ECE 471 – Embedded Systems
Lecture 5

Vince Weaver

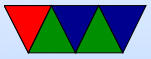
http://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

11 September 2020

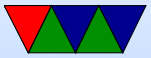
Announcements

- HW#2 was posted
- rasp-pi config? Might default thinking you are in England



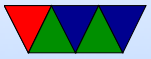
Homework #1 Review

- Be sure to put your name in the assignment.
- Characteristics of embedded system
  - embedded inside – sometimes hard to know. Is a raw pi one? Pi used as desktop? Pi used as retro-pi? Pi controlling a 3D printer?
    Lack of being able to update not necessarily the same
  - resource constrained
  - dedicated purpose
  - real-time



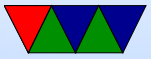
- Embedded System Question
  - Toothbrush is actual specs I came across
  - Real-Time Confusion: we will discuss this more in future.

  - Toothbrush: Just turning off the motor, and it takes an extra 1/2s is not really considered a real time thing. No one dies, no hardware destroyed, just mild annoyance if noticed at all. Now if somehow it had to keep the waveform to H-bridge exact within 1ms or the motor would overheat and catch on fire, that could be



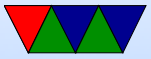
a real-time issue.

- Microwave: having a clock doesn't make it real time. Hopefully the door control has a physical interlock, but you never know. Usually when cooking food second granularity and some jitter not matter much.
- Limited Hardware
  bitness of processor: while 8 or 16 bit probably embedded these days, 32 vs 64 bit not necessarily a sure sign.
- Low-cost is complicated. Something like a desktop

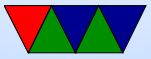


might be optimized for cost extremely, while a one-off embedded system might not, and in fact might be over-engineered (like a space probe) because has to operate in tough conditions.

  - Operating system?
    Can have an OS and still be considered embedded.
  - Be strong in your convictions!
- bits
  - ARM1176 is generally considered 32-bits
  - 6502 generally considered 8 bits



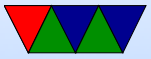
- don't add up the bits
- ASIC
  - cost/power. Depends a lot on numbers made, process, and how well designed it is.
  - Extra hardware overhead? ASIC mostly just flip flops and gates. SoC internally a lot more, but these days not much else is needed.
  - More secure? Can you reverse engineer an ASIC?



C compiling, Makefiles

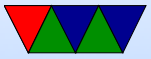
- C compiling on Linux
  We will use gcc (what others exist. clang?)
  Typical command line is something like:
  `gcc -O2 -Wall -o hello_world hello_world.c`
  `-O2` is optimization, `-Wall` is show all warnings
  A lot more options, see man page

- We use a Makefile to automate the process. What is make?
  You give it a list of dependencies, then it automatically



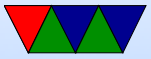
sees what files have changed and then runs commands to build things

Feel free to play with it, but a warning, tabs are significant so weird errors if you use spaces instead.



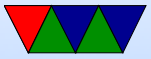
Cross compiling

- 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. Including me the graded. Including you trying to re-use some code a year from now.
  Having your name and a description of what the overall file and each function does doesn't hurt.
  Even fancier commenting conventions companies will



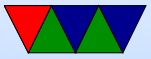
have for automated tools.

Mostly comment non-obvious stuff.

So `for(i=0;i<10;i++)` not so much.

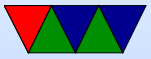
But something like `i=4.3+10*j;` yes.

You can't really over-comment (well you can, but it's harder to over-comment than under-comment)



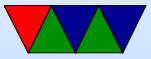
Using git

- 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



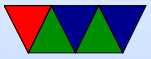
Documentation on Linux commands

- 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`



Using the Pi for this Class – Two Challenges

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

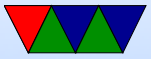


Installing Linux

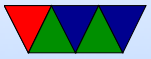
- Any Linux fine, I typically use Raspbian
  Using the same that I do is easiest and I can more easily help

- Easiest way is to buy SD card with image pre-installed
  Also can get NOOBS which will give you the option to select from a variety of images via menu (allowing to install Raspbian)

- If starting with a blank SD card,

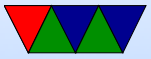


`https://www.raspberrypi.org/downloads/`
has good step by step instructions for getting an image and putting it on a card for a variety of operating systems.
Warning: it's a large download (900MB?) and takes a while to write to SD (which is slow)
dd on Linux, be sure **to get right partition**

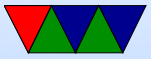


Booting Linux

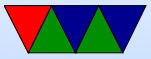
- Why called booting? Bootstrapping? Pull oneself up by own bootstraps? Meant to do something impossible
- Easiest if you have a USB keyboard and HDMI display connected.
  - Put SD card in
  - Hook up input/output (see later)
  - Plug in the USB power adapter; *NOTE* can also draw power over serial/usb and HDMI
  - Lights should come on and blink and should boot



A number of raspberries should appear and some Linux boot messages
  - Things can also go wrong in ways hard to troubleshoot
- First boot a menu comes up. You probably want to do a number of things:
  - Expand to fill disk.
  - Change password if you want
    pi/raspberry is default
  - Change locale— probably defaults to England giving pound char for #. en_US.UTF8, not GB
  - change hostname?



  - for this class, advanced options, enable i2c and spi
- You can get back to the original menu with `sudo raspi-config`
- Don't make fun of the text interface, once upon a time it's all we had.

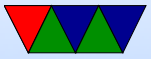


Other Optional things you can do

- Install updates
  sudo apt-get update
  sudo apt-get upgrade

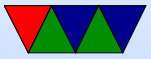
- Add a user account
  adduser vince

- Give new user sudo access
  involves text editing /etc/sudoers



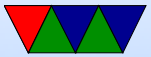
Connecting to the Pi

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



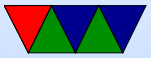
Monitor and Keyboard

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

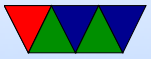


Network/Ethernet Connection

- 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 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

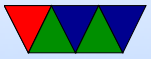


wired Ethernet on PC side to something like 192.168.1.1 and then use ssh to connect)



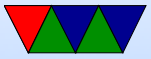
Network/netatalk

- Only works with MacOS (?)
- Some students in the past have used netatalk to connect to their Pi and copy files
- Look for info on Raspberry Pi and "netatalk"



Serial Connection

- Old fashioned, but very good skill to have.
- 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).



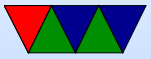
Transferring Files

- Easiest: Putting USB key in rasp-pi
  Easier on B+ (4 USB ports)
  In theory the Pi should auto-mount the drive for you
  May need to mount / umount by hand or be root

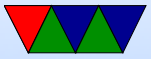
- Network: just use ssh/scp

- Serial: sz/rz ZMODEM

- Putting sd-card (after unpowering!) in another machine.

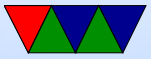


Challenge: Filesystem is in Linux format (ext4) so Windows and Macs can't read it by default.



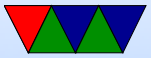
What you will do before starting HW2

- 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



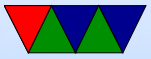
SD Card Digression

- Why are they so slow?
- BACK UP YOUR WORK. ALL THE TIME. SD cards corrupt easily. Why?
- SHUTDOWN CLEANLY
- 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

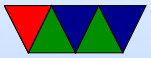
- If using monitor/keyboard you can type `startx` after logging in and getting a nice GUI interface.
- You can do many things through that, but in this class we will use the command line for many things.
- You can select `lxterm` to get a terminal.
- Also if you log in over ssh or connect via serial port all you will get is the command line.



Command-Line Linux

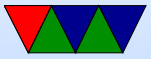
The way we did things in the old days.
Some of us still prefer the command line.

You come up in the "shell". Default is bash, the "Bourne Again Shell" (more computer person humor). There are various shells available (bash, sh, zsh, csh, tcsh, ksh) and you can select via `chfn`.

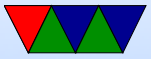


Things for Homeworks from command line

- Editing files: nano, vim, emacs, gui based, just copy over
- Listing files (ls)
- Creating/Changing directories (mkdir, cd)
- Tab Completion
- Suspending jobs

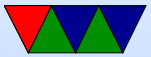


- Running jobs `./hw1`
- Compiling/Make
- Debugging. printf/gdb
- sudo, shutdown
- man to get manpage

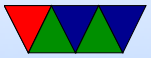


Root Filesystem Layout

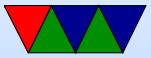
- Executables in `/bin`, `/usr/bin`
- System executables under `/sbin`, `/usr/sbin`
- Device nodes under `/dev`
- Config files under `/etc`
- Home directories under `/home`, also `/root`
- Temp Files under `/tmp`. Often wiped at reboot.



- Magic dirs under `/proc`, `/sys`
- Libraries under `/lib`, `/usr/lib`, sometimes `lib64` too
- Boot files under `/boot`
- `/usr` historically only files needed for boot in /, stuff that can be shared over network (or stored on a second drive if your first drive was too small) would be under `/usr`
- `/opt` often commercial software installed there

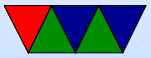


- `/srv`, `/run`, `/var` these are where server programs store data
- `/media`, `/mnt` places to mount external disks like memory keys and CD roms
- `/lost+found` where the disk checker may store lost files it finds when fixing a disk after unclean shutdown

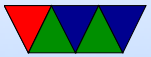


Interesting Config Files

- /etc/fstab – the filesystems to mount at boot time
- /etc/passwd – list of all users, world readable
- /etc/shadow – passwords stored here for security reasons
- /etc/hostname – name of the machine
- /etc/hosts – list of local machines, usually searched before resorting to DNS lookup over network



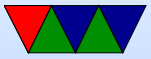
- /etc/resolv.conf – where your nameserver address is put
- /etc/sudoers – list of users allowed to use “sudo”
- /etc/network/interfaces – on Debian the network settings are stored here
- /etc/rc* – what gets run at boot



Devices

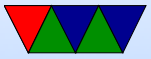
Block vs Char devices

- /dev/sd* – SCSI (hard disks)
- /dev/tty* – tty (teletype, logins, serial ports)
- /dev/zero
- /dev/full
- /dev/random , /dev/urandom



- /dev/loop

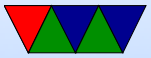
Network devices are an exception.



Interesting /proc Files

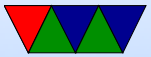
These files are not on disk, but "virtual" and created on-the-fly by the operating system when you request them.

- /proc/cpuinfo – info on cpu
- /proc/meminfo – memory info
- Each process (running program) has its own directory that has info about it



Processes

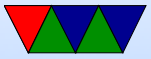
- Each program assigned its own number, a process id, often called a "pid"
- Can list processes with `ps -efa`
- Also can get real-time view of what's going on in a system with `top`



Common Commands

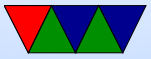
- `ls` : list files
  `ls -la` : list long output, show all (hidden) files. on Linux any file starting with . is hidden
  `ls -la /etc` : list all in /etc directory
  `ls *.gz` : show all ending in gz. * and ? are wildcards and can be used as regular expressions.

- `cd DIR` : change directories (folders)
  `cd ..` : go to parent directory
  `cd .` : go to current directory



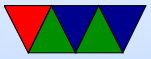
`cd /` : go to root directory
`cd ~` : go to home directory

- `cat FILE` – dump file to screen (originally used to conCATenate files together but more commonly used to list files)
- `more / less` – list contents of file but lets you scroll through them. less more advanced version of more
- `exit / logout / control-D` – log out of the machine
- `df / du` – show disk space



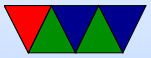
`df -h` pretty-prints it

- `man command` – show documentation (manual) for a command. For example `man ls`
- `rm` remove file. CAREFUL! Especially famous `rm -rf`. In general on Linux you cannot undo a remove.
- `cp` copy file. CAREFUL! By default will overwrite the destination without prompting you.
- `mv` move file. CAREFUL! Can overwite!
  `mv -i` will prompt before overwrite



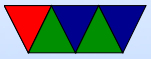
- `tar` create archive file `tar cvf output.tar dir`
  `tar xzvf output.tar.gz` uncompresses a .tar.gz file

- `gzip / gunzip / bzip2 / bunzip2` compress/uncompre
  a file. gzip and bzip2 are two common formats, many
  more exist

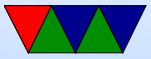


Compiler / Devel Commands

- `make` – build a file based on list of dependencies in Makefile
- `gcc` – C compiler. Simplest something like this: `gcc -O2 -Wall -o hello hello.c`
- `g++` C++ `gfortran` Fortran
- `as`, `ld` – assembler and linker
- `gdb` – debugger

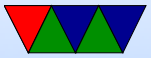


- `strace` – list system calls
- `git` – source code management

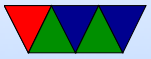


Other Commands

- `shutdown` – used to shutdown / reboot
- `last` – list last people to log in
- `su` / `sudo` – switch to root, run command as root
- `uptime` – how long machine has been up
- `date` – show the date
  as root you can use `date -s` to set the date

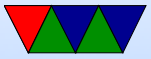


- `whoami` – who are you
- `write` / `wall` / `talk` – write to other users
- `finger` – get info on other users
- `w` / `who` – see who is logged in
- `wc` – count words/bytes/lines in a file
- `dmesg` – print system and boot messages
- `ln` – link files together, sort of like a shortcut

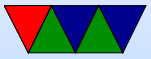


`ln -s goodbye.c hello.c` – symbolic link. also hard links

- `dd` – move disk blocks around, often used for creating disk images
- `mount` / `umount` – mount or unmount filesystems
- `mkfs.ext3` – make new filesystem
- `e2fsck` – filesystem check
- `ifconfig` / `route` – show and setup network config



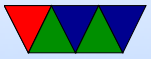
- `dpkg` / `apt-get update/upgrade/install` – Debian only package management
- `ssh` / `scp` – log into other machines, copy files remotely
- `lynx` – text-based web browser
- `reset` – clear the screen and reset settings (useful if you accidentally cat a binary file and end up with a screenful of garbage). Control-L also refreshes the screen
- `linux_logo` – my program



Editing files

Linux and UNIX have many, many editors available. Most famous are vi and emacs. On our board using nano might be easiest.

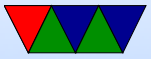
- `nano` – a simple text editor.
  `nano FILENAME` – edit a filename
  It shows the commands you can do at the bottom. `^O` means press control-O
  control-O : writes
  control-X : exits



control-W : searches

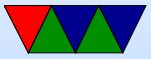
control-\: search and replace

control-C : prints line number



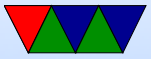
Redirection and Pipes

- redirect to a file : `ls > output`
- redirect from a file : `wc < output`
- pipe from one command to another : `ls | wc`, `dmesg | less`
- re-direct stderr : `strace 2> output`



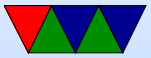
Suspend/Resume

- Press control-C to kill a job
- Press control-Z to suspend a job
- Type `bg` to continue it in the background
- Type `fg` to resume it (bring to foreground)
- Run with `&` to put in background to start with. (ie, `mpg123 music.mp3 &`).



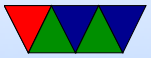
Permissions

- user, group – use chgrp
- read/write/execute – use chmod



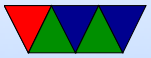
Shell Scripts

- Create a list of files in a dir
- Start with the shell, `#/bin/sh` (or perl, etc)
- Make executable `chmod +x myfile`



Command Line History

- Can press “tab” to auto-complete a command
- Can press “up arrow” to re-use previous commands
- Can use “control-R” to search for previous commands



Environment Variables

- `env`
- Varies from shell to shell.
- `export TERM=vt102`
- PATH, and why "." isn't in it. This is why you have to run self-compiled binaries as `./blah`