

Ch 1. Overview

Oct. 27, 2006

Hyun-koo Jee

Senior researcher, DWE

Contents

- 1.1. History
- 1.2. Standards
- 1.3. Free Software & Open Source
- 1.4. A Quick Survey of Linux Distro.'s
- 1.5. Kernel Release Info.
- 1.6. Linux on Power
- 1.7. What is an OS?
- 1.8. Kernel Organization
- 1.9. Overview of Linux Kernel
- 1.10. Portability & Architecture Dependence

Contents

1.1. History

1.2. Standards

1.3. Free Software & Open Source

1.4. A Quick Survey of Linux Distro.'s

1.5. Kernel Release Info.

1.6. Linux on Power

1.7. What is an OS?

1.8. Kernel Organization

1.9. Overview of Linux Kernel

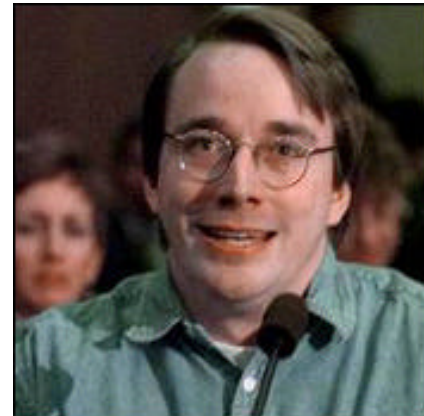
1.10. Portability & Architecture Dependence

1.1. History (1/2)

- MULTICS (1960's)
- UNIX (1970)
 - By Ken Thompson
 - Began with stripping-down of MULTICS
- UNIX in C (1973)
 - By Ken Thompson & Dennis Ritchie
 - A Kernel in High-Level Language?!
Simple, Portable

1.1. History (2/2)

- UNIX Variants (1980's)
 - BSD (UC Berkeley)
(FreeBSD, NetBSD, OpenBSD)
 - System V (AT&T)
(AIX by IBM)
- Linux (1991)
 - A UNIX Clone
 - Originally by Linus Torvalds, for 80386
 - Began as an extension of Minix (by Andrew Tannenbaum)



Linus Torvalds

*Hav U ever heard of
the debate B2in
Linus & Andy?*

Contents

1.1. History

1.2. Standards

1.3. Free Software & Open Source

1.4. A Quick Survey of Linux Distro.'s

1.5. Kernel Release Info.

1.6. Linux on Power

1.7. What is an OS?

1.8. Kernel Organization

1.9. Overview of Linux Kernel

1.10. Portability & Architecture Dependence

1.2. Standards

- POSIX
 - The standard spec. for UNIX API
 - Formed by IEEE
 - For the sake of Portability!
 - Ex) Written on BSD, running on Linux!

Contents

1.1. History

1.2. Standards

1.3. Free Software & Open Source

1.4. A Quick Survey of Linux Distro.'s

1.5. Kernel Release Info.

1.6. Linux on Power

1.7. What is an OS?

1.8. Kernel Organization

1.9. Overview of Linux Kernel

1.10. Portability & Architecture Dependence

1.3. Free Software & Open Source (1/2)

- Linux is an Open-Source Project!
 - GPLv2
 - See the file 'COPYING'
 - Anyone can
MODIFY, READ, REDISTRIBUTE
the source code, BUT...
- * Kernel developers don't like GPLv3. ☹️

1.3. Free Software & Open Source (2/2)

- 2 Major Open-Source Camps
(differ in ideology)
 - FSF (<http://www.fsf.org>)
 - The open-source group (<http://www.opensource.org>)

1.3. Free Software & Open Source (2/2)

- 2 Major Open-Source Camps

*"Unquestionably one of the great seminal figures of the hacker culture."
—Eric Raymond, open source evangelist and author of The Cathedral and the Bazaar*

FREE AS IN FREEDOM

RICHARD STALLMAN'S
CRUSADE FOR FREE SOFTWARE



Richard Stallman,
the founder of GNU

SAM WILLIAMS



Stallman at MIT...

*Now you see
what the typical hackers
look like?*

Contents

1.1. History

1.2. Standards

1.3. Free Software & Open Source

1.4. A Quick Survey of Linux Distro.'s

1.5. Kernel Release Info.

1.6. Linux on Power

1.7. What is an OS?

1.8. Kernel Organization

1.9. Overview of Linux Kernel

1.10. Portability & Architecture Dependence

1.4. A Quick Survey of Linux Distro.'s

- What do you mean by “Linux”:
Linux Kernel + Many Applications
- Linux Distributions
 - **Redhat/Fedora**: the most famous (esp. in Korea?)
 - **Debian**: real hackers like this!
 - **Ubuntu**: HOT!
 - **Gentoo**: compile 'em all with portage!
 - **Linux from Scratch**: do it yourself!
- * <http://www.distrowatch.com>
- * <http://osnews.com>

Contents

1.1. History

1.2. Standards

1.3. Free Software & Open Source

1.4. A Quick Survey of Linux Distro.'s

1.5. Kernel Release Info.

1.6. Linux on Power

1.7. What is an OS?

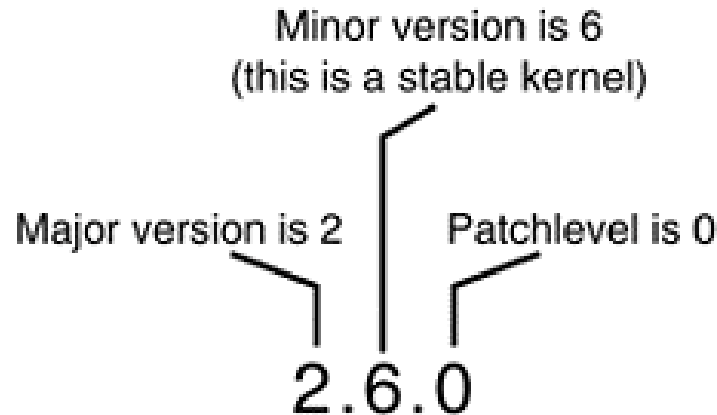
1.8. Kernel Organization

1.9. Overview of Linux Kernel

1.10. Portability & Architecture Dependence

1.5. Kernel Release Info. (1/2)

- Old Versioning Scheme
 - 2.1, 2.3, 2.5, ... : development tree
 - 2.2, 2.4, 2.6, ... : stable branch (mainline kernel)



1.5. Kernel Release Info. (1/2)

- Old Versioning Scheme
 - 2.1, 2.3, 2.5, ... : development tree
 - 2.2, 2.4, 2.6, ... : stable branch (mainline kernel)
- But, now...

Code that might normally go into a development tree is being included in the stable 2.6 tree.

Specifically, “...*the mainline kernel will be the fastest and most feature-rich kernel around, but not, necessarily, the most stable. Final stabilization is to be done by distributors (as happens now, really), but the distributors are expected to merge their patches quickly*”

[Jonathan Corbet via <http://kerneltrap.org/node/view/3513>].

1.5. Kernel Release Info. (2/2)

- Who are the kernel developers?
 - They chat via **LKML** (Linux Kernel Mailing List). See <http://vgers.kernel.org>.
 - They use **git** (a source-code version control system). See <http://kerneltrap.org/node/4982>.
 - They use **bugzilla**. See <http://bugzilla.kernel.org>.
 - Linus Torvalds decides whether a given patch can be accepted or not.
 - Wanna join? See the chapter *'Patches, Hacking, and the Community'* in [R.Love]

Contents

- 1.1. History
- 1.2. Standards
- 1.3. Free Software & Open Source
- 1.4. A Quick Survey of Linux
- 1.5. Kernel Releases and Versions
- 1.6. Linux PowerPC
- 1.7. What is an OS?
- 1.8. Kernel Organization
- 1.9. Overview of Linux Kernel
- 1.10. Portability & Architecture Dependence

We have 5 more
sections left
... in this chapter.

Q1: How can I analyze the monster called KERNEL?

**Use the source,
Luke!**

... or R.T.F.S.



The KERNEL SOURCE???

Q2: Where is it?

-Use Redhat/Fedora?

Get the SRPM file.
(kernel-***.src.rpm)

-Use Debian/Ubuntu?

apt-get install linux-source-*

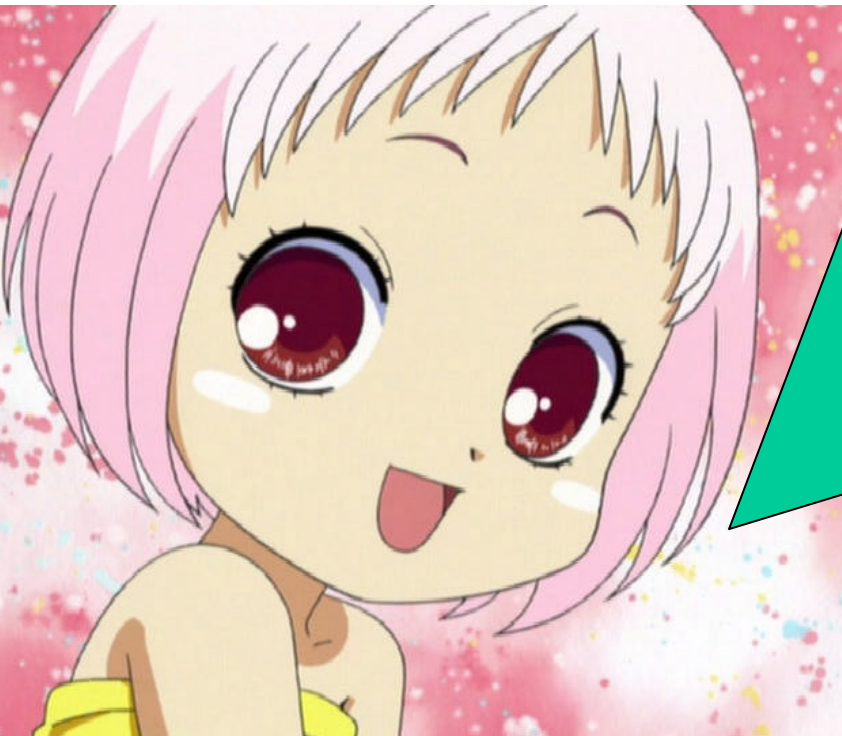
**-Wanna get the Vanilla Kernel?
(officially released by Linus)**

<http://www.kernel.org>

**-Never want to install the source
to your HDD?**

<http://lxr.linux.no>

<http://tamacom.com/tour.html>



The KERNEL SOURCE???

Q2: Where is it?

Oh,
don't forget to
check your
/usr/src/linux* first.



Ok. Now I have the source.

Q3: How can I browse it?

You have lovely tools available...

-cscope

Try 'make ARCH=x86_64 cscope'
at /usr/src/linux

-grep(with -R), find, diff(and vimdiff)

-vim + ctags

Try 'make ARCH=x86_64 tags'
at /usr/src/linux

-emacs + etags

Try 'make ARCH=x86_64 TAGS'
at /usr/src/linux

**-{Source Insight / Visual Slickedit /
...} + samba**

-kscope, {Eclipse+CDT}, ...

**-notepad? wordpad? Visual Studio?
R U serious???**

vim kata

- I use vim, because it's the fastest code-navigation tool.

– H. Jee

- Fasten your seatbelt 😊

vim kata (1/8)

- Go to tag: “:tj”, “:ts”, “:ta”
/ “:stj”, “:sts”, “:sta” / “:ptj”, “:pts”, “:pta”
 - you can use “:vert” before these commands
 - enter “ctags -R . * ” at the shell prompt
 - append “:set tags+=tags,..../tags,..../tags,..../tags” to *.vimrc*
- Go to tag under cursor: “^]”, “g]”
 - In new window: “^w]”, “^wg]”
 - In preview window: “^w}”, “^wg}” (close: “^wz”)
 - * you can enter a ‘number’ before press one of these...
- Go back: “^t”
- Go to the global definition in this file: “gD”
- Go to the local definition in this block: “gd”

vim kata (2/8)

- Find a phrase: “/”, “?”
 - you can use <up>, <down> key at the command-line
- Find next
 - same direction: “n”
 - reversed direction: “N”
- Find the word under cursor
 - forward: “*”
 - backward: “#”

vim kata (3/8)

- Autocompletion
 - word completion: “`^p`” (backward), “`^n`” (forward)
 - line completion: “`^x^l`”
 - filename completion: “`^x^f`”
 - “`:help i_^x^l`”, “`:help i_^x^d`”, ...
- Split window
 - “`:sp [filename]`”, “`:vsp [filename]`”, “`^ws`”, “`^wv`”
 - “`:vert [command]`”
- New window
 - “`:new`”, “`:vne`”, “`^wn`”

vim kata (4/8)

- Browse the directory including current file:
 - “:e %:h”
 - Of course, “:Ex” or “:Sex” is better.
 - have ever tried “O after :Sex” or “p after :Ex” ?
- Insert the output of external command:
 - “:r !ls”
 - “^wn” followed by “:r !man printf”
- Hex view
 - “:%!xxd”
- you can “make” or “grep” in vim
 - “:make [options]”, “:grep [options]”
 - try “:cw”, “:cl”, “:cn”, “:cp”, ...

vim kata (5/8)

- open the file under the cursor: “gf”
 - In new window: “^wf”
 - maybe you have to modify “PATH”
 - “:set path+=.,include/,/usr/include”
- Search the definition in header files
 - “[I”, “[]i”,
 - “[][^i”, “[]^wi”, “:is”, “:il”, “:ij”, “:isp”
 - “[D”, “[]d”
 - “[][^d”, “[]^wd”, “:ds”, “:dl”, “:dj”, “:dsp”
 - list all the header file included: “:che”

vim kata (6/8)

- To use cscope in vim

cd /usr/src/linux

make cscope # better than “cscope -b” for Linux kernel

ln -s cscope.in.out cscope.out.in

ln -s cscope.po.out cscope.out.pot

vim

“:cs a .”, “:cs a ..”, “:cs a ...”, ... # can be done in .vimrc

“:cs f {c/d/e/f/g/i/s/t}”

“:scs f {c/d/e/f/g/i/s/t}”

- Pull the object under the cursor to the command-line
 - “^r^w”, “^r^a”
 - “^r^f”, “^r^p”

vim kata (7/8)

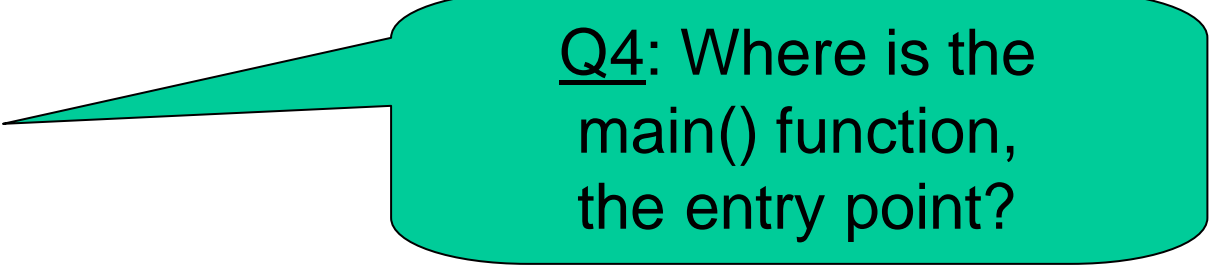
- match parenthesis or #if/#ifdef/#else/#elif/#endif: “%”
- “H”, “M”, “L”, “^e”, “^y”
- “[#”, “]#”
- go to the beginning point of this function: “[[“
- “0”, “^”, “\$”
- “w”, “W”, “b”, “B”, “e”, “E”, “ge”, “gE”
- “t”, “T”, “f”, “F”, “.”, “,”
- “C”, “cw”, “ce”, “A”, “I”, “D”
- “.”
- “v”, “V”, “^v”
- “‘ ‘ ”, “^O”, “^I”

vim kata (8/8)

- “{command} | vim –” at shell prompt
- useful macros
- plugins for programmers
 - Taglist
 - Minibufexplorer
 - gdbvim
- Use Windows? No Problem.
 - <http://vi.kldp.org/jsboard/read.php?table=newqna&no=7>

vim kata

- Frustrated? Use *Source Insight*. It's good!
 - *Visual Slickedit*, *Eclipse+CDT*, *SciTe* and *kscope* are good, too.
 - Of course, *Emacs* is great.
- But isn't it fascinating to edit with *vim*?



Q4: Where is the
main() function,
the entry point?

Boot Process for x86 (1/3)

copied from <http://kelp.or.kr/korweblog/stories.php?story=04/11/18/0830228>

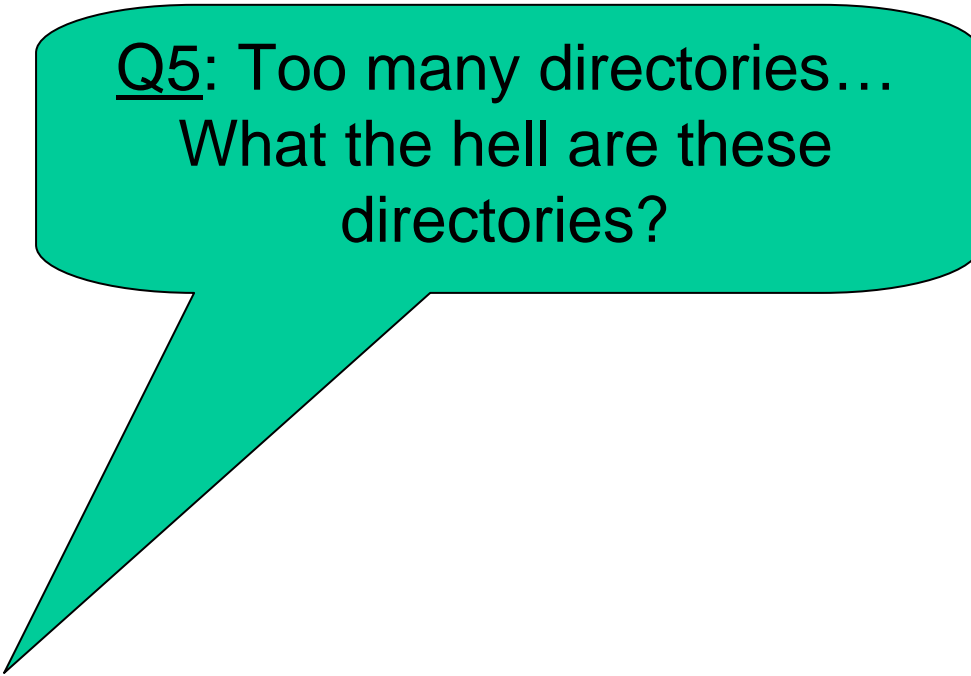
- Power On
 - CS=0xF000, EIP=0x0000FFFF setting
- BIOS
 - CPU real mode
 - POST (Power On Self Test)
 - HW
 - Bootdisk load
 - 0x00090000 jump
- (arch/i386/boot/bootsect.S)
 - 0x00090200(=SETUPSEG) setup code
 - 0x00010000 or 0x00100000
- Setup code (arch/i386/boot/setup.S)
 - RAM (BIOS)
 -
 - idt, gdt
 - protected mode
 - head.S startup_32() jump
- arch/i386/boot/compressed/head.S
 - Segmentation register stack
 - decompress_kernel()
 - 0x00100000 jump

Boot Process for x86 (2/3)

- startup_32() at 0x00100000
 - arch/i386/kernel/head.S
 - Segment register
 - PID=0 (init_task) KMS(Kernel Mode Stack)
 - lss stack_start, %esp
 - ENTRY(stack_start) init_task_union 8kB 가
 - start_kernel() jump
 - call SYMBOL_NAME(start_kernel)
- start_kernel()
 - init/main.c (c , architecture-independent)
 - - trap_init(): exception handler(ISR1)
 - init_IRQ(): hardware interrupt handler(ISR1)
 - rest_init(): init (PID=1)
 - kernel_thread(init, ...);
 - current->need_resched = 1;
 - cpu_idle(NULL);
- init(): (PID=1)
 - do_basic_setup();
 - (bdflush, kpiod, kswapd, ...)
 - , ISR2
 - execve("/sbin/init", ...);

Boot Process for x86 (3/3)

- /sbin/init
 - /etc/inittab
 - | | exec+fork | process | |
|--------------------|----------------|-------------------|-------------------|
| (inetd, getty | |) | |
| • inetd -> telnetd | (id |) -> login (pw |) -> shell -> ... |
| • getty(id |) -> login (pw |) -> shell -> ... | |



Q5: Too many directories...
What the hell are these
directories?

The Structure of the Kernel Source Tree

- init: kernel initialization codes
- kernel: process, program execution, signals, ...
- mm: memory management
- fs: file systems
 - fs/proc: /proc virtual file system
 - devpts: /dev/pts virtual file system
 - ext2, xfs, nfs, ...
- arch: platform-dependent code (process, program execution. interrupt, mm, utility, boot, ...)
 - arch/mips
 - arch/mips/dec
 - arch/mips/sgi
 -
 - arch/i386
 -
- ipc: System V's InterProcess Communication
- net: networking
- block: I/O schedulers for block devices
- drivers: device drivers
 - drivers/block, drivers/char, drivers/net, ...
- lib: general-purpose kernel functions
- include: header files
 - include/asm-mips
 - include/asm-i386
 -



One more Question.

Q6: Is Linux really
portable?

Is Linux Portable?

“Linux is not portable (uses 386 task switching etc.), and it probably never will support any thing other than AT-hard disk, as that’s all I have”

– Linus Torvalds, Aug. 25, 1991.



Is Linux Portable?

But now Linux is really portable
(Arch-dep parts & arch-indep parts are well separated),
and ported to numerous platforms, from a toaster to a
supercomputer.



thank you