ECE 531 – Advanced Operating Systems
Lecture 2

Vince Weaver

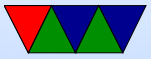
https://web.eece.maine.edu/~vweaver

vincent.weaver@maine.edu

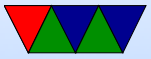
5 September 2025

Announcements

- Homework 1 will be posted, short answer
- Will distribute Pis next week
- Note: ECE574 will not be offered in the Spring

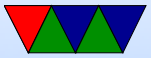


More OS Background



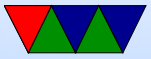
What's included with an OS

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



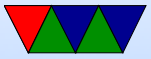
Linux Distributions

- The files, libraries, userspage applications
- RedHat/Fedora/Suse/Ubuntu/Debian
- The included applications differ, but still all use the same Linux kernel



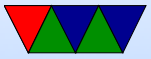
What Does Linux Kernel Provide

- Boot/initialization
- Hardware drivers
- Network (TCP/IP and others)
- Interrupts, DMA
- Multi-tasking/Job scheduling
- Virtual Memory
- Filesystems
- Security(?)
- Graphics?



Operating Systems Types

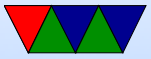
- Monolithic kernel – everything in one big address space. Something goes wrong, lose it all. Faster
- Microkernel – separate parts that communicate by message passing. can restart independently. Slower.
- Microkernels were supposed to take over the world. Didn't happen. (GNU Hurd?)
- Famous Torvalds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



What Language do you write OS in?

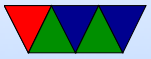
A System Programming Language

- Assembly Language? (why not)
- C?
- C++? (why not)
- Java? Python? Javascript?
- Rust? Go? Zig?



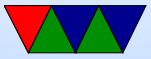
Before you can run an Operating System you first have to Compile it from Source Code

- Have you ever built your own kernel?
- Have you ever built your own C-library?
- Have you ever built your entire userspace? web-browser?
- How about building a compiler?



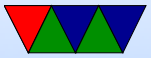
Aside on Build Systems

- Usually you don't compile code one file at a time
- There are systems that can automate or script this
- The traditional Linux/UNIX way is with a tool called "make" and "Makefiles" that describe dependencies and how to build the code
- If you use an IDE it might have its own way of doing things
- People are constantly proposing alternatives, things like CMake but they all have their own issues

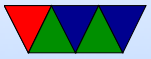


Linux Kernel Source Code

- What language is the kernel written in?
- C and assembly (with some helper shell/perl scripts)
- Why C?
  - Low-level, close to hardware (portable assembler)
  - Fast
  - Historical
  - Downsides: buggy, security bugs
- Why not C++ (or Java or Rust or Go)
  - Historical reasons, cost to change

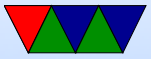


- Overhead/speed (is 10-15% slower OK?)
- Higher level languages harder to predict (operator overload, exception handling, garbage collection, etc)
- Recently there's been a push to allow writing parts of the kernel in Rust. Ongoing.



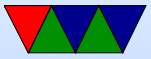
Large Open-source Project Development

- Linux is a prime example
- Communications: mailing list, forum, etc (Linux: linux-kernel)
- Way to submit changes: git pull requests, patches made with diff tool
- Once a project gets large enough it will need to have rules

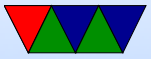


Source Code Management

- Allows tracking changes to source code, authorship, commit messages describing changes to code, branches, etc
- Allow debugging via bisect
- Historical: SCCS, CVS, subversion, mercurial
- git
  - Linux lasted long time w/o SCM
  - Linus got burned out. McVoy came up with (proprietary) bitkeeper

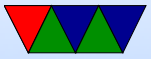


- bitkeeper hit limits and also trouble with users trying to reverse-engineer
- Linus got fed up and took a few weeks to invent git



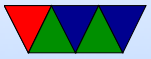
Linux kernel releases

- Currently 6.16 (explain modern version numbering)
- Linus Torvalds releases kernel
- Spends next two weeks in “merge window” merging all the well-tested patches that have accumulated. Then releases -rc1
- Series of -rc as things are tested
- After -rc7 or -rc8 releases final version. Repeat
- Distributions or volunteers will often maintain older 'stable' versions that aren't quite as cutting edge



Linux kernel size

- Git checkout on my machine is 5.5G (before building)
- After building, 15G
- 73k files, 31 architectures (these numbers are older)
- For comparison, September 1991, Linux 0.01
  512kB disk, 100 files, 1 architecture



Building Linux Kernel by Hand

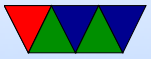
- Check out with git or download tarball:

  `git clone git://git.kernel.org/pub/scm/linux/kernel/git/torvalds/linux.git`

  `http://www.kernel.org/pub/linux/kernel/v6.x/`

- Configure. (complicated and verbose)
  `make config` or `make menuconfig`
  Also can copy existing `.config` and run `make oldconfig`

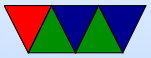
- Compile.



`make`

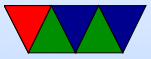
What does `make -j 8` do differently?

- Make the modules.
  `make modules`
- `sudo make modules_install`
- `sudo make install` or manually copy bzImage to boot, update boot loader
- Cleanup, `make clean` and `make mrproper`



Building Linux Automated

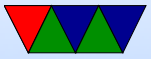
- If in a distro there are other commands to building a package.
- For example on Debian `make-kpkg --initrd --rootcmd fakeroot kernel_image`
- Then `dpkg -i` to install; easier to track



Overhead (i.e. why not to do it natively on a Pi)

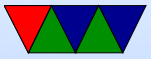
- Size – clean git source tree (x86) 1.8GB, compiled with kernel, 2.5GB, compiled kernel with debug features (x86), 12GB!!!
  Tarball version with compiled kernel (ARM) 1.5GB

- Time to compile – minutes (on fast multicore x86 machine) to hours (18 hours or so on Pi-B+)



Developing Linux

- Fun!
- Despite news reports, odds of getting flamed by Linus (or even have him realize you exist) are very low.
- Can be tedious, can take months to get a change committed
- Much of low-hanging fruit already gone
- Code is not really all that well commented
- Might be stuck bisecting for days



Linux on the Pi

- Mainline kernel, bcm2835/bcm2836 tree
  Missing some features
- Raspberry-pi foundation bcm2708/bcm2709 tree
  More complete, not upstream
- Why everything not upstream? Common problem, especially on ARM. Getting upstream is hard, high standards. Takes patience and time, small one-off ARM boards do not have the resources for the process.