Program submissions: Up to 2 extra days.
Notes: Turn in notes before Monday of the next week using git. PDF, plaintext, or Word. These notes can be used on tests the following week.
Coding help: Add a comment in your code acknowledging who helped you.
Schedule: Programs are assigned Saturday night. Design documents are due Thursday night (LaTex with diagrams preferred, Word discouraged). Programs are due on Sunday night.
Late Policy: If you turn in programs late, you lose 25% each day afterwards.
Documentation:
DESIGN.pdf

- Describes the design of your program and answers pre-lab questions.
- Describes the algorithms and design decisions you have made.
- Describes the problem you are solving, inputs, and expected outputs.
- Includes pictures, diagrams, and pseudocode.

WRITEUP.pdf

- Contains your analysis of running your program.
- For example, this file would contain the results of your program, like a science experiment.

C ancestry

- Came from language called B
- Created in 1972, 2011 C11 was made, 2018 C18 was made

#include <stdio.h> → angled bracelets mean Standard I/O package
Return 0 = success
Int main(void) = no arguments passed
printf("%3.0f") = print 3 numbers with 0 after decimal point
printf("%6.1f") = print 6 numbers with 1 after decimal point
Scoping

- A scope is defined by curly braces (for loops, while loops)
- Internal variables have higher priority over higher scope variables.

String (*char)

- Should always end with a null byte ('\0')
-

./pig < pig.c - passes pig.c as a text file to ./pig, so functions such as getchar() read text from pig.c.

Compilation process of a C program

- 1. Preprocessor- processes all directives starting with a #, like include or DEFINE
    - Comments are removed
    - Creates an intermediary file, hello.i
- 2. The compiler- Converts hello.i into assembly code
    - Lexical phase- Groups key words, punctuation, variables together, throw away whitespace
    - Syntax analysis phase- Parses things, checks if things are syntactically correct.

        - Creates a tree of the tokens created from the lexical phase. Nodes of this tree might be “if”, “statement”, “>”, while roots might be “return”, “True”, “3”
- 3. The Assembler- Converts assembly language instructions into binary
    - Two pass- maps symbols first, then traverses through assembly to generate binary code.
- 4. The Linker- Links hello.c to other objects and libraries.
- 5. The Loader- Lives in the operating system. Allocates space in memory, resolves symbols between references, fix all dependent locations to point to allocated memory.
- 6. The Memory
    - Bottom layer- The text- your code
    - 4th layer- initialized data
    - 3rd layer- uninitialized data
    - 2nd layer- heap- grows upwards
    - 1st layer- stack- grows downwards

Compiler vs Interpreter

- Compiler translates a programming language, goes through a sequence of translations, then outputs an executable.
    - Translates entire program at once.
- Interpreter - Directly executes code without needed to compile.
    - Translates program one line at a time.
    - Hundreds of times slower.

GCC vs cc vs clang

- GCC- gnu C compiler, default on linux
- CC- Unix/Linux environment variable that points to default compiler
- Clang- Default on Mac

Makefile

- Automates building executables from source code, so you don’t have to compile every time.
- Makefile contains rules on how the Make should run.