

Introduction to the Linux Kernel Source and Layout

arch/, mm/, fs/, oh my!

Muli Ben-Yehuda

`mulix@mulix.org`

IBM Haifa Research Labs

introduction

- The Linux kernel is written in C, and licensed under the GPL
- A handful of assembly thrown in
- Two development trees, stable and experimental
- This talk is about the stable branch, currently at 2.4.22-pre5.
- Several millions lines of code
- We will survey the kernel source, what functionality can be found where?

build system and general info

- Top level Makefile and Rules.Make
- Documentation/Configure.help for the configuration documentation
- README - Read it!
- REPORTING-BUGS - because all code sucks

kernel documentation

- Arguably, the most important directory... contains, as you probably guessed, the documentation.
- Unfortunately, sometimes out of date. Google is your friend for updated documentation.
- Important files:
 - 00-INDEX
 - BUG-HUNTING
 - CodingStyle
 - SubmittingBugs / SubmittingPatches
 - modules.txt
 - spinlocks.txt

architecture dependant

- The arch/ directory contains many subdirectories, one for each architecture Linux supports. 18 for 2.4.22-pre5, not including sub-architectures.
- 90% of the people only care about i386, which can be found at arch/i386/
- boot - assembly files, related to booting Linux
- lib - optimized routines for common tasks (e.g. memcpy)
- mm - i386 specific memory management
- kernel - the bulk of the i386 code, including IRQ handling, processes, signals and pci support, to name a few areas.

drivers

- The bulk of the kernel's code
- net/, sound/, usb/, atm/, ide/, scsi/, etc, etc
- character device drivers live in char/, including /dev/null and /dev/zero
- block device drivers live in block/
- drivers are actually one of the more important parts of the kernel from a user point of view, but some suffer from serious lack of maintenance

fs

- fs has filesystem code
- includes the Linux vfs “virtual file system” layer, which is often held as an example of good kernel code
- also includes the binary formats for executable file support, in `binfmt_*`
- the majority of Linux users use `ext2` and `ext3`
- local file systems, distributed file systems (`nfs`, `intermezzo`), old and grungy file systems (`msdos`, anyone?)

include

- header files live here
- asm-* include architecture specific header files (complement arch/)
- the 'asm' symbolic link is created as part of the build process depending on which architecture we are compiling for
- grep here first when looking for an API or a constant
- most important (relevant) header files live in include/linux

mm

In my humble opinion, the most interesting kernel code...

- virtual memory management, including swap in and swap out, shared memory support, slab caches.
- definitely not trivial to understand
- one of the few places where micro optimizations can be considered
- only 15,000 lines of code!

kernel

- various files which don't belong elsewhere
- scheduler!
- various process related system calls: ptrace, exit
- kernel infrastructure such as softirqs, printk
- capability (security) support
- fork - the point where all processes are created

the rest

- lib - generic library support routines
- net - networking support, ipv4 and v6, tcp, other esoteric protocols
- ipc - SYSV interprocess communications mechanisms
- init - kernel initialization and startup
- crypto (new addition) - cryptographic support
- scripts - various scripts, some used for the build system