %lf  A string %s  \n",
                  123,        0.456,       "sweet");
```

Formatting Output

`%lf` is a *format specifier*

- What to print (`double` in this case)
- How to print it (default in this case)

Many options available to alter appearance of numbers. An important one: number of digits beyond decimal

`%.8lf`	8 digits
`%.6lf`	6 digits (default)
`%.3lf`	3 digits
`%.1lf`	1 digits
`%.0lf`	0 digits

```
double pi = 3.141592654;
printf("%lf\n%.8lf\n%.6lf\n%.3lf\n%.1lf\n%.0lf\n",
         pi,    pi,    pi,    pi,    pi,    pi);
```

See `printfing.c`

scanf

- For input, especially from terminal
- Format string specifies kind of input

```
/* Demonstrate some scanf functions, relevant for HW1 */
#include <stdio.h>
int main(){
  printf("Input an integer and a real\n");
  int myint;
  scanf("%d", &myint);          /* & ??? */

  double mydoub;
  scanf("%lf", &mydoub);        /* %lf ??? */

  printf("i: %d   d: %lf\n", myint, mydoub);

  printf("Again!\n");
  scanf("%d %lf", &myint, &mydoub);
  printf("i: %d   d: %lf\n", myint, mydoub);
}
```

Multiple Inputs w/ scanf

scanf is also variadic

```
int main(){
  int i,j,k;
  double x,y;
  printf("Give me an int: ");
  scanf("%d",&i);
  printf("Give me 2 ints, 2 doubles: ");
  scanf("%d %d %lf %lf", &j,&k,&x,&y);
}
```

Doubles in I/O

WARNING: abou 20% of you will use

```
double x;
scanf("%f",&x);
```

and wonder WTF is wrong. You will eventually change it to

```
double x;
scanf("%lf",&x);
```

find your program now works fantastically and want to strangle the `libc` guys.

For simplicity use `%lf` for both `printf` and `scanf` with doubles

```
double x = 1.5;
printf("%lf\n",x);
printf("Enter x value: ");
scanf("%lf\n",&x);
```

Exercise: Lawn Mower Man

Spec

- Write a program that takes the length and width of a rectangular yard and the length and width of a rectangular house situated in the yard.
- Your program should compute the time requried to cut the grass at the rate of two square feet per second.
- Read the inputs 2 at a time.
- Print the number of seconds with only 1 digit after the decimal point.

Demo

```
lila [w01-2-code]% gcc lawn.c
lila [w01-2-code]% ./a.out
Yard length and width (ft):
120.5 90.1
House length and width (ft):
80 40.2
Time to cut yard (seconds):
3820.5
lila [w01-2-code]% ./a.out
Yard length and width (ft):
310.4 180.3
House length and width (ft):
200.1 400.1
Time to cut yard (seconds):
-12047.4
```

In First Programs Covered...

- ☒ Comments
- ☒ Statements/Expressions
- ☒ Variable Types
- ☒ Assignment
- ☒ Basic Input/Output
- ⊟ Function Declarations (`main`)
- □ Conditionals (if-else)
- □ Iteration (loops)
- □ Aggregate data (arrays, structs, objects, etc)
- ⊟ Library System (`#include <stdio.h>`)

BREAKTIME

Back in 15 minutes

Goals

- More on `#include`
- Meet `math.h`
  - Needed for HW 1, Problem 1
- Brief overviews of other C stuff

Compilation and Preprocessing

gcc performs a bunch of steps

- Parse, syntax check, optimize, generate assembly, assemble, link. . .
- One step is especially tied to C: preprocessing

Preprocessor

- A partner language to C
- Change program text before compilation
- Add files, Substitute text
- Use directives: `#include` and `#define` mostly
- Makes early changes to the program (pre in preprocessor)

Before and After

Before

```
#include <stdio.h>
#define SOME_NUMBER 42
#define SOME_STRING "Good Stuff"
#define SOME_CODE (x = 2*x)

int main(){
  printf("string: %s\n",
 SOME_STRING);
  int x = 1 + SOME_NUMBER;
  SOME_CODE;
  printf("number: %d\n",x);
}
```

After

```
... stuff from stdio.h ...
...
...


int main(){
  printf("string: %s\n",
  "Good Stuff");
  int x = 1 + 42;
  (x = 2*x);
  printf("number: %d\n",x);
}
```

Typical Preprocessor Use

- Constant declaration
  - Convention: `CONSTANT_IN_ALLCAPS`
  - `#define PI 3.14159`
  - `#define KMS_PER_MILE 1.609`
  - Contrast to constant global variables
- Including other files
  - Headers (`xxx.h`)
  - `#include <stdio.h>` - bring in `printf`

Notice: no semicolons for preprocessor statements

Math Library

Need Math Functions/Library for HW 1

- Square root `sqrt`
- Rounding up and down with `ceil` and `floor`

Calling Functions

Usually `x = functioname(arg1, arg2, arg3);`
Compiler checks

- `functioname` is defined somewhere
- Number of args (3 here) matches number expected
- Types of args match expected
- Stores answer in variable `x`

Math Library

Provides math functions like

square root `sqrt(x)`

natural logarithm `log(x)`

trigonometry `cos(x) sin(x)`

exponentiation `exp(x) pow(x,y)`

rounding `round(x) floor(x) ceil(x)`

Full list on Wikipedia

Note on Math calls

Haven't talked about reading function declarations yet.

- `sqrt`, `log`, `ceil`, `floor` all take a single double and return a double

  ```
  double sqrt( double arg );
  double log( double arg );
  double floor( double arg );
  double ceil( double arg );
  ```

- `pow` takes two doubles and returns a double

  ```
  double pow(double base, double exp);
  ```

`math.h` functions

- See online Ref: http://en.cppreference.com/w/c/numeric/math

Include a header

- For standard input/output

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

- For math

```
#include <math.h>
```

- What about other functions
  - String functions?
  - Time functions?
  - Numerical limits?

Using Math Functions

In `mathdemos.c`

```c
/* Demonstrate use of math functions.  */
#include <stdio.h>
#include <math.h>

int main() {
  double x = 12.5;
  double y = 5.8;
  printf("log(x) = %f\n", log(x));
  printf("cos(y) = %f\n", cos(y));
  printf("x^y = pow(x,y) = %f\n", pow(x,y));
  printf("floor(y) = %f\n", floor(y));
  printf("ceil(y) = %f\n", ceil(y));
  return 0;
}
```

Not as simple as that

`math.h` What functions are in math library

- Not the function definitions

`libm.so` The actual binary library

- Could be called something else `libm.so.6` or `libm.a`

Linking: Obtuse Library System

Tell `gcc` to *link* the math library to your program

```
gcc mathdemos.c -lm
```

- `-l` means link something
- `-lm` means link the `libm` library (math)
- `-lstuff` means link the `libstuff` library

Where are these libraries?

All over the place - compiler searches, ask it where

```
/lib/
/usr/lib/
/lib/x86_64-linux-gnu/4.6/
/lib/x86_64-linux-gnu/
/usr/lib/gcc/x86_64-linux-gnu/4.6/
/usr/lib/gcc/x86_64-linux-gnu/4.6/../../../../x86_64-linux-gnu/lib/x86_64-linux-gnu/4.6/
/usr/lib/gcc/x86_64-linux-gnu/4.6/../../../../x86_64-linux-gnu/lib/x86_64-linux-gnu/
/usr/lib/gcc/x86_64-linux-gnu/4.6/../../../../x86_64-linux-gnu/lib/../lib/
/usr/lib/gcc/x86_64-linux-gnu/4.6/../../../x86_64-linux-gnu/4.6/
/usr/lib/gcc/x86_64-linux-gnu/4.6/../../../x86_64-linux-gnu/
/usr/lib/gcc/x86_64-linux-gnu/4.6/../../../../lib/
/lib/../lib/
/usr/lib/x86_64-linux-gnu/4.6/
/usr/lib/x86_64-linux-gnu/
/usr/lib/../lib/
/usr/lib/gcc/x86_64-linux-gnu/4.6/../../../../x86_64-linux-gnu/lib/
/usr/lib/gcc/x86_64-linux-gnu/4.6/../../../
```

Can also tell compiler to look in other spots - later

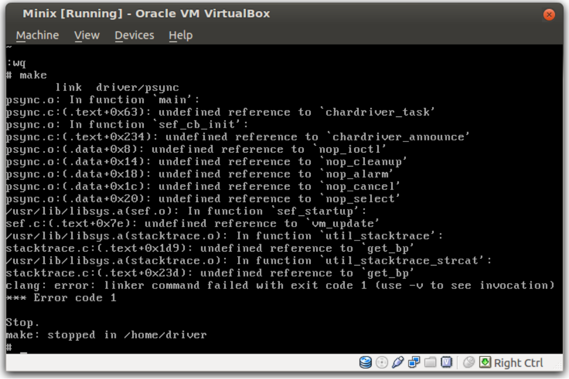
What do libraries look like?

Binary files, usually ELF format. Usefule unix commands are

- `nm`: show *names* in a binary executable (works on cygwin)
- `readelf`: read info about binary exectuabl (linux only)

For our class

Typically won't have to mess around with too many libraries.



```
~
:wq
# make
       link  driver/psync
psync.o: In function `main':
psync.c:(.text+0x63): undefined reference to `chardriver_task'
psync.o: In function `sef_cb_init':
psync.c:(.text+0x234): undefined reference to `chardriver_announce'
psync.o:(.data+0x8): undefined reference to `nop_ioctl'
psync.o:(.data+0x14): undefined reference to `nop_cleanup'
psync.o:(.data+0x18): undefined reference to `nop_alarm'
psync.o:(.data+0x1c): undefined reference to `nop_cancel'
psync.o:(.data+0x20): undefined reference to `nop_select'
/usr/lib/libsys.a(sef.o): In function `sef_startup':
sef.c:(.text+0x7e): undefined reference to `vm_update'
/usr/lib/libsys.a(stacktrace.o): In function `util_stacktrace':
stacktrace.c:(.text+0x1d9): undefined reference to `get_bp'
/usr/lib/libsys.a(stacktrace.o): In function `util_stacktrace_strcat':
stacktrace.c:(.text+0x23d): undefined reference to `get_bp'
clang: error: linker command failed with exit code 1 (use -v to see invocation)
*** Error code 1

Stop.
make: stopped in /home/driver
# _
```

In the real world, compiler problems with libraries will bring you hours of joy.

Simple Practice Task

Compute

$$\frac{x^{1.5} \times \cos(y/2)}{\ln(x) + \log_{10}(y)}$$

- Prompt for inputs
  - x integer input
  - y real input
- Compute above expression
- Print output to 4 digits beyond decimal
- Assume $x, y > 0$

Program in `mathy.c`

Briefly - Functions

Declare a function

```
int add_and_double(int a, int b){
  int c = a + b;
  return 2*c;
}

void print_name(char *name){
  printf("The name is %s\n", name);
}
```

Briefly - Iteration

```
int i = 0;
while( i < 10){
  printf("i is %d\n", i);
  i = i + 1;
}

for(int i = 0; i < 10; i++){
  printf("i is %d\n", i);
}
```

Briefly - Aggregate Data

Homogeneous, Repeated

```
int myints[10];
myints[5] = 100;
myints[0] = 1;
myints[9] = myints[5] + myints[0];
```

Heterogeneous

```
typedef struct {
  double height;
  int age;
  char name[100];
} person_t;
...
person_t chris = {.height=70.5, .age=33, .name="Chris"};
```

Wrap-up

Hot Seats write card-count

- HW 1 due next Tuesday
- For next week - Zyante 3-6
- 2 weeks from today - Exam 1