

Linux and the Free Software Movement

I202, Fall 2003
Session 9
Thomas Haigh

Invention of UNIX

- Produced in 1970s at Bell Labs
 - Along with C programming language
 - Rapidly spreads in academic world
- Key advantages
 - Starts simple and easily ported
 - Core ("kernel") is small
 - Other components optional, e.g. the shell
 - Tools mixed and matched as needed, shared library grows
 - The "pipe" mechanism allows interconnection of tools
 - Capabilities gradually added
 - Networking
 - Virtual memory



Unix in the 1980s

- Standard OS for
 - Computer science research and teaching
 - Internet use
 - Emerging market for workstation, e.g. SUN
 - Gets X-Windows graphics support
- Never succeeds much on PC
- Hampered by fragmentation
 - Many different versions
 - AT&T tries to make money from selling it



Stallman and GNU

- Free Software Foundation
 - Established by Richard Stallman, "last" MIT hacker
- Coordinates GNU project
 - GNU is Not Unix (recursive name)
 - Intended to produce open, free version of Unix
 - GCC (GNU C Compiler) is biggest success
- Stallman is fanatical, intolerant
 - Ideological rather than pragmatic argument
 - Opposed to closed, proprietary software (even non-free manuals)

GNU GPL

- Extreme commitment to open source
- GNU General Public License (GPL)
- "copyleft" license
 - anything incorporating GPL licensed code must be issued under GPL license (including public source code)
 - Prevents incorporation into commercial products



Linux: Origins

- Personal project of Linus Torvalds
 - Begun in 1991 as undergrad in Finland
 - Inspired by the teaching system MINIX
- Free version of the UNIX kernel
 - Ran on Intel computers
 - Very early version released on net
 - Others pitch in with drivers, patches, etc.
- Torvalds continues to manage process
 - Pick who to trust, how to evolve kernel
 - When to release new version, etc.



Linux: Commercialization

- Under GPL, Linux is free
 - Provides missing kernel need for GNU project
- But, "distributions" are sold
 - Bundle tested and compatible assortments of thousands of different programs & drivers
 - Add installation routines, non-free products & technical support
- Dozens of rival versions
 - Some targeted at different niches
 - Server versions can cost thousands of dollars



Linux: Today

- Estimated #2 in overall server OS market
 - Also gaining share in embedded systems
- Very popular among
 - Computer scientists
 - Hobbyists and organizations with little money
 - People needing web servers or basic file/print stuff
- Gained almost all features of modern UNIX
 - Support for multiple processors
 - Ever wider range of hardware support
- Ported to an enormous number of platforms
 - IBM backing in a big way, using for mainframes
 - Runs commercial software, e.g. Oracle, DB2

Limits of Linux

- Tiny share of desktop/laptop market
 - Limited support for office applications & games
- Still much harder than Windows to administer
 - Installing & software is a pain
 - Hackers love flexibility, but price is confusion
- Now it's cloned UNIX, what next?
 - What are non-ideological benefits?
 - Is copying Windows interface best plan?

The BSDs

- Family of UNIX derived free systems
 - FreeBSD, OpenBSD, etc.
- BSD = Berkeley Standard Distribution
 - Started as a bundle of UNIX tools
 - Eventually became full OS
- Widely used today
 - Different license, so code can be used in commercial products
 - MacOS X is based on one

Open Source vs. Free

- Two kinds of free:
 - Free as in freedom
 - Able to inspect code, modify, create own version
 - Developed collaboratively
 - Free as in beer
 - Doesn't cost money
- Not all open source software necessarily free
 - Until 1980s, operating system code usually public
- Some software cost free but closed

Free Software Projects

- Internet is ideal open source environment
 - Swapping of patches, download of new versions
 - Original browsers and servers were open
 - So was BSD code for TCP/IP
- CVS system coordinates code and patches
 - Allows splitting into "trees"
- Examples:
 - Sendmail
 - Another Berkeley program, developed in 1970s
 - Handles most Internet mail transport
 - Mozilla
 - Reworking of Netscape browser source code

Apache

- World's most widely used webserver
 - July 2003 has 63% of all websites
 - Began as patch to original NCSA code
 - By 1995, completely rewritten
- Less restrictive "Apache license"
 - Commercial versions can be made

Non-Internet Projects

- Open Office
 - Free version of Sun's Star Office suite
 - Word processor, spreadsheet, etc.
- Mame videogame emulator
 - Core team coordinate thousands of drivers
- WINE (Wine Is Not an Emulator)
 - Implementation of Windows API for Linux
 - Commercial off-shoots for games and MSOffice

Current Limits of Open Source

- Great for stuff hackers like to do
 - Web servers, File, print servers, routers, etc.
 - Hacker type programming tools
 - Science & engineering applications
- Weak on stuff you don't for fun
 - Database software
 - Office and "productivity" applications
 - Corporate apps like accounting
- This may be a fundamental limit

Raymond

- Various projects
 - Author of "fetchmail"
 - Maintainer of "Jargon File" (Hackers Dictionary)
- Strong personality
 - Libertarian
 - Science fiction fan
 - Gun enthusiast
 - Free markets guy
- Big time self promoter
 - Widely interviewed
 - Made paper fortune in Linux boom
 - Popularized "open source" over "free software"



Cathedral and Bazaar

- Linux as template for new kind of large scale programming project
- "Cathedral" is traditional approach
 - Elaborate design and planning
 - Centralized control
 - Produces single huge and finished structure
- "Bazaar" is collaborative version
 - Incremental, frequent releases
 - Debugging & improvement by users

Reading "Cathedral"

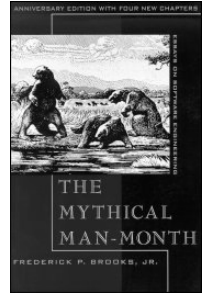
- Don't get too confused by early specifics
 - POP3, SLIP, MTA vs. MDA etc.
- Look for the principles & assumptions
 - Started with a personal problem to solve
 - presence of multiple existing free systems to fetch POP mail onto his PC
 - Ability to adapt existing system
 - Releasing early, collaborating with users

Latter Part More Important

- Raises some big issues
 - Are “all bugs shallow with enough eyeballs”?
 - Can “heroic” hacker debugging compensate for flaws in initial design?
 - Can all great software start from personal need?
 - Does this work for all kinds of software?
 - E.g. accounting package with major bugs will never be used
- Social issues
 - Are programmers motivated by desire for recognition?
 - Does a strong leader need to dominate project?

Context: Fred Brooks

- Classic book “The Mythical Man Month”
 - Described his leadership of IBM OS/360 project in 1960s
- A founding work of software engineering
 - Problems of managing programming different from managing manual work
 - E.g. “Adding more programmers to a late project makes it later”
- Raymond assumes his audience know Brooks
 - & that tying his ideas into this will make people take them seriously



Context: Egoless Programming

- Psychology of Computer Programming by Gerald Weinberg
 - Another 1970s classic
 - Based on experience of doing and teaching programming
- Treats code as something to be read
 - Should have elegant style
 - Popularizes “walk through” & “pair programming”
 - Programmers must be “ego less”, accept constructive criticism.
- Major influence on modern “extreme programming”
- Raymond says that his model builds on Weinberg’s insights

