ECE 471 – Embedded Systems
Lecture 7

Vince Weaver

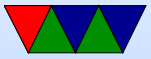
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

15 September 2021

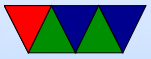
Announcements

- HW#2 due Friday



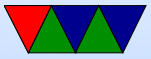
C Review

In past years sometimes the reason a HW assignment didn't work was due to using C poorly rather than misunderstandings of the desired algorithm.



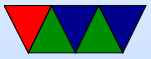
Loops in C

- `for(i=0;i<10;i++) {}`
- `while(i<10) { i++; }`
- `do { i++; } while(i<10);`
  Always runs at least once



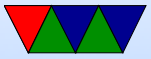
`printf()` in C

- printf()
- Lots of options, see man page
- How print an integer? `printf("%d",i);`
- Character? String? floating point?
  `printf("%c %s %f %x",c,s,f,x);`
- More advanced formatting stuff
  `printf("%0.3f",f);`
- Escape characters like percent, newlines and quotes
  `printf("\t \n \" \%");`



Common C Pitfalls – Memory

- Can dynamically allocate memory with `malloc()` and `calloc()`
- Should check returned value against NULL. What happens if you de-reference a NULL pointer?
- Need to `free()` memory at end or you can leak memory
- Note at program exit the operating system will close files/free memory
- Out of bounds memory access and double-frees can be problem. **Valgrind** utility can help debug these errors.

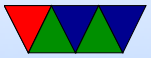


Common C Pitfalls – Braces

- Missing braces

```
if (a==0)
        b=2;

if (a==0)
        b=2;
        c=3;
```

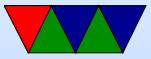


Common C Pitfalls – equality check

- = vs ==

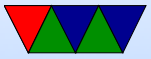
```
if (a=0) do_something_important();
```

- Never ignore warnings from the compiler!
- Some people will use `if (0=a)` to force an error



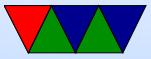
Coding Style

- How should you format your code?
- Does C have rules? Not really.
- International Obfuscated C Code Competition (IOCCC)
- Your company or open-source project might have strict rules
- Things like how tabs vs spaces, how wide are tabs, if curly brace goes after function declaration or line down
- Also rules about commenting style



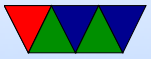
Debugging – when things go wrong

- Use a debugger like gdb
  - Compile your code with `-g` for debug symbols
  - Run `gdb ./hello`
  - `bt` backtrace, `info regis` gives register, `disassem` disassembles, etc.
- Sprinkle printf calls



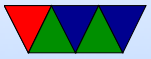
How Executables are Made

- Compiler generates ASM (Cross-compiler)
- Assembler generates machine language objects
- Linker creates Executable (out of objects)



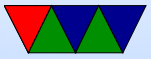
Tools – Compiler

- takes code, usually (but not always) generates assembly
- Compiler can have front-end which generates intermediate language, which is then optimized, and back-end generates assembly
- Can be quite complex
- Examples: gcc, clang
- What language is a compiler written in? Who wrote the first one?



Tools – Assembler

- Takes assembly language and generates machine language
- creates object files
- Relatively easy to write
- Examples: GNU Assembler (gas), tasm, nasm, masm, etc.



Tools – Linker

- Creates executable files from object files
- resolves addresses of symbols.
- Links to symbols in libraries.
- Examples: ld, gold