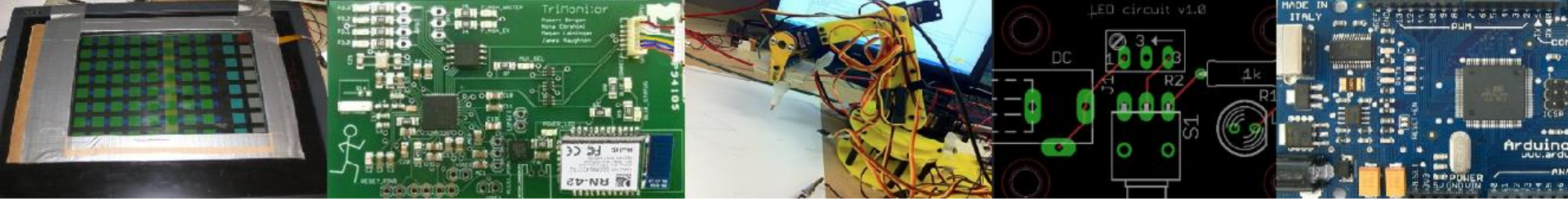
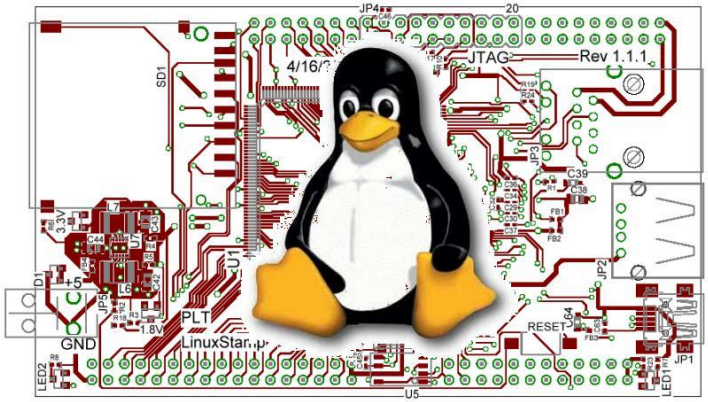


EECS 473
Advanced Embedded Systems

Lecture 5
Evaluation boards & GPL

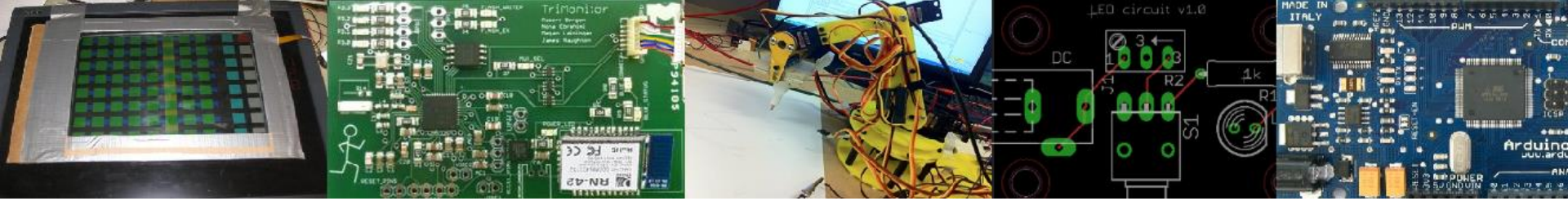
Start on Linux (time allowing)



MICHIGAN

Outline

- FreeRTOS finish up (10 minutes)
- Off-the-shelf boards (20 minutes)
    - Why they are useful
    - An example
- Copyright and the Gnu Public License (20 minutes)
  - What it is, and why it matters for embedded engineers
- Start on Embedded Linux
  - Introduction only (if that) Details next lecture.



FreeRTOS

Finishing up

MICHIGAN

Creating a task: example

```
int main( void )
{
    /* Create one of the two tasks.  Note that a real application should check
    the return value of the xTaskCreate() call to ensure the task was created
    successfully. */
    xTaskCreate(    vTask1,  /* Pointer to the function that implements the task. */
                    "Task 1",/* Text name for the task.  This is to facilitate
                             debugging only. */
                    1000,    /* Stack depth - most small microcontrollers will use
                             much less stack than this. */
                    NULL,    /* We are not using the task parameter. */
                    1,       /* This task will run at priority 1. */
                    NULL );  /* We are not going to use the task handle. */

    /* Create the other task in exactly the same way and at the same priority. */
    xTaskCreate( vTask2, "Task 2", 1000, NULL, 1, NULL );

    /* Start the scheduler so the tasks start executing. */
    vTaskStartScheduler();
```

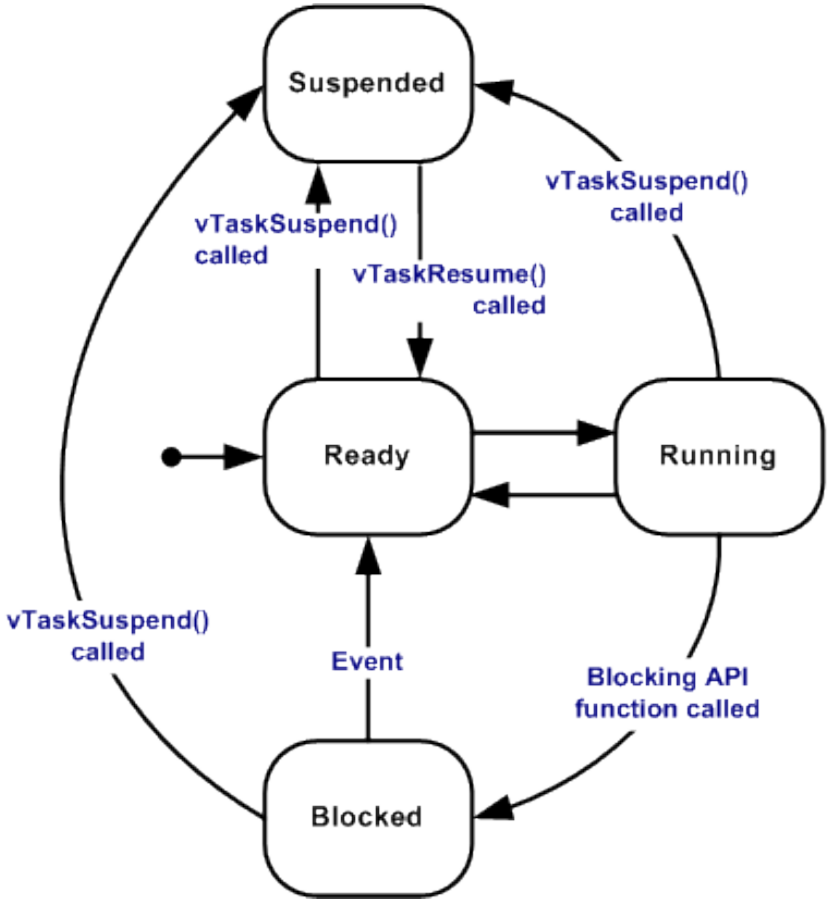
OK, I've created a task, now what?

- Task will run if there are no other tasks of higher priority
  - And if others the same priority will RR.
- But that begs the question: "How do we know if a task wants to do something or not?"
  - The previous example gave *always* wanted to run.
    - Just looping for delay (which we said was bad)
    - Instead should call **`vTaskDelay(x)`**
      - Delays current task for X "ticks" (remember those?)
    - There are a few other APIs for delaying…

Now we need an "under the hood" understanding

Task status in FreeRTOS

- **Running**
  - Task is actually executing
- **Ready**
  - Task is ready to execute but a task of equal or higher priority is Running.
- **Blocked**
  - Task is waiting for some event.
    - **Time**: if a task calls vTaskDelay() it will block until the delay period has expired.
    - **Resource**: Tasks can also block waiting for queue and semaphore events.
- **Suspended**
  - Much like blocked, but not waiting for anything.
  - Tasks will only enter or exit the suspended state when explicitly commanded to do so through the vTaskSuspend() and xTaskResume() API calls respectively.



Mostly from http://www.freertos.org/RTOS-task-states.html

Tasks: there's a lot more

- Can do all sorts of things
  - Change priority of a task
  - Delete a task
  - Suspend a task (mentioned above)
  - Get priority of a task.
- Example on the right
  - But we'll stop here…

```
void vTaskPrioritySet(
xTaskHandle pxTask,
unsigned uxNewPriority
);
```

Set the priority of any task.

- **pxTask:** Handle to the task for which the priority is being set. Passing a NULL handle results in the priority of the calling task being set.
- **uxNewPriority:** The priority to which the task will be set.

Outline

- Quick review of real-time systems
- Overview of RTOSes
  - Goals of an RTOS
  - Features you might want in an RTOS
- Learning by example: FreeRTOS
  - Introduction
  - Tasks
  - Interrupts
  - Internals (briefly)
  - What's missing?

Interrupts in FreeRTOS

- There is both a lot and a little going on here.
  - The interface mainly uses whatever the native environment uses to handle interrupts
    - This can be **very** port dependent. In Code Composer Studio you'd set it up as follows:

```
#pragma vector=PORT2_VECTOR
interrupt void prvSelectButtonInterrupt( void )
```

  - That would cause the code to run on the PORT2 interrupt.
    - Need to set that up etc. Very device specific (of course).

More: Deferred Interrupt Processing

- The best way to handle complex events triggered by interrupts is to **not** do the code in the ISR.
  - Rather create a task that is blocking on a semaphore.
    - When the interrupt happens, the ISR just sets the semaphore and exits.
      - Task can now be scheduled like any other. No need to worry about nesting interrupts (and thus interrupt priority).
      - FreeRTOS does support nested interrupts on some platforms though.
  - Semaphores implemented as one/zero-entry queue.

Semaphore example in FreeRTOS

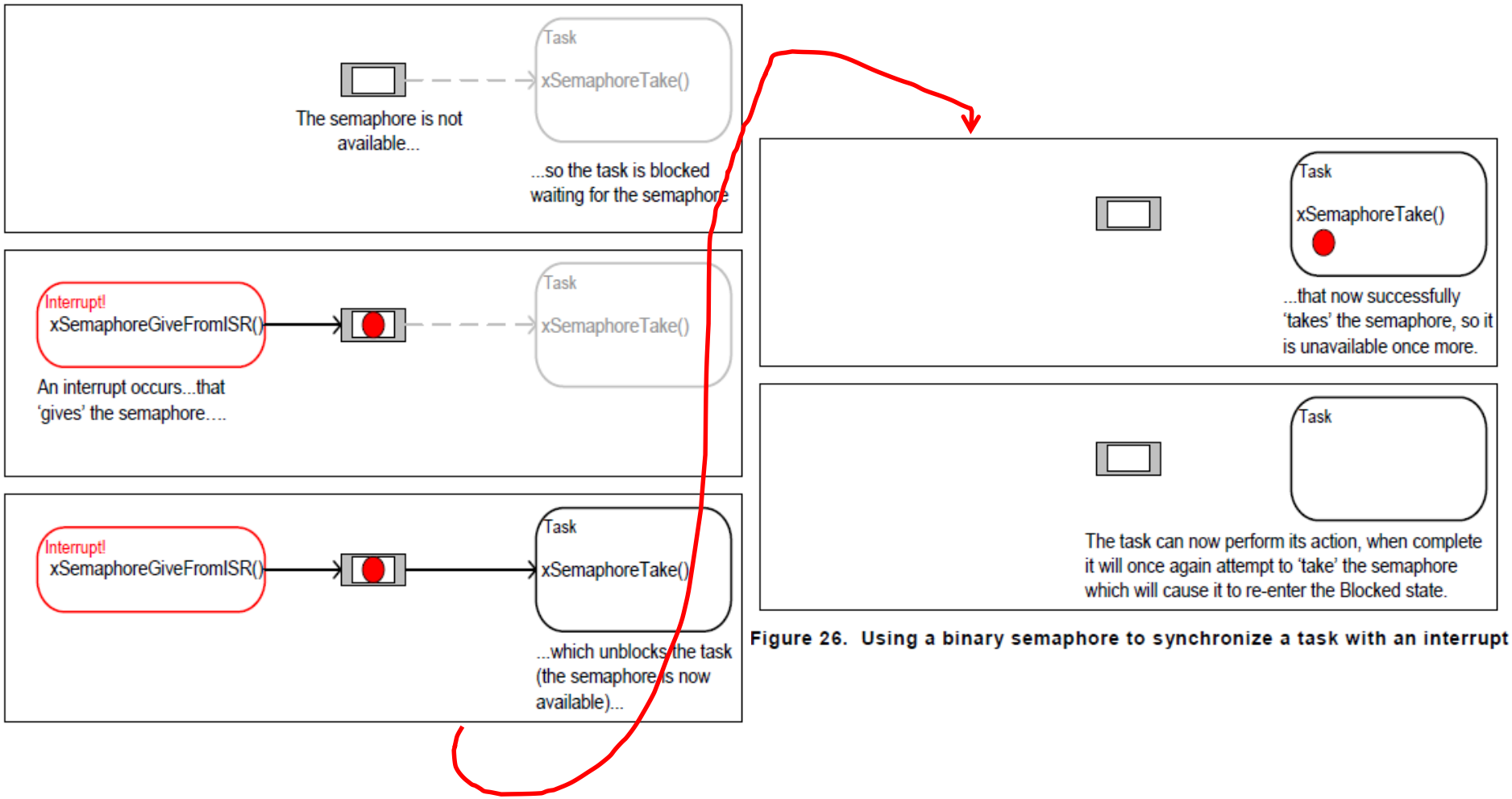


Figure 26. Using a binary semaphore to synchronize a task with an interrupt

Figure from ***Using the FreeRTOS Real Time Kernel*** (a pdf book), fair use claimed.

Semaphore take

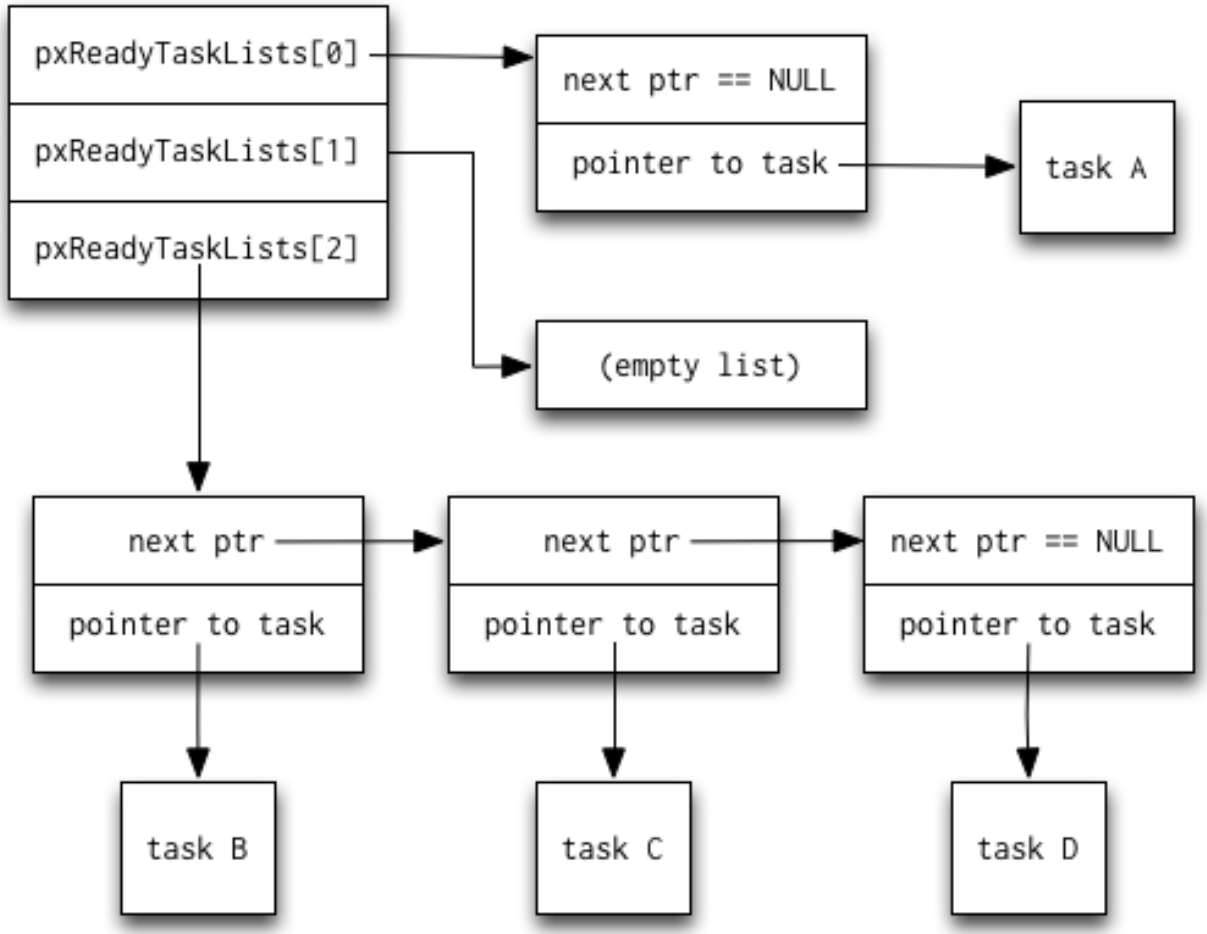
```
xSemaphoreTake(
                xSemaphoreHandle xSemaphore,
                portTickType xBlockTime
               )
```

- *Macro* to obtain a semaphore. The semaphore must have previously been created.
- **xSemaphore** A handle to the semaphore being taken - obtained when the semaphore was created.
- **xBlockTime** The time in ticks to wait for the semaphore to become available. The macro portTICK_RATE_MS can be used to convert this to a real time. A block time of zero can be used to poll the semaphore. There is a way to make this indefinitely.
- TRUE if the semaphore was obtained.
- There are a handful of variations.
  - Non-binary version, etc.

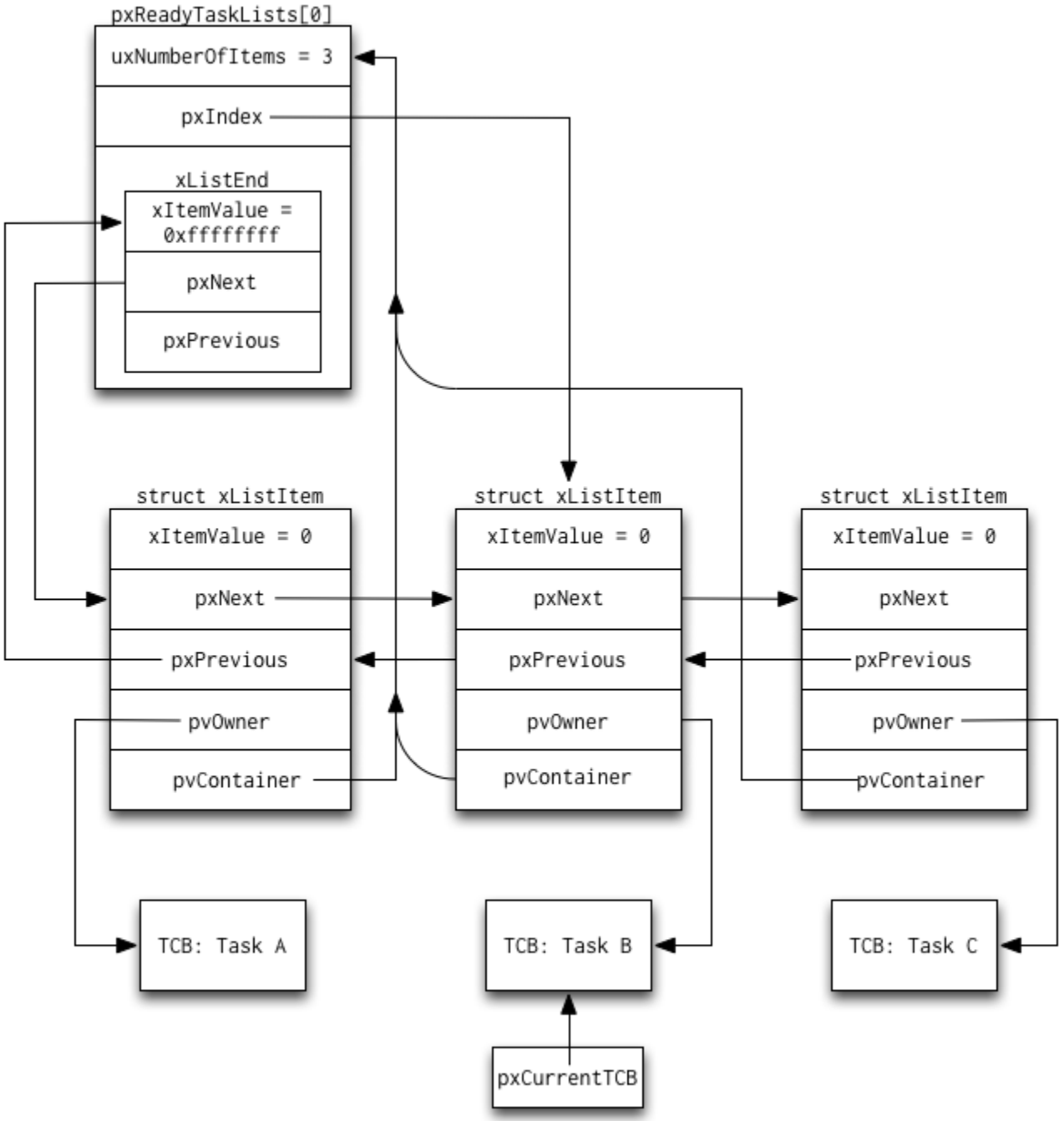
Outline

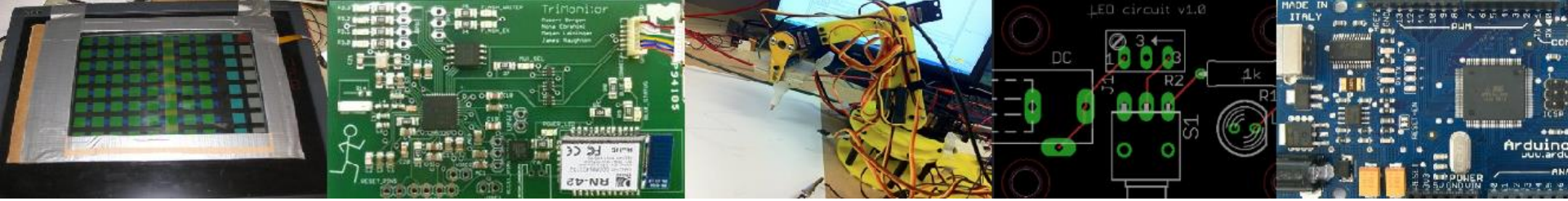
- Quick review of real-time systems
- Overview of RTOSes
  - Goals of an RTOS
  - Features you might want in an RTOS
- Learning by example: FreeRTOS
  - Introduction
  - Tasks
  - Interrupts
  - Internals (briefly)
  - What's missing?

Common data structures



This figure and the next are from http://www.aosabook.org/en/freertos.html


pxReadyTaskLists[0]
uxNumberOfItems = 3
pxIndex
xListEnd
xItemValue = 0xffffffff
pxNext
pxPrevious
struct xListItem
xItemValue = 0
pxNext
pxPrevious
pvOwner
pvContainer
struct xListItem
xItemValue = 0
pxNext
pxPrevious
pvOwner
pvContainer
struct xListItem
xItemValue = 0
pxNext
pxPrevious
pvOwner
pvContainer
TCB: Task A
TCB: Task B
TCB: Task C
pxCurrentTCB



Off-the-shelf boards

Using them in 473

MICHIGAN

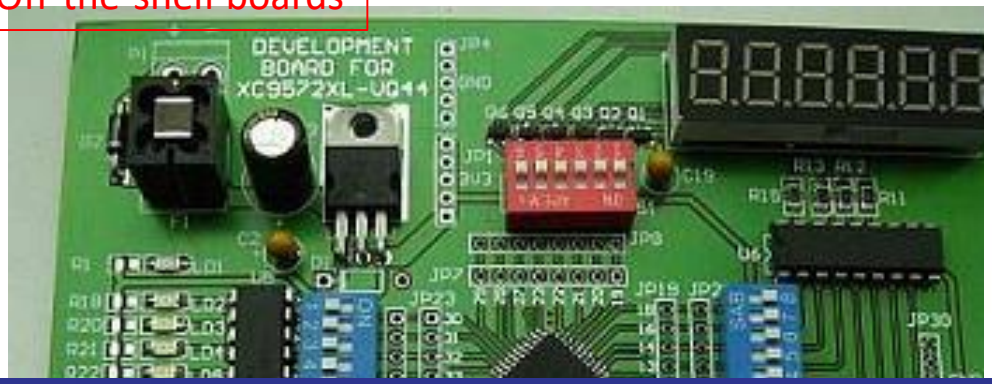
Designing an Embedded System

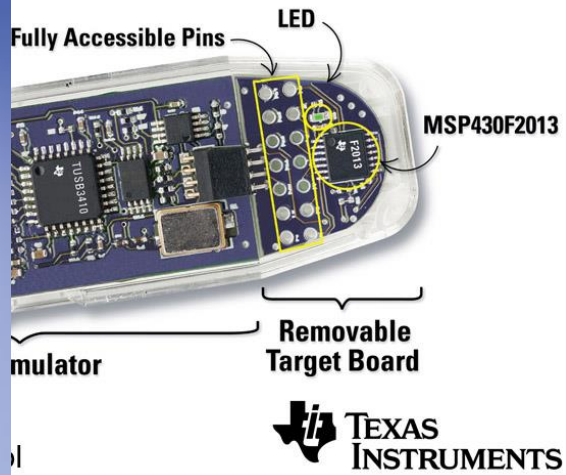
- There are a number of high-level choices for the “processor”
  - Micro-controller, SoC, desktop processor, FPGA etc.
  - But beyond that we’ve got to figure out what exactly we need.

How do we pick a platform?

- The very first question should almost always be "can I use an off-the-shelf solution?"
  - In this case, I mean a pre-built board.
- Unless you have significant quantities, you are almost always better off with an existing board.
  - No engineering cost (to you)
  - (Hopefully) no testing or debugging phase
    - Well, some testing during evaluation…
- But we'll find we can almost never use a pre-built board.
  - Why not?

Off-the-shelf boards





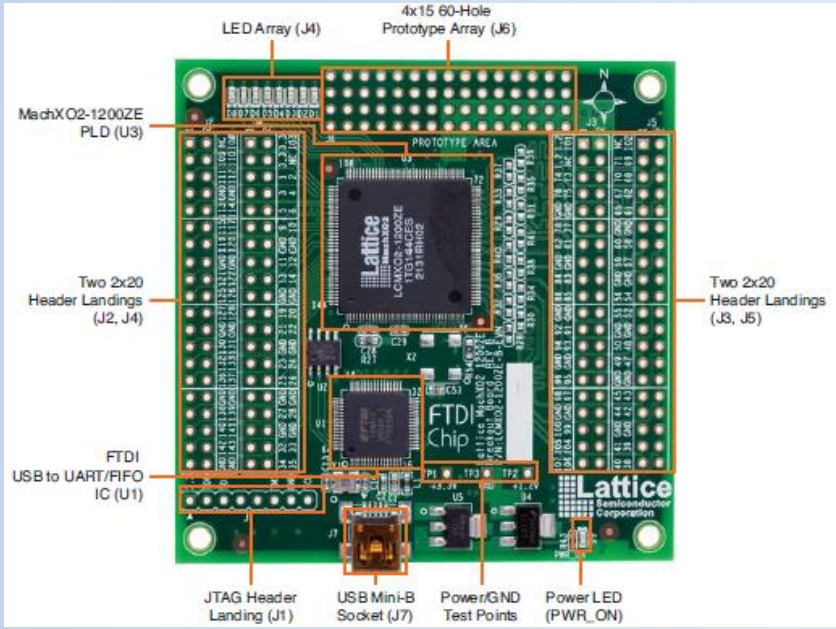
TEXAS INSTRUMENTS

Personal Example:

- I was “specing” a board a few years ago. It needed to be:
  - Be able to mostly do the 270 labs
    - Switches, buttons, LEDs, FPGA with decent free software.
      - 7-segment displays would be really nice.
    - 2000+ gates, 100+ flip-flops
      - Ideally twice that or more
  - Cheap in quantity--$20 target price at 5,000 units.
    - No external wiring or power supply other than a USB cable.
- First check is what’s out there…

Market check

- Best I could find:
  - Digilent C-Mod
    - $22 in quantities of 1, has solid FPGA.
    - Basically no I/O, requires expensive programming cable.
  - Polmaddie2
    - Has *traffic-light* LEDs!, plenty of FPGA, only USB cable
    - $80 in quantities of 100, not enough I/O.
- Asked around and found
  - MachXO2-1200ZE
    - Plenty of potential I/O (lots of headers), USB-only, $26 in quantities of 1 from Digikey (lots of potential for lower price), has LEDs.
    - No actual switches, 108 LUTs, price may be too much?



Even that board isn't ideal

- But encouraging that I might be able to design one (or have one designed) that fits the specification.
  - Spend an extra dollar on the FPGA and probably get one more than large enough.
  - Switches/7-segment displays remarkably expensive.
    - Probably $3 in parts for what I want?

So…

- There may be a board out there that does what you need.
  - Even if price/form factor/power isn't acceptable, still **really** useful for development!
  - Always look.
- You'd rather develop on an existing board.
  - You'd also like to have it as a starting point.
  - And if quantities needed are low and the off-the-shelf boards have the needed size and power characteristics, you might just use the off-the-shelf board.

But…

- All that said, you ***really*** want to *start* your design using a prebuilt board even if it isn't your final solution.
  - You can do testing, develop code, and generally get things working.
  - You *may* also get a PCB design as a starting point
    - These folks are generally using the board to sell chips. They are often happy to have you copy their PCB work.
    - But read licenses and understand any restrictions they place on you.

How do you price out a custom board?

- Good question.
  - Historically it's been a tricky to do.
    - Honestly finding reasonable prices for parts is a pain
      - Digikey is not what you want to be looking at for any kind of large-ish run. Just costs too much.
    - PCB costs are a lot easier to figure out
      - Solid PCB costs
    - And manufacturing costs are really tricky.
      - Generally require a human to give you a quote and pricing seems fairly random.

PCB quotes

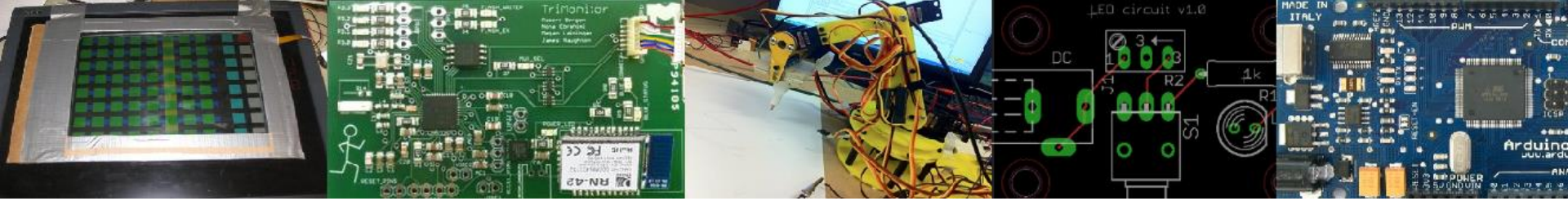
- PCBs can be quite cheap.
  - When I started this class, costs were often close to $150 for 3 to 5 boards.
  - Now can be close to $20.
    - And even free in some cases
    - Talk to GSIs about vendors.

Manufacturing quotes

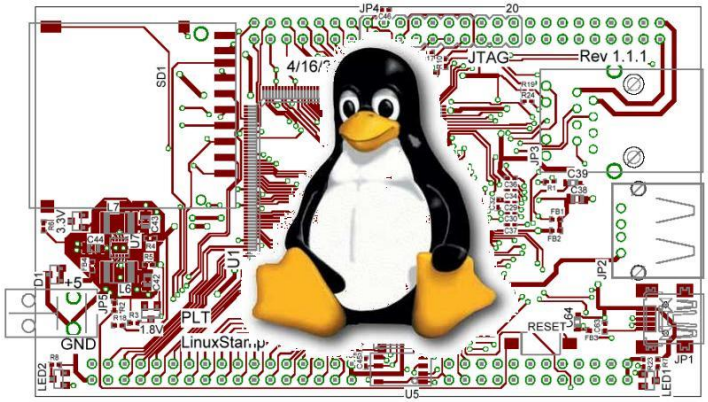
- Turns out there are companies that give automatic quotes
  - Including PCB, parts and assembly.
  - https://circuithub.com was the first (AFAIK).
- Sample board
  - An Arduino (with some extras, 16 vs 7 day ship)
    - 1 board: $255 $653
    - 10 boards: $50/each $86
    - 100 boards: $21/each $28
    - 1000 boards: $13/each N/A

Newer pricing

- PCBs of “about” Arduino complexity
  - 10: $440+ parts
  - 100: $365+parts (huh?)
  - 1000: $766+parts



Copyright, Copyleft, and Legal Issues



MICHIGAN

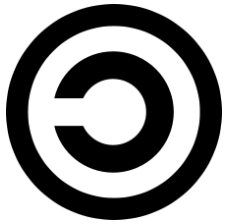
The basic copyright options

- No license:
  - Without a license, the code is copyrighted by default. People can read the code, but they have no legal right to use it. To use the code, you must contact the author directly and ask permission.
- Public domain:
  - If your code is in the public domain, anyone may use your code for any purpose whatsoever. Nothing is in the public domain by default; you have to explicitly put your work in the public domain if you want it there. Otherwise, you must be dead a long time before your work reverts to the public domain.

Text from https://blog.codinghorror.com/pick-a-license-any-license/, also includes a (too) terse summary of the common software licenses.

Linux?

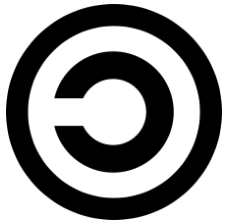
- A POSIX-compliant and widely deployed desktop/server operating system licensed under the GPL
  - POSIX
    - Unix-like environment (shell, standard programs like awk etc.)
  - Desktop OS
    - Designed for users and servers
    - Not designed for embedded systems
  - GPL
    - GNU Public License. May mean you need to make source code available to others.
      - First “copyleft” license.
    - Linux is licensed under GPL-2, not GPL-3.



GPL in three slides (1/3)

- A licensee of GPL v2-licensed software can:
  - copy and distribute the program's unmodified source code
  - modify the program's source code and distribute the modified source
  - distribute compiled versions of the program, both modified and unmodified
- Provided that:
  - all distributed copies (modified or not) carry a copyright notice and exclusion of warranty
  - all modified copies are distributed under the GPL v2
  - all compiled versions of the program are accompanied by the relevant source code, or a viable offer to make the relevant source code available

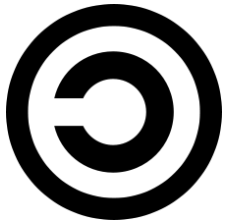
Largely taken from http://www.oss-watch.ac.uk/resources/gpl.xml



GPL in three slides (2/3)

- Some points
  - If you don't redistribute the code, you don't need to share the source.
  - You can bundle software with GPL-ed software and not have to license the bundled software.
    - "Mere aggregations" aren't impacted.
  - Loadable Kernel Modules are tricky though
    - Often we need device drivers for our application (we'll be writing them later)
    - But they touch the Linux code in a non-trivial way.
      - There is some debate about if a LKM is an aggregation or a modification of the original kernel.
      - In general there are proprietary drivers out there and even open source groups that help support said drivers.
- General theme:
  - Be sure you understand the law before you use software licensed under the GPL on a proprietary project.
    - Using gcc to compile or ddd to debug is fine, but when you are modifying the code of software licensed under the GPL you might be obligated to release your code.
- Read (or at least scan): *The Right to Read* before the midterm.
  - https://www.gnu.org/philosophy/right-to-read.en.html

GPL in three slides (3/3)



- GPL v3
  - Prevents using GPL on hardware that won't run other code ("Tivoization")
    - Though only for consumer hardware (IBM has a business model here?)
  - Addresses patents
    - Can't sue for (software?) patent on code you release.
- Lesser GPL
  - Mainly for libraries/APIs.
  - Makes it clear can use libraries in proprietary code without having to release proprietary code.

OK, one more

- gcc is GPL v3
  - But has a runtime exception.
  - When you use GCC to compile a program, GCC may combine portions of certain GCC header files and runtime libraries with the compiled program. The purpose of this Exception is to allow compilation of non-GPL (including proprietary) programs to use, in this way, the header files and runtime libraries covered by this Exception.
- Linux is GPL v2
  - Fear of limiting DRM and private keys keeps them away from v3.
- There is a short preamble:
  - This copyright does *not* cover user programs that use kernel services by normal system calls - this is merely considered normal use of the kernel, and does *not* fall under the heading of "derived work".

Another common License: **Creative Commons**

- These are basically a free *configurable* license.
  - Attribution (**by**)
    - Must give author credit
  - ShareAlike (**sa**)
    - Like GPL, they must distribute any changes.
  - NonCommercial (**nc**)
    - Only for non-commercial
  - NoDerivatives (**nd**)
    - Can't make changes.
- Idea is, someone smart wrote the words to do what you probably want to do.
  - You still hold ownership, so if they want a different license, they can come to you to negotiate.
- Wikipedia is CC BY-SA

Creative Commons is generally a poor license for code but still used for that.
See https://creativecommons.org/faq/
But quite useful for documentation and databases

MIT license

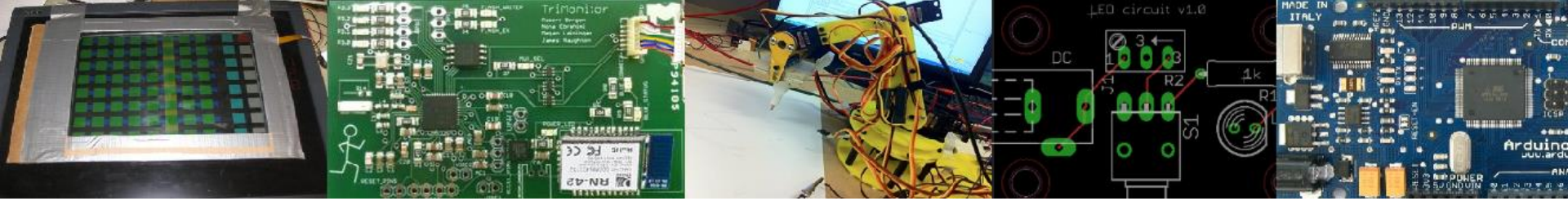
- Permission is hereby granted, free of charge, to any person obtaining a copy of this software and associated documentation files (the "Software"), to deal in the Software without restriction, including without limitation the rights to use, copy, modify, merge, publish, distribute, sublicense, and/or sell copies of the Software, and to permit persons to whom the Software is furnished to do so, subject to the following conditions:
- The above copyright notice and this permission notice shall be included in all copies or substantial portions of the Software.
- (And then a disclaimer of liability)

- Basically, just grants permission.
  - Can add to a closed-source product, etc.
  - More-or-less public domain with the disclaimer of liability being required in any place the code is used.

Fair use

- Fair use lets you use copyrighted works
  - But in general there are guidelines for what you can use.
- Four factors are:
  - the purpose and character of your use
    - Transformative or not (parody falls here)
  - the nature of the copyrighted work
    - Published/not published, factual works.
  - the amount and substantiality of the portion taken
    - The amount taken and the % taken both play a role.
  - the effect of the use upon the potential market
    - If you reduce the market for the original work, that's a problem.
- Fair use in code is tricky at best.
  - APIs sometimes fit here (see Oracle America vs. Google Inc.)

http://fairuse.stanford.edu/overview/fair-use/four-factors/



Start on Linux

Just some context

We'll get to the stuff you need for lab 4 next time.

MICHIGAN

What is a kernel? (1/2)

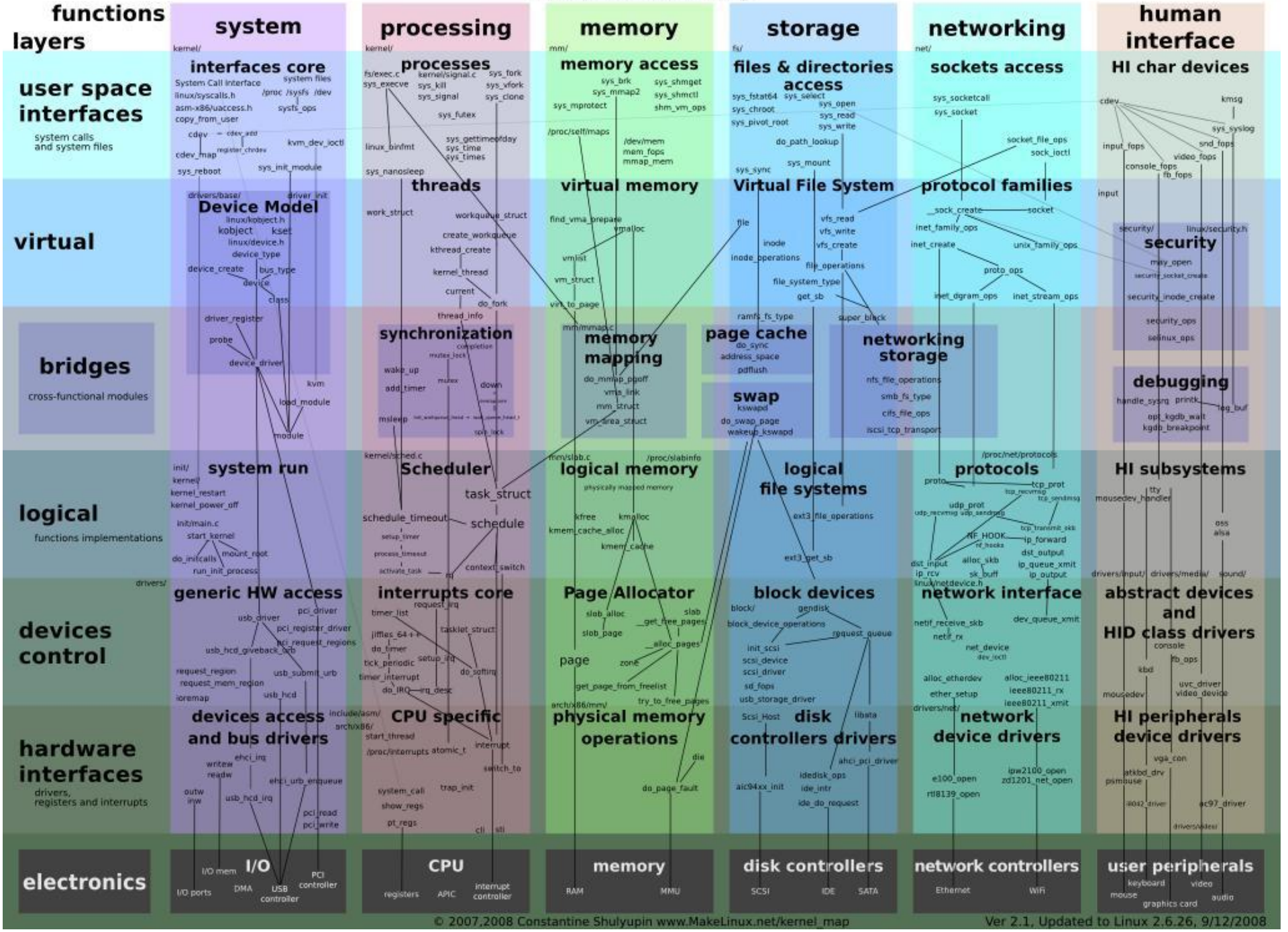
- The kernel's job is to talk to the hardware and software, and to manage the system's resources as best as possible.
  - It talks to the hardware via the drivers that are included in the kernel (or additionally installed later on in the form of a "kernel module").
    - This way, when an application wants to do something (say change the volume setting of the speakers), it can just submit that request to the kernel, and the kernel can use the driver it has for the speakers to actually change the volume.

This part taken from http://www.makeuseof.com/tag/linux-kernel-explanation-laymans-terms/

What is a kernel? (2/2)

- The kernel is highly involved in resource management.
  - It has to make sure that there is enough memory available for an application to run, as well as to place an application in the right location in memory.
  - It tries to optimize the usage of the processor so that it can complete tasks as quickly as possible. It also aims to avoid deadlocks, which are problems that completely halt the system when one application needs a resource that another application is using.
  - It's a fairly complicated circus act to coordinate all of those things, but it needs to be done and that's what the kernel is for.

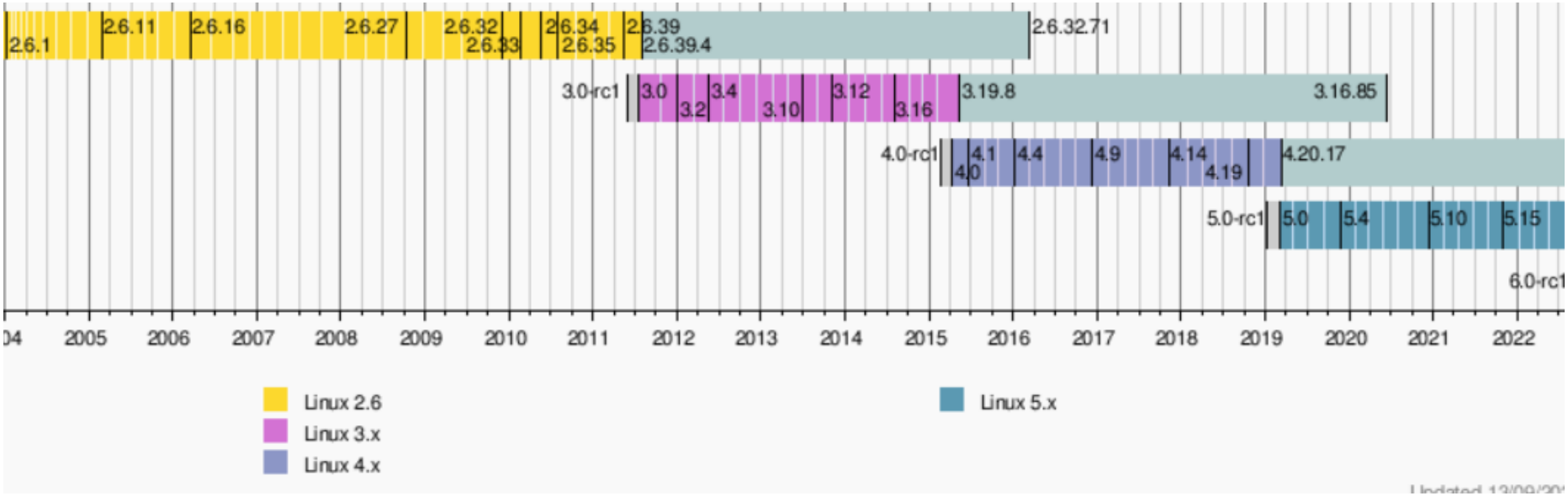
Linux kernel map



Ver 2.1, Updated to Linux 2.6.26, 9/12/2008

Background: versions

Linux Kernel history



- 3.0 isn't a radical change from 2.6, instead a 2.6 upgrade was move to 3.0 for Linux's 20th anniversary.
  - **"So what are the big changes?** NOTHING. Absolutely nothing. Sure, we have the usual two thirds driver changes, and a lot of random fixes, but the point is that 3.0 is *just* about renumbering…" [https://lkml.org/lkml/2011/5/29/204]
  - And 4.0 is more of the same.
  - http://www.itworld.com/article/2887558/linus-torvalds-bumps-linux-kernel-to-version-4x.html

I'd like to point out (yet again) that we don't do feature-based releases, and that "5.0" doesn't mean anything more than that the 4.x numbers started getting big enough that I ran out of fingers and toes.

Linus Torvalds

Figure from "Linux kernel version history" article on Wikipedia.

What version am I working with?

- If running, use “uname” command
  - “uname -a” for all information
- If looking at source
  - First few lines of the top-level Makefile will tell you.

How do I download and build the kernel?

- Use git.
- Typing “make” with no target at the top-level should build the kernel.
  - Need gcc installed (no other compiler will do).
  - Should generate an ELF file called “vmlinux”
    - But lots of configuration stuff

Kernel configuration

- There is a file, “.config” which drives the build.
  - It determines the functionality (including cross-compling?) of the kernel.
    - Things like USB support, etc.
  - It is setup in any number of ways.
    - The default is to ask a huge number of questions.
    - There are editors and defaults you can use instead.
      - **`make defconfig`** should just do all defaults for example.
    - **`make help`** should give a solid overview of options
- The .config file scheme has some problems.
  - It is easy to miss, as files that start with a “.” (dot) are hidden files in Linux ("**`ls -a`**" will show them)
  - It is easy to blow away.
    - **`make distclean`** will delete it for example…

Linux user basics--shell

- You have a *shell* which handles user commands
  - May just search for an executable file (application) in certain locations
  - Allows for moving data between those applications.
    - Pipes etc.
  - Is itself a programming language.
  - There are many of these (bash, sh, tcsh, csh, ksh, zsh)
    - Most have very similar interfaces (type application name, it runs), but the programming language part varies quite a bit.
    - Geek humor:
      - **sh** is called the Bourne shell, written by Stephen Bourne
      - **bash**, often treated as an upgrade to **sh**, is the "Bourne again shell"

Linux user basics—file systems

- Linux supports a huge variety of file systems.
  - But they have some commonalities.
- Pretty much a standard directory structure with each directory holding either other directories or files.
  - Each file and directory has a set of permissions.
    - One owner (a single user)
    - One group (a list of users who may have special access)
    - There are three permissions, read, write and execute
      - Specified for owner, group, and world.
- There are also links (hard and soft)
  - So rather than copying files I can point to them.

FHS: File System Hierarchy Standard

- There is a standard for laying out file systems
  - Part of this is the standard top-level directories

TABLE 6-1 Top-Level Directories

Directory	Contents
`bin`	Binary executables, usable by all users on the system[1]
`dev`	Device nodes (see Chapter 8, "Device Driver Basics")
`etc`	Local system configuration files
`home`	User account files
`lib`	System libraries, such as the standard C library and many others
`sbin`	Binary executables usually reserved for superuser accounts on the system
`tmp`	Temporary files
`usr`	A secondary file system hierarchy for application programs, usually read-only
`var`	Contains variable files, such as system logs and temporary configuration files

A “minimal” file system

LISTING 6-1 Contents of a Minimal Root File S

```
.
|-- bin
|   |-- busybox
|   `-- sh -> busybox
|-- dev
|   `-- console
|-- etc
|   `-- init.d
|       `-- rcS
`-- lib
    |-- ld-2.3.2.so
    |-- ld-linux.so.2 -> ld-2.3.2.so
    |-- libc-2.3.2.so
    `-- libc.so.6 -> libc-2.3.2.so

5 directories, 8 files
```

What makes a Linux install "embedded"?

- It's one of those poorly defined terms, but in general it will have one or more of the following
  - small footprint
  - flash files system
  - real-time extensions of some sort

Small footprint--Busybox

addgroup, adduser, ar, ash, awk, basename, blkid, bunzip2, bzcat, cat, chattr, chgrp, chmod, chown, chpasswd, chroot, chvt, clear, cmp, cp, cpio, cryptpw, cut, date, dc, dd, deallocvt, delgroup, deluser, df, dhcprelay, diff, dirname, dmesg, dos2unix, dumpkmap, dumpleases, echo, egrep, env, expr, false, fbset, fbsplash, fdisk, fgrep, find, free, freeramdisk, fsck, fsck.minix, fuser, getopt, getty, grep, gunzip, gzip, halt, head, hexdump, hostname, httpd, hwclock, id, ifconfig, ifdown, ifup, init, insmod, ip, kill, killall, klogd, last, less, linuxrc, ln, loadfont, loadkmap, logger, login, logname, logread, osetup, ls, lsmod, makedevs, md5sum, mdev, microcom, mkdir, mkfifo, mkfs.minix, mknod, mkswap, mktemp, modprobe, more, mount, mv, nc, netstat, nice, nohup, nslookup, od, penvt, passwd, patch, pidof, ping, ping6, pivot_root, poweroff, printf, ps, pwd, rdate, rdev, readahead, readlink, readprofile, realpath, reboot, renice, reset, rm, rmdir, rmmod, route, rtcwake, run-parts, sed, sed, setconsole, etfont,sh, showkey, sleep, sort, start-stop-daemon, strings, stty, su, sulogin, swapoff, swapon, switch_root, sync, sysctl, syslogd, tail, tar, tee, telnet, telnetd, test, tftp, time, top, touch, tr, traceroute, true, tty, udhcpc, udhcpd, umount, uname, uniq, unzip, uptime, usleep, vi, vlock, watch, wc, wget, which, who, whoami, xargs, yes, zcat

- A single executable that implements the functionality of a massive number of standard Linux utilities
  - `ls, gzip, ln, vi`. Pretty much everything you normally need.
    - Some have limited features
      - Gzip only does the basics for example.
    - Pick which utilities you want it to do
      - Can either drop support altogether or install real version if needed
    - Highly configurable (similar to Linux itself), easy to cross-compile.
- Example given is around 2MB statically compiled!

Using busybox

- Two ways to play
  - Invoke from the command line as busybox
    - **`busybox ls /`**
  - Or create a softlink to busybox and it will run as the name of that link.
    - So if you have a softlink to busybox named “ls” it will run as ls.

Links done for you

- Normally speaking, you'll use the softlink option.
  - You can get it to put in all the links for you with "make install"
    - Be darn careful you don't overwrite things locally if you are doing this on the host machine.
      - That would be bad.

```
|-- bin
|    |-- busybox
|    |-- cat -> busybox
|    |-- dmesg -> busybox
|    |-- echo -> busybox
|    |-- hostname -> busybox
|    |-- ls -> busybox
|    |-- ps -> busybox
|    |-- pwd -> busybox
```

```
$ make CONFIG_PREFIX=/mnt/remote install
  /mnt/remote/bin/ash -> busybox
  /mnt/remote/bin/cat -> busybox
  /mnt/remote/bin/chgrp -> busybox
  /mnt/remote/bin/chmod -> busybox
  /mnt/remote/bin/chown -> busybox
...
  /mnt/remote/usr/bin/xargs -> ../../bin/busybox
  /mnt/remote/usr/bin/yes -> ../../bin/busybox
  /mnt/remote/usr/sbin/chroot -> ../../bin/busybox
```

System initialization

- Busybot can also be “init”
  - But it’s a simpler/different version than the init material covered above.
  - More “bash” like

```
#!/bin/sh
echo "Mounting proc"
mount -t proc /proc /proc

echo "Starting system loggers"
syslogd
klogd

echo "Configuring loopback
   interface"
ifconfig lo 127.0.0.1

echo "Starting inetd"
xinetd

# start a shell
busybox sh
```

BusyBox Summary

- BusyBox is a powerful tool for embedded systems that replaces many common Linux utilities in a single multicall binary.
- BusyBox can significantly reduce the size of your root file system image.
- Configuring BusyBox is straightforward, using an interface similar to that used for Linux configuration.
- System initialization is possible but somewhat different with BusyBox