ECE 471 – Embedded Systems
Lecture 5

Vince Weaver

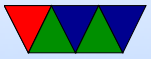
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

13 September 2024

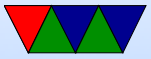
Announcements

- HW#2 will be posted
- Note you won't need to bring your Pi to class



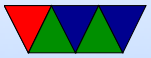
Why C?

- System Programming Language (OS/embedded)
- Portability (sort of) (i.e. how big is an `int`)
- Higher than assembly (barely)
  Pearce: "all the power of assembly with all the ease-of-use of assembly"



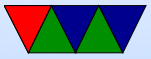
Why C over Java or C++?

- They can hide what is actually going on.
- C statements map more or less directly to assembly.
- With things like operator overload, templates, exceptions, garbage collection, extra layers are added.
- This can matter for both size, speed, determinism, and real time.
- On embedded bight be restricted to a C++ subset



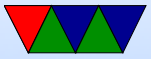
Why C over Python?

- Python is interpreted
- Mostly speed. (although you can JIT)
- Also if accessing low level hardware, in general you are calling libraries from python that are written in C anyway.



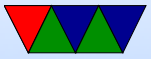
Why C over Rust? (or go or Zig)

- Don't overlook momentum of an old platform, sample code, libraries, etc.
- Rust still a moving target, needs to settle down a bit
- Maybe in the future



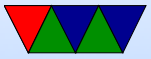
Downsides of C? – Undefined Behavior

- Compiler is allowed to do anything it wants (including dropping code) if it encounters something undefined by the standard.
- This can be something as simple as just overflowing a signed integer or shifting by more than 32.
- People joke of "nasal demons" i.e. standard says anything can happen here, even demons flying out of your nose



Downsides of C? – Too Much Trust

- "Enough rope to shoot yourself in the foot".
- C gives a lot of power, especially with pointers.
- It assumes you know what you are doing though.
- With great power comes great responsibility.

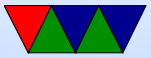


Downsides of C – Security

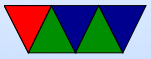
- Biggest issue is memory handling (lack thereof)
- Buffer overflows

```
int a[5];
a[0]=1;                 // fine
a[10000000]=1;          // obviously bad
a[5]=1;                 // subtly bad
```

- How can that go wrong? Crash? Corrupt Memory? Wrong results? Total system compromise?
- Overwriting stack can be bad, as return address there



- Especially if user input going into the variable

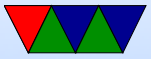


Downsides of C – Security

- Why can't the compiler stop you? Maybe it can in above example.
- What if the offset read from user input instead?

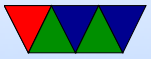
```
scanf("%d",&i);
a[i]=1;
```

- Could still maybe detect this, but would need to add extra code which might be slow?
- Problem is arrays and pointers are same in C



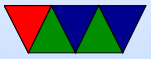
Downsides of C – Pointers

- Wouldn't we be better off w/o pointers?
- Philosophy of safer languages (can even be faster! Can avoid aliasing pessimizations!)
- The actual processor does pointers a lot though
- We are close to hardware
- In embedded we might need to poke to exact memory accesses (MMIO and such)



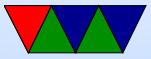
Program in C (song/video)

- https://www.youtube.com/watch?v=tas00586t80



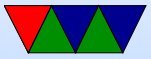
Homework #2 – Background

- It's mostly about getting everyone up to speed on the Pi as not everyone has used one before
  - Many ways to set up your Pi for use, everyone has a different preference
  - Be sure to change your password from default
- Also a small C coding assignment, and some short-answer questions.



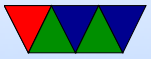
Using the Pi for this Class – Two Challenges

- Getting Pi to the point you can log in
- Getting files onto and off of the board. (Definitely needed for homework)

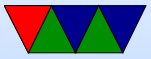


Installing Linux

- Any Linux fine
  I typically use Raspberry Pi OS (Raspbian)
  If you use same as me I can more easily help
- Easiest way is to buy SD card with image pre-installed
- If starting with a blank SD card:
  - `https://www.raspberrypi.com/software/`
    has images and even a tool you can use that will help you install things.
  - If Pi3 or later you can install the 64-bit version. Either

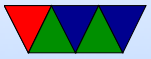


32-bit or 64-bit both fine for this class.
  - This includes the ability to pre-configure things like password, ssh, keyboard, locale, wifi
  - Warning: it's a large download (900MB?) and takes a while to write to SD (which is slow)
- If you end up instead manually writing an image to SD using command-line linux (the `dd` tool or similar) be sure **to get right partition** as the destination. It's easy to accidentally overwrite your laptop/desktop's hard drive



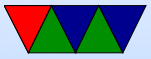
Booting Linux on Pi

- Why called booting?
- Bootstrapping?
- Pull oneself up by own bootstraps? Meaning to get something going starting with nothing



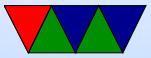
Getting Linux going

- Put SD card in
- Hook up input/output (see later)
- Plug in the USB power adapter; *NOTE* can also draw power over serial or usb or sometimes HDMI
- Lights should come on and blink and should boot
- If you have a display hooked up you will get a rainbow on screen which is GPU starting up.
  If installed GUI edition it will have a splash screen, if barebones text then instead a number of raspberries



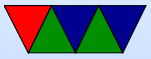
should appear and some Linux boot messages

- Things can also go wrong in ways hard to troubleshoot



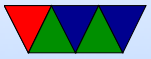
First Boot

- First boot a menu comes up. This varies depending on what OS you are on, I'm assuming Raspbian v11 (Bullseye) here
  - Will prompt for keyboard type. You want standard US English keyboard, but by default it will give you UK (Pis are from England). It can be a bit hard to navigate to what you want
  - It will ask for username password
    In the old days it would just default to pi/raspberry.



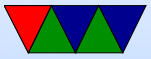
Why was that bad security-wise?

- At this point it might continue until you get to a login prompt

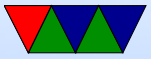


More Configuration

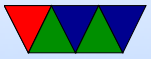
- The system will prompt you to configure things. If for some reason it doesn't you can always run "raspi-config" to configure more
- System Settings
  - Enable Wifi
  - Pick a Hostname
  - various other things
- Display Options
  - Only matter if you are using a TV as a display



- Interfaces
  - Can enable ssh for remote network logins
  - Can also enable SPI, i2c, and 1-wire that we'll use in class
- Advanced
  - You might be able to expand the disk image to fill the whole sd-card, not sure if that's automatic
  - Performance: can overclock, select how much RAM used by GPU
- Localization
  - Probably already picked keyboard

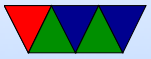


- Can pick language. Probably your best bet is `en_US.UTF-8`
- Pick timezone: Americas/NY
- Can configure Wifi more (what frequencies it can use depend on what country you are in)



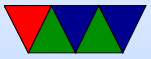
Other Optional things you can do

- If on network, can install updates
  sudo apt-get update
  sudo apt-get upgrade



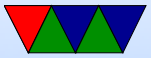
Connecting to the Pi

- Monitor/Keyboard (Easiest)
- Network Connection
- Serial Connection



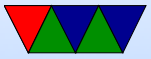
Monitor and Keyboard

- HDMI monitor, USB keyboard, USB mouse (optional unless using gui)
- Need HDMI cable (micro-HDMI on pi4/pi5)
- Used to be a nice setup in the Electronics Lab but I don't think that exists anymore unfortunately.

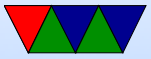


Wired Ethernet Connection

- Ethernet cable
- Either an Ethernet port, or connect direct to PC
- If something goes wrong on boot hard to fix
- Can also try this with a wireless connector
- Can hook it onto dorm network, but complex (and maybe need to request a static IP)

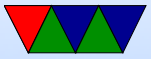


- Can also direct connect between PC (configure pi with a local address like 192.168.1.2 and set your wired Ethernet on PC side to something like 192.168.1.1 and then use ssh to connect)



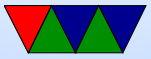
Wifi

- Recent Pis have built-in wifi
- I haven't done this much, but you can find directions online
- Setting up for edu-roam might be tricky as there are extra steps

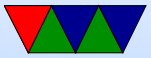


Serial Connection

- Just mentioning this as it's possible, but it's unlikely you want to do this
- Need USB/serial adapter
- Need another machine to hook to, with a comm program minicom, putty
- Thankfully unlike old days don't need specific NULL modem cable. Still might need to set some obscure



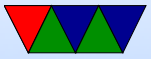
COM port settings (BAUD, stop bits, parity) and console TERM settings (ANSI, VT102).



Other/obsolete

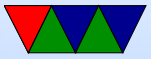
- In way past times people used “netatalk” on MacOS
  I think this doesn’t work anymore

- There are rumors about being able to login using USB-C on Pi4 but haven’t seen it done

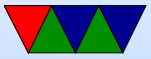


Transferring Files

- Easiest: if set up like a desktop just download with browser
- USB-KEY: transfer data using a regular USB-key In theory the Pi should auto-mount the drive for you
  May need to mount / umount by hand or be root
- Network: just use ssh/scp
- Serial: sz/rz ZMODEM
- Putting sd-card (after unpowering!) in another machine. Challenge: Filesystem is in Linux format (ext4) so

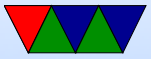


Windows and Macs can't read it by default.



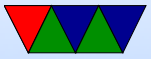
Homework #2 – Unpacking Assignment

- It's a .tar.gz file. What is that?
- Sort of the Linux equivalent of a zip file
- tar = tape archive (ancient history) that runs lots of files together
  gz = gzip, which compresses it (makes it smaller)
- you may see other (Z, bz2, xz). What are the differences?
  Mostly in compressed size vs compress/uncompress resources
  gzip good enough for what we are doing.



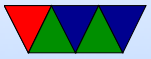
Homework #2 – Editing C on Pi

- Take some existing C code and modify it.
- Can use the editor of your choice. Many on Linux.
  - "nano" is easy
  - "vim" if you are serious about Linux.
  - "emacs" – I've known some emacs wizards
  - Also various graphical ones
  - Modern (MS VS Studio? Eclipse? Atom?) but can take more RAM than the Pi has



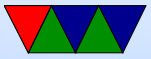
What you will do before starting HW2

- Get Linux installed
- Login with the default user/password (on Raspbian it is pi / raspberry)
  You can use `adduser` to add a new user and/or `passwd` to change a password.
- Learning a little bit of Linux. Most importantly compiling C/asm programs and transferring HW assignments in and out



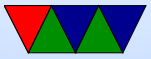
SD Card Digression

- Why are they so slow?
- BACK UP YOUR WORK. ALL THE TIME. SD cards corrupt easily. Why?
- SHUTDOWN CLEANLY
  menu or `shutdown -h now`
- Try to get things done a little before the deadlines, that way you have some time to recover if a hardware failure does happen.



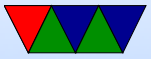
Using the Pi

- I usually assume you'll be doing things at the command line, either at the text console or by starting a terminal emulator (like `lxterm`) in the GUI interface



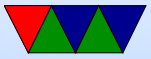
Homework #2 – Editing C on your own computer?

- if you want you can even code it up on your desktop/laptop, but you probably want to copy it over to test before submitting.
- Be careful not to introduce errors if cutting and pasting
- Be sure to test before submitting! If it won't compile I'll take points off



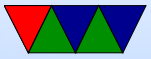
C compiler

- C compiling on Linux
  We will use gcc (what others exist. clang?)
  Typical command line is something like:
  `gcc -O2 -Wall -o hello_world hello_world.c`
  `-O2` is optimization, `-Wall` is show all warnings
  A lot more options, see man page



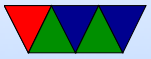
Makefiles

- We use a Makefile to automate the process.
- What is make?
- You give it a list of dependencies, then it automatically sees what files have changed and then runs commands to build things
- Feel free to play with it, but a warning, tabs are significant so weird errors if you use spaces instead.
- If you for some reason add things with Makefiles, be sure you update submit to include the extras



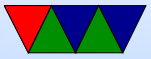
Cross compiling

- Can compile for a different architecture, for example x86 to ARM
- Why do it? Faster. Target doesn't have enough resources. Want to target multiple devices.
- To test would need an emulator (like qemu)



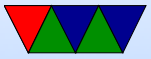
Comment your Code!

- Comment your code!!!!!
- Why?
- I will take points off it you don't.
- Also helps other people looking at your code figure out what's going on.
- Especially the grader
- But also, you looking at your own code at a later date



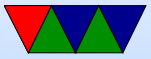
Comment your Code!

- Mostly comment non-obvious stuff.
- for(i=0;i<10;i++) // loop 1 to 10
- for(i=0;i<10;i++) // loop through all i2c LEDs
- Things like \tt i=4.3+10*j; definitely need to
- You can't really over-comment (well you can, but it's harder to over-comment than under-comment)



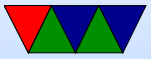
Code Style

- Some classes/projects will have enforced style guide to follow
- For this class not picky
- Having your name and a description of what the overall file and each function is fine
- Even fancier commenting conventions companies will have for automated tools.



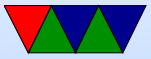
Using git

- Not using gitlab like ECE271, was huge hassle
- Still idea to use some sort of source control management (SCM)
- There are actually worse than git out there
- Who wrote git? Linus Torvalds.



Documentation on Linux commands

- Use `man command` where command is what you are interested in
- Use `man ls` to see how to use ls
- Also useful for functions `man -a printf` or random stuff `man ascii`



HW#2 Something Cool

- ANSI escape color/art – Demoscene / BBS!
- In old days: how could you do colors on screen?
- Over serial port, so couldn't directly write to video card
- Escape sequences: A pattern not normally typed by accident
- ESCAPE (ASCII 27) followed by [then some pattern of characters.
- Can move cursor, change colors, etc.
- Back in the day I used to make a lot of ANSI/ASCII art.