Open Source OS's – CS 450

Peter Chinetti
School of Electrical and
Computer Engineering
Illinois Institute of Technology
Chicago, Illinois 60616
Email: peter@chinetti.me
A20265405

***Abstract*—An overview of three OSS OS's (Open Source Software Operating Systems): Linux, Minux and FreeBSD.**

I. Linux

A. Introduction

According to the kernel.org site: "Linux is a clone of the operating system Unix, written from scratch by Linus Torvalds with assistance from a loosely-knit team of hackers across the Net" [1]. Linux began life in 1991 as a college project by Linux Torvalds: a Finnish student. In the broader computer world at the time the GNU project was a kernel away from having a UNIX clone. Work on that kernel has continued, however a stable release is still forthcoming. Red Hat Inc., a company selling Linux support, has grown into a $8.9 billion dollar company and was added to the S&P 500 in 2009 [2]. As an aside, this paper was entirely written on an [Arch] Linux machine using mostly open source software. According to Bagozzi and Dholakia, "Since its first lines of code were written by Linus Torvalds ..., Linux has become the best known, most successful, and widely adopted OSS to date. [6]"

B. History

To discuss the history of Linux, we must first discuss the history of the Free Software. All of the operating systems discussed here are somewhat based off the original UNIX, written by Dennis Richie and Ken Thompson while they worked for Bell Labs circa 1969. It was (and is) a well designed system of abstraction away from the base hardware using a relatively straightforward interface. It was widely used, however it remained the intellectual property of Bell. This made it very expensive, and protected by licenses that prohibited modification by the end users. Fast forward to 1984, when Richard Stallman (rms) was dissatisfied with the encroachment of closed source, prohibitive software and decided to personally take action. He resigned his job at MIT (although he was allowed to keep his office) and started on what would become the Free Software Foundation and the GNU (a "recursive" acronym which stands for "GNU's Not Unix") Project. In 1991, the year of the birth of Linux, many of the tools required for a UNIX clone (a compiler, a shell etc.) had been written by the FSF. The kernel (named HURD, as in a herd of gnus), however, was (and still is) nowhere near completion. Linus, therefore, took it upon himself to write a free kernel. In some of his annotated emails from the beginning of the project he writes: "I'm doing a (free) operating system (just a hobby, won't be big and professional like gnu) [3]." Little did he know that people would jump on his kernel and begin using it for their own. By "two years after Linus' post, there were 12000 Linux users. [4]"

C. Size

The Linux kernel now has 1229 commits a week (175.6 a day) from 326 authors. The development work on it is huge. It is easily one of the largest open source projects.

D. Maturity

Linux is a mature project, it has support for a number of architectures and has performance comparable to other major operating systems.

II. Minux

A. Introduction

Minux is the system that directly inspired Linux. Its author, Andrew Tanenbaum, was and is a professor of Computer Science (specializing in operating system design) at the VU University Amsterdam. In an email seemingly designed to spark a flame war, he laid out his view that "Linux is Obsolete." In it, he argued that he "could go into a long story here about the relative merits of the two designs, suffice it to say that among the people who actually design operating systems, the debate is essentially over. Microkernels have won."

B. History

Professor Tanenbaum wrote Minux as a proof of concept, then hamstrung it with restrictive licensing. In 1987, he released a book called "Operating System Concepts," which was lauded as being different because it was able to "not only does it discuss all the principles in detail, but it also presents and discusses the complete source code of a small UNIX-like operating system called MINIX. By studying both the principles and the source code of a real system, the reader gets a far deeper understanding of how operating systems really work [5]." Originally, the source code was distributed with a license that prohibited modification and redistribution, so people looking to improve the operating system had to

distribute patches rather than source. This is a labor-intensive process, and hampered the spread of the system. Additionally, MINUX had a cost. It cost something like $70 for a copy of the source, which was significantly cheaper than other proprietary options, but was still a limitation for students.

C. Size

MINUX is a small project, used in very few "production" applications. It is tiny compared to almost every other kernel, although it may have a larger install base than the GNU project's microkernel.

D. Maturity

Although MINUX has been under development for more years than Linux, it has had fewer developers, and as a result fewer "man-hours" invested into the project.

III. FreeBSD

A. Introduction

FreeBSD came out of the Berkeley Security Distribution (BSD) project. BSD was developed by the Computer Systems Research Group of the University of California, Berkeley starting in 1977. BSD is the closest relative of UNIX, with the initial releases being based directly on it. Eventually, the UNIX copyright became a problem as the new owner asserted their copyright. This slowed the uptake of (what eventually became) FreeBSD.

B. History

FreeBSD is ancient. It began, (as stated above) in *1977*, *37 years* ago. The project called "FreeBSD" started in 1993, as patches on 385BSD. Eventually, it forked and released code from 4.3BSD-Lite, 368BSD and FSF code in September of 1993. The legal wrangling around BSD took its toll on FreeBSD as well: forcing the developers to stop and re-implement code to allow the legal issues to cease. A clean version was published in January 1995. By 1996, FreeBSD was replacing systems in big names like Hotmail and Yahoo!. Eventually support for modern features like SMP and ELF binaries was included.

C. Size

FreeBSD is the largest of the BSD family, with 77% of the user base. However, the BSD project is still significantly smaller than the Linux community. FreeBSD, however, has a more business-friendly license than Linux and is used in some proprietary devices and appliances. The single most obvious contribution of FreeBSD to the commercial world is Mac OS X, which is based on Darwin, which is based on FreeBSD.

D. Maturity

FreeBSD has an amazing history and is certainly mature. FreeBSD's developer base is smaller than Linux, but they have managed to create a powerful, secure system.

IV. Conclusion

Linux is the largest free Operating System in the world today, for technical and historical reasons. The author uses it for his daily computing, and even had a job in a "Linux Engineering" department last summer. The other OSS OSes form a vital part of the community, with code moving across projects (as is the case with FreeBSD), or serving asan inspiration for others (as with MINUX). OSS OSes are a huge part of the world today.

Note

This paper was written using the IEEE LaTeX style found here: IEEE - Manuscript Templates for Conference Proceedings

References

[1] *About Linux Kernel*. kernel.org, 2013. Web.

[2] Tom McCallum, VP Investor Relations *Quarterly Fact Sheet*. Red Hat Inc., North Carolina, 2013. Web.

[3] Linus Torvalds *LINUX's History by Linus Torvalds*. Carnegie Mellon University, Pittsburgh PA,

[4] *Introduction to Linux – History*. tldp.org, 2008. Web.

[5] *Operating Systems - Design and Implementation*. http://minix1.woodhull.com/osdi2/ 2008. Web.

[6] Richard P. Bagozzi and Utpal M. Dholakia. *Open Source Software User Communities: A Study of Participation in Linux User Groups*. Management Science. Vol. 52, No. 7, Open Source Software (Jul., 2006). pp. 1099-1115 Published by INFORMS Stable URL: http://www.jstor.org/stable/20110583 2008. Web.

Problems

A. 1.15

Symmetric multiprocessing differs from asymmetric multiprocessing in that in SMP, there is no "controller" processor, all processors are equals. This allows for resources to be more efficiently utilized. A (the) difficulty with multiprocessor systems is synchronization of memory. However, multiprocessing systems can be more responsive to many tasks, can be faster if tasks can be paralleled, and it is generally cheaper to put more cores/processors on a chip than it is to make a core/chip go faster.

B. 1.16

Clusters are connected by networks which are about 3 orders of magnitude slower than the internal interconnects of a computer. This means that total memory synchronization cannot efficiently happen, and as such the cluster must find cheaper, faster, less complete methods of synchronization to provide a service.

C. 1.19

Interrupts exist to serve asynchronous needs. If something changes in the environment of the computer (either a device or even a sensor to the external world), it needs to be handled promptly. Traps are mainly mean to branch into a section of privileged code from user land code.

D. 1.20

1) With DMA, the CPU initializes the transfer.
2) The device interrupts the CPU only when the I/O is complete.
3) The I/O process does not interfere directly unless the memory spaces collide. Bus contention is a problem, however.

E. 1.24

1) There is no real cache coherence problem in single processor systems, just marking cache lines as dirty and deciding when to write back to memory.
2) With Multiprocessors Systems, cache coherence becomes a big problem. There is hardware designed to ensure cache coherence, however, it is slow.
3) Distributed systems make it unreasonable to synchronize processor caches, however synchronization of higher level caches is possible, done in software, and *slow*.

F. 2.17

Yes, that is why there are so many shells (t, c, bash, zsh, etc.).

G. 2.18

Shared memory (no synchronization, full speed, maximum possibility for conflict) and message passing (synchronized, slower, less possibility for conflict).

H. 2.21

Microkernels can be more reliable. All interaction is done through message passing, which allows components to be removed, replaced and restarted without affecting one another. That message passing slows the system.

I. 2.25

The Synthesis method is awesome. However, much like other awesome things, such as fighter jets, the engineering work needed to keep the system running and debugged may easily eat the performance benefits of assembling in kernel.