ECE 471 – Embedded Systems
Lecture 9

Vince Weaver

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

vincent.weaver@maine.edu

27 September 2016

Announcements

- HW#4 was posted.
- Permissions!
  Unless your user is configured to have gpio permissions you'll have to run as root or use sudo. raspbian there's a "gpio" group which has permissions `sudo addgroup vince gpio`
  `udev` is responsible for updating permissions as the files are created and it can take a fraction of a second to detect and update.

This might not work if you have an older version of Raspbian

Homework 3

- Should do HW, even if you only do the short-answer part. Good practice for midterm
- Comment code!
  Make sure comment makes sense, especially if cut and pasting.
- `print_ number()` code
  - No conversion to binary, number is in binary in register.
  - The divide by 10 code is almost more interesting.

– Good to be able to look at code and see what doing. Reverse engineering, but also debugging code you don't have the source to.

```
print_number:
        push    {r10,LR}        // Save registers
        ldr     r10,=buffer     // what does = mean?  where is buffer?
        add     r10,r10,#10     // why 10 bytes?

divide:
        bl      divide_by_10    // why no div instruction?
        add     r8,r8,#0x30     // why add 0x30?
        strb    r8,[r10],#-1    // why moving backwards?
        adds    r0,r7,#0        //
        bne     divide          //

write_out:
        add     r1,r10,#1       // why adjust pointer?

        bl      print_string    //

        pop     {r10,LR}        //
        mov     pc,lr           //
```

how would you convert to hex? Why 10 chars reserved?

```
divide_by_10:
        ldr     r4,=429496730           @ 1/10 * 2^32
        sub     r5,r0,r0,lsr #30
        umull   r8,r7,r4,r5             @ {r8,r7}=r4*r5
        mov     r4,#10                  @ calculate remainder
        mul     r8,r7,r4
        sub     r8,r0,r8
        mov     pc,lr
```

- strlen code example, many ways to do this

```
        mov     r2,#0
print_loop:
```

```
ldrb    r0,[r1,r2]
add     r2,r2,#1
cmp     r0,#0
bne     print_loop
```

- THUMB code should have been less.
  You need to run `strip` on this to see it. Why?
  Debug info, including extra thumb debug as well as the longer filename.
  You can use `readelf -a` and `readelf -s` to see the space the various segments take up.
  Look at executables, *not* the C source code.

arch	unstripped	stripped
arm32	1444	624
thumb	1460	600
thumb2		596
C	6k	2k

You would think THUMB2 would be much smaller, but the assembler makes some poor decisions about wide/narrow instructions.

C code is larger, but also remember to include the C library:

```
ls -lart /lib/arm-linux-gnueabihf/libc-2.19.so
-rwxr-xr-x 2 root root 1226392 Sep  6 01:57 /lib/arm-linux-gnueabihf/libc-2.19.so
```

There are embedded C libraries, musl, newlib, uclibc, which are much smaller and often used in embedded systems.

- Illegal instruction error usually because there are *two* calls to print string, need to make sure both are blx
- cal. Missing days. Julian to Gregorian calendar. People sad who paid weekly but paid rent monthly.
  Be careful using Google. First hit you get might be a

humor link.

Bypassing Linux to hit hardware directly

- Linux does not support things like pullups, but people have written code that will poke the relevant bits directly.
- Also useful for speed:

`http://codeandlife.com/2012/07/03/benchmarking-raspberry-pi-gpio-speed/`

sysfs/gpio	40Hz
proc/mem	2.8kHz
C Rpi "Native"	22MHz
C libbcm2835	5.4MHz
C WiringPi	4.6MHz
RPi.GPIO	70kHz
Python WiringPi	28kHz

Why Use an Operating System?

- Provides Layers of Abstraction
  - Abstract hardware: hide hardware differences. same hardware interface for classes of hardware (things like video cameras, disks, keyboards, etc) despite differing implementation details
  - Abstract software: with VM get linear address space, same system calls on all systems
  - Abstraction comes at a cost. Higher overhead, unknown timing

- Multi-tasking / Multi-user
- Security, permissions (Linus dial out onto /dev/hda)
- Common code in kernel and libraries, no need to re-invent

What's included with an OS

- kernel / drivers – Linux definition
- also system libraries – Solaris definition
- low-level utils / software / GUI – Windows definition
  Web Browser included?
- Linux usually makes distinction between the OS Kernel and distribution. OSX/Windows usually doesn't.

Operating Systems Types

- Monolithic kernel – everything in one big address space. Something goes wrong, lose it all. Faster
- Microkernel – separate parts that communicate by message passing. can restart independently. Slower.
- Microkernels were supposed to take over the world. Didn't happen. (GNU Hurd?)
- Famous Torvalds (Linux) vs Tannenbaum (Minix) flamewar

Common Desktop/Server Operating Systems

- Windows
- OSX
- Linux
- FreeBSD / NetBSD / OpenBSD
- UNIX (Irix/Solaris/AIX/etc.)
- BeOS/Haiku

Embedded Operating Systems

- Microsoft WinCE, Windows Mobile
- Linux / Android
- VXworks – realtime OS, used on many space probes
- Apple iOS
- QNX – realtime microkernel UNIX-like OS, owned by Blackberry now
- Cisco iOS

Embedded Linux Distributions

- linaro – consortium that work on ARM software
- openwrt – small distro initially designed for wireless routers
- yocto – Linux Foundation sponsored embedded distro
- maemo – embedded distro originally by Nokia (obsolete)
- MeeGo – continuation of maemo, also obsolete

- Tizen – Follow up on MeeGo, by Samsung and Intel
- Ängstrom – Merger of various projects
- And many others. It's very easy to put together a Linux distribution

Linux/UNIX History

- UNIX invented early 70s at Bell Labs
- Widely distributed by academics
- Berkeley makes their own BSD version
- By the 90s many companies selling UNIX workstations. Expensive.
- Linus Torvalds in 1991 wanted own UNIX-like OS. Minix (which he used for development) limited to academic use

and non-free. The various BSDs caught up in lawsuit with AT&T. So he wrote his own.