CSIS10C C Introduction

Fundamentals of
Modern Software Development

Agenda

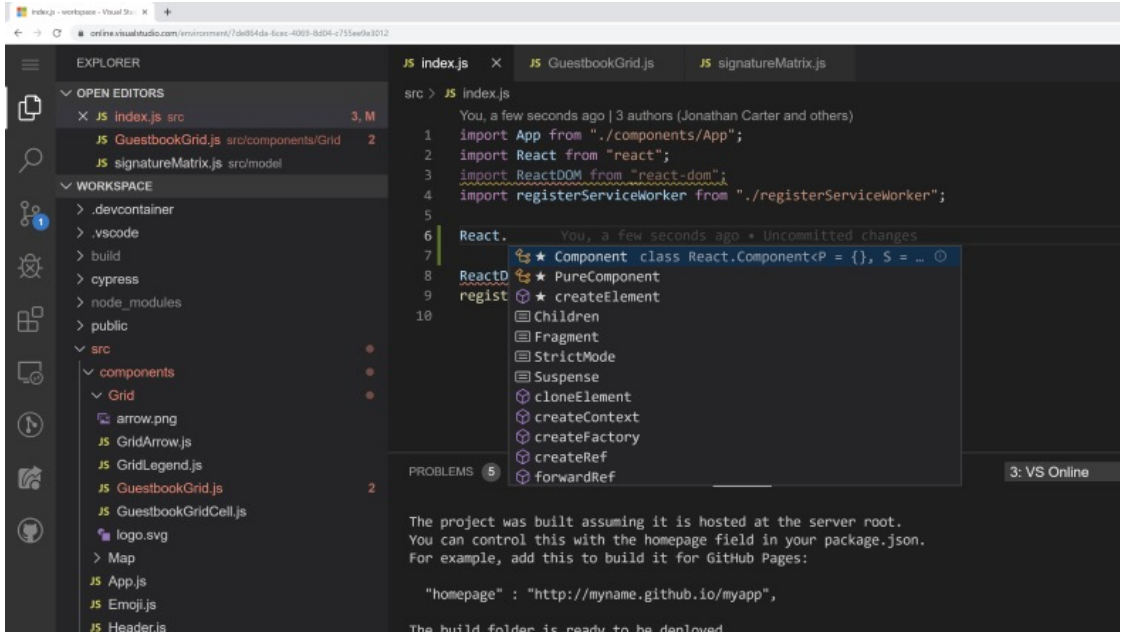
1. The Arduino
2. Software Development
3. Tool Chains
4. Command Line
5. Tools
   - git
   - gcc
   - make
   - avrdude
   - environments

Three Components of the Arduino

1. **Hardware** – Arduino Uno
2. **Application** – Arduino IDE
3. **Language/Library** – C++/ Arduino Libraries

Software Development

- Modern software development => Integrated Development Environment (IDE)
- Values of an IDE:
  - Automation
  - Abstraction
  - All-in-One
- Issues of an IDE
  - Single point of failure
  - Hides critical details
  - Complex tool



- Arduino is a good IDE for learning C++ and embedded microcontrollers…same issues of an IDE
- We'll use same tools, however…via the command line to better understand the process

Software Development

- Embedded microcontroller software development tasks:
  1. *Editing*
  2. *Compiling*
  3. *Linking*
  4. *Create an executable*
  5. *Upload the executable*

- The Arduino IDE provides all these tasks
- We'll break down the tasks and execute them separately to gain a better understanding of what is required

Tool Chains

- What do we call the set of individual tools?
- A tool chain
- A tool chain is a set of tools, usually executed on the command line in order to create an application
- Tool chains are based on a specific language, operating environment and application
- Tools in chain…compilers, linkers, debuggers, programmers

Using the C Tool Chain

```
lkoepsel@M1Mini examples/blink (main) »
make flash
avr-gcc -Og -ggdb -std=gnu99 -Wall -Wundef -Werror -funsigned-char -funsigned-bitfields -fpack-struc
t -fshort-enums  -ffunction-sections -fdata-sections  -DF_CPU=16000000UL   -DBAUD=9600UL -DSOFT_RESE
T=0 -I.  -I../../Library -mmcu=atmega328p -c -o main.o main.c
avr-gcc -Wl,-Map,main.map  -Wl,--gc-sections  -mmcu=atmega328p main.o ../../Library/analogRead.o ../
../Library/analogWrite.o ../../Library/button.o ../../Library/delay.o ../../Library/digitalRead.o ..
/../Library/digitalWrite.o ../../Library/pinMode.o ../../Library/sysclock.o ../../Library/tinymt32.o
 ../../Library/tone.o ../../Library/uart.o ../../Library/unolib.o  -o main.elf
avr-objcopy -j .text -j .data -O ihex main.elf main.hex
avrdude -c Arduino -p atmega328p -F -V -P /dev/cu.usbmodem3101 -b 115200 -U flash:w:main.hex

avrdude: AVR device initialized and ready to accept instructions

Reading | ################################################## | 100% 0.00s

avrdude: Device signature = 0x1e950f (probably m328p)
avrdude: NOTE: "flash" memory has been specified, an erase cycle will be performed
         To disable this feature, specify the -D option.
avrdude: erasing chip
avrdude: reading input file "main.hex"
avrdude: input file main.hex auto detected as Intel Hex
avrdude: writing flash (1092 bytes):

Writing | ################################################## | 100% 0.22s

avrdude: 1092 bytes of flash written

avrdude done.  Thank you.
```

Command Line

- Command line is the simplest interface and the one closest to the fundamentals of the operating system
- As such, the most common method for implementing a tool chain and it is easy to automate
- Command line applications:
  - Windows – cmd.exe, Windows Terminal,
  - Linux – terminator, konsole, tilda, many programs
  - macOS – Terminal, iTerm, warp
- Common commands
  - pwd - present working directory
  - ls – list files in directory
  - cd – change directory
  - rm – rm file or directory

Tools

- git – software version control
- gcc – compiler, linker and loader
- make – automation tool
- avrdude – AVR upload tool (programmer)
- environments – a term for your specific operating system, hardware and tool set

git – software version control

- git
  - Provides the ability to ”roll-back” changes if a problem is found
  - Allows many people to work on the same code then merge each other’s work
  - An industry standard for software distribution
- Installation
  - sudo apt install git (Linux, WSL)
  - brew install git (macOS)
  - git-version-xx.exe (Windows) https://gitforwindows.org
- Primary commands
  - git init – initialize a directory to be tracked by git
  - git status – show changed and uncommitted files
  - git add –A – stage all changed and uncommitted files
  - git commit –m “text” – commit all files and update the tracking number

gcc – compiler, linker and loader

- gcc
  - The gnu compiler, works on many languages, C, C++… https://gcc.gnu.org
  - Well known and well maintained free set of compiler tools
  - The version specific to the Uno's ATmega328P chip is called **avr-gcc**
- Installation
  - sudo apt install avr-gcc (Linux, WSL)
  - brew install avr-gcc (macOS)
  - Download latest https://blog.zakkemble.net/avr-gcc-builds/
- Primary commands (compile, link and load)
  - avr-gcc -Os -DF_CPU=16000000UL -mmcu=atmega328p -c -o main.o main.c
  - avr-gcc -mmcu=atmega328p main.o -o main
  - avr-objcopy -O ihex -R .eeprom main main.hex

make – automation tool

- make
  - The compile/link/load commands can be complex
  - make allows the commands to be in a file and executed based on parameters
  - The first step in automating the tool chain
- Installation
  - sudo apt install make (Linux, WSL)
  - brew install make (macOS)
  - Download latest https://blog.zakkemble.net/avr-gcc-builds/
- Primary commands
  - make – compile, link and load (CLL) the file into a hex file for upload to Uno
  - make flash – upload hex file to Uno, recompile/link/load if files have changed
  - make clean – delete all non-source files for a complete new CLL

avrdude – AVR upload tool

- avrdude
  - Standard method of uploading files to AVR microcontrollers
  - Used by Arduino
  - Seeing a resurgence in development
- Installation
  - sudo apt install avrdude (Linux, WSL)
  - brew install avrdude (macOS)
  - Download latest https://blog.zakkemble.net/avr-gcc-builds/
- Primary command
  - avrdude -F -V -c arduino -p ATMEGA328P -P /dev/ttyACM0 -b **115200** -U flash:w:main.hex
  - Requires specific environment to be setup based on users operating system, hardware and microcontroller board (env.make)

environments – your specific OS, hardware and tool set

- Similar function to Arduino -> Tools -> Port and -> Board
- env.make:

```
# Arduino UNO and compatible boards
MCU = atmega328p
SERIAL = /dev/cu.usbserial-01D5BFFC
F_CPU = 16000000UL
BAUD  = 9600UL
SOFT_RESET = 0
LIBDIR = ../../Library
PROGRAMMER_TYPE = Arduino
PROGRAMMER_ARGS = -F -V -P $(SERIAL) -b 115200
```

- Typical SERIAL definitions:
  - Linux: /dev/ttyACM0, /dev/ttyUSB
  - macOS: /dev/usbserial-01D5BFFC
  - Windows: COM3, COM4

C Reference Material

- Book: *"The C Programming Language"* Kernighan and Ritchie

Second Edition

  - The original C programming book
  - Still very, very good for learning C
- AVR-libc contains the C Library Reference material
  - https://avr-libc.nongnu.org
  - https://avr-libc.nongnu.org/user-manual/index.html
- C Tutorial Websites
  - https://www.tutorialspoint.com/cprogramming/index.htm
  - https://www.guru99.com/c-programming-tutorial.html

Next Steps

1. Install Arduino IDE and test your system with your Uno (and get your serial port number!)
2. Install git, make, and the AVR tool chain per:

   https://www.wellys.com/posts/avr_c_gettingstarted/
3. For your OS, work the exercises of:
   1. Using gcc and avrdude to upload blink to Uno
   2. Using make to automate the uploading
4. Use git to clone the Lab repository on your system

   https://github.com/lkoepsel/Labs_10C_Class