AFTER THE SOFTWARE WARS

Home page

Draft 1.029
Words: 6925, Pages: 34

KEITH CURTIS

Free Software Battle

Some people think much faster computers are required for Artificial Intelligence, as well as new ideas. My own opinion is that the computers of 30 years ago were fast enough if only we knew how to program them.

—John McCarthy, computer scientist, 2004

This IBM 305 RAMAC Computer, introduced in 1956, was the first computer containing a (5 MB) hard drive on 24 huge spinning platters. Today you can get 1000 times more memory in something the size of your thumb.

Given the technology that's already available, we should have cars that drive us around, in absolute safety, while we lounge in the back and sip champagne. All we need is a video camera on the roof, plugged into a PC, right? We have all the necessary *hardware*, and have had it for years, but don't yet have robot-driven cars because we don't have the *software*. This book explains how we can build better software and all get our own high-tech chauffeur.

The key to faster technological progress is the more widespread use of free software. Free versus proprietary (or non-free) software is similar to the divide between science and alchemy. Before science, there was alchemy, where people guarded their ideas because they wanted to corner the market on the means to convert lead into gold. The downside of this “strategy” is that everyone would have to learn for themselves that drinking mercury is a bad idea.[1] The end of the Dark Ages arrived when man started to share advancements in math and science for others to use and improve upon. In fact, one way to look at history is to divide it between periods of progress and stagnation.

...

This book presents a vision of the future, but I believe we could have had these advancements decades ago. While the details of this futuristic world are unclear, what is clear is how to make that world happen faster. Free software's paradoxical success should also cause us to question other assumptions about copyright, patents, and other topics that will also be discussed.

iBio

I first met Bill Gates at the age of twenty. He stood in the yard of his Washington lake-front home, Diet Coke in hand, a tastefully small ketchup stain on his shirt, which no one had the courage to point out, and answered our questions, in-turn, like a savant. As a college summer intern, I had planned for a potential encounter and I approached him with questions that interested me but which would be arcane to non-computer mortals.[2]

His answers demonstrated that he was one of the top software experts on the planet and convinced me that I would be wise to start off my career at Microsoft.

---

1 The digital version of this book has a number of hyperlinked words that take you to references, like this video of writer Cory Doctorow at a Red Hat Summit.

2 I asked him about the performance of Microsoft Exchange's database storage engine as compared to the one inside Microsoft SQL Server, and about NetWare's newly-announced clustering technology called SST Level 3.

...

Like many of my fellow employees, I was only vaguely familiar with free software when I left and randomly decided to check out this thing called Linux. At Microsoft, I got all the software I wanted for free, and I always thought free software would be behind proprietary software. For 15 years I had made it a priority to learn about many aspects of Microsoft technologies, and my office contained rows of books on everything from *Undocumented Windows* to *Inside SQL Server*. When running Windows I felt as comfortable as Neo in the Matrix, without the bullets and leather, so while I was willing to look around, I was half-forcing myself and didn't want this little experiment to mess up my main computing environment.

Every technical decision was big for me: which version of Linux should I try? Should I get an extra machine or can I try dual-boot? Can I really trust it to live on the same hard drive as Windows? I got some tips and assurance from a Microsoft employee who had recently tried Linux, and with that, and the help of Google, I proceeded with the installation of Red Hat's Fedora Core 3.

While I came to not be all that thrilled with Fedora itself, I was floored merely by the installation process. It contained a graphical installer that ran all the way to completion, it resized my NTFS partition — which I considered a minor miracle, setup dual boot, and actually did boot, and let me surf the Web. I didn’t have a clue what to do next, but the mere fact that this all *worked* told me more about the potential of Linux than anything I had read so far. You cannot, by accident, build an airplane that actually flies.

Over time, what impressed me the most about Linux was the power of it all. It came with tons of applications: Firefox, OpenOffice, GIMP, Audacity, Mono, MySQL, and many more for me to discover. The UI was simple, responsive, polished and customizable. Installing the Apache web server took just a few seconds and gave me access to a vast world of PHP. Installing the WordPress blog took me 15 minutes the first time, but I knew when I became more proficient at things, I could do it in one. I came to understand that beyond its poorly debugged device drivers, a Windows computer is a sad joke. By mid-2005, I was in love with computers again!

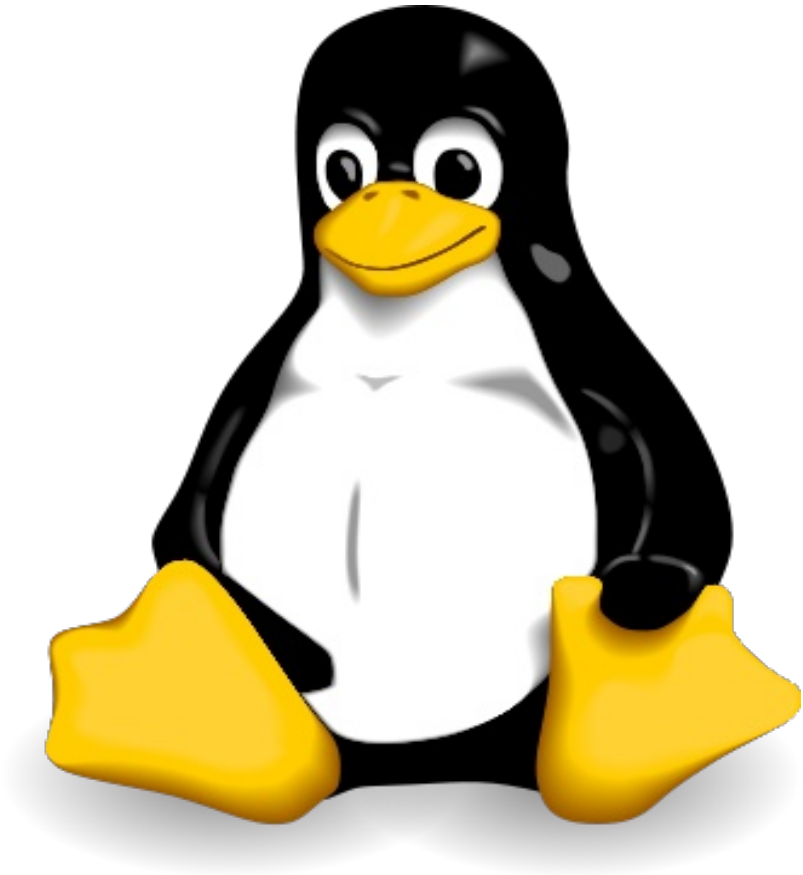
I've spent three years in diligent research on the key subjects of this book, talking to hundreds of programmers, attending many conferences, and reading source code, magazines, websites and books. This book isn't really about the death of Microsoft as much as it is about the Microsoft proprietary development model that has per-

vaded or even infected computing. I have absolutely zero bitterness towards Microsoft although I now believe they are toast. I loved working there, learned an enormous amount, and enjoyed the privilege of working alongside many brilliant minds. Like many things in life, it was fun while it lasted.

Linux

Really, I'm not out to destroy Microsoft. That will just be a completely unintentional side effect.

—Linus Torvalds, 2003



The Linux mascot, Tux, created by Larry Ewing

The kernel of an operating system (OS) is the central nervous system of a computer. It is the first piece of software that the computer executes, and it manages and mediates access to the hardware. Every piece of hardware needs a corresponding kernel device driver, and you need *all* of your drivers working before you can run *any* of your software. The kernel is the center of gravity of a software community, and the battle between free software and Windows is at its lowest level a battle between the Linux and Win-

dows kernels. Microsoft has said that it has bet the company on Windows, and this is not an understatement! If the Windows kernel loses to Linux, then Windows, and Microsoft, is also lost.[1]

The Linux kernel is not popular on desktops yet, but it is widely used on servers and embedded devices because it supports thousands of devices and is reliable, clean, and fast. Those qualities are even more impressive when you consider its size: printing out the Linux kernel's 8,000,000 lines of code would create a stack of paper 30 feet tall! The Linux kernel represents 4,000 man-years of engineering and 80 different companies, and 3,000 programmers have contributed to Linux over just the last couple of years.

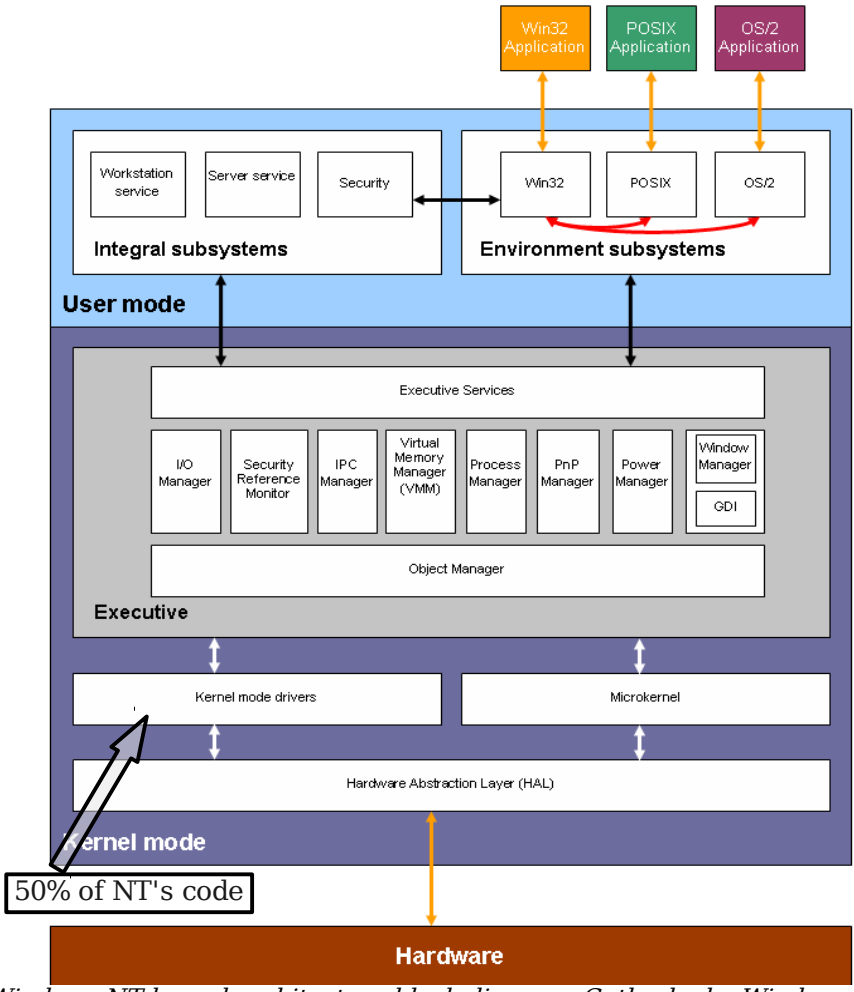
That 30-foot stack of code is just the basic kernel. If you include a media player, web browser, word processor, etc., the amount of free software on a computer running Linux might be 10 times the kernel, requiring 40,000 man-years and a printout as tall as a 30-story building.

This 40 man-millennia even ignores the work of users reporting bugs, writing documentation, creating artwork, translating strings, and performing other non-coding tasks. The resulting Linux-based free software stack is an effort that is comparable in complexity to the Space Shuttle. We can argue about whether there are any motivations to write free software, but we can't argue it already exists!

One of the primary reasons I joined Microsoft was I believed their Windows NT (New Technology) kernel, which is still alive in Windows Vista today, was going to dominate the brains of computers, and eventually even robots. One of Bill Gates' greatest coups was recognizing that the original Microsoft DOS kernel, the source of most of its profits, and which became the Windows 9x kernel, was not a noteworthy engineering effort. In 1988, Gates recruited David Cutler from Digital Equipment Corporation, a veteran of ten operating systems, to design the product and lead the team to build the Windows NT kernel, that was released as I joined in 1993.

1 While cloud computing, the movement of increasing number of applications and services provided over the Internet, is one of the hot topics of today, it is unrelated to the Windows vs. Macintosh vs. Linux war that is going on. Even in a future where applications like word-processing are done over the Internet, you still need a kernel, a web browser, a media player, and so forth.

The kernel Cutler and his team developed looks like this:



Windows NT kernel architecture block diagram. Cutler had a Windows 95 doormat outside his office; you were encouraged to wipe your feet thoroughly before entering.

Unfortunately for Microsoft, the original kernel lived on through Windows 95, Windows 98, and into Windows Me. (Microsoft also had Windows CE, a small kernel for embedded devices. Microsoft had three separate kernels for most of my tenure, whereas the same Linux kernel is used on small and big devices.)

Windows has become somewhat popular for servers and devices, but it never achieved the dominance it did on desktop PCs. Perhaps the biggest reason is that its code wasn't available for others to extend and improve upon. The Linux kernel took off because there

are people all over the world, from Sony to Cray, who tweaked it to get it to run on their hardware. If Windows NT had been free from the beginning, there would have been no reason to create Linux. However, now that there is the free and powerful Linux kernel, there is no longer any reason but inertia to use a proprietary kernel.

There are a number of reasons for the superiority of the Linux kernel. But first, I want to describe the software development process. When you understand how the Linux kernel is built, its technical achievements are both more impressive and completely logical.

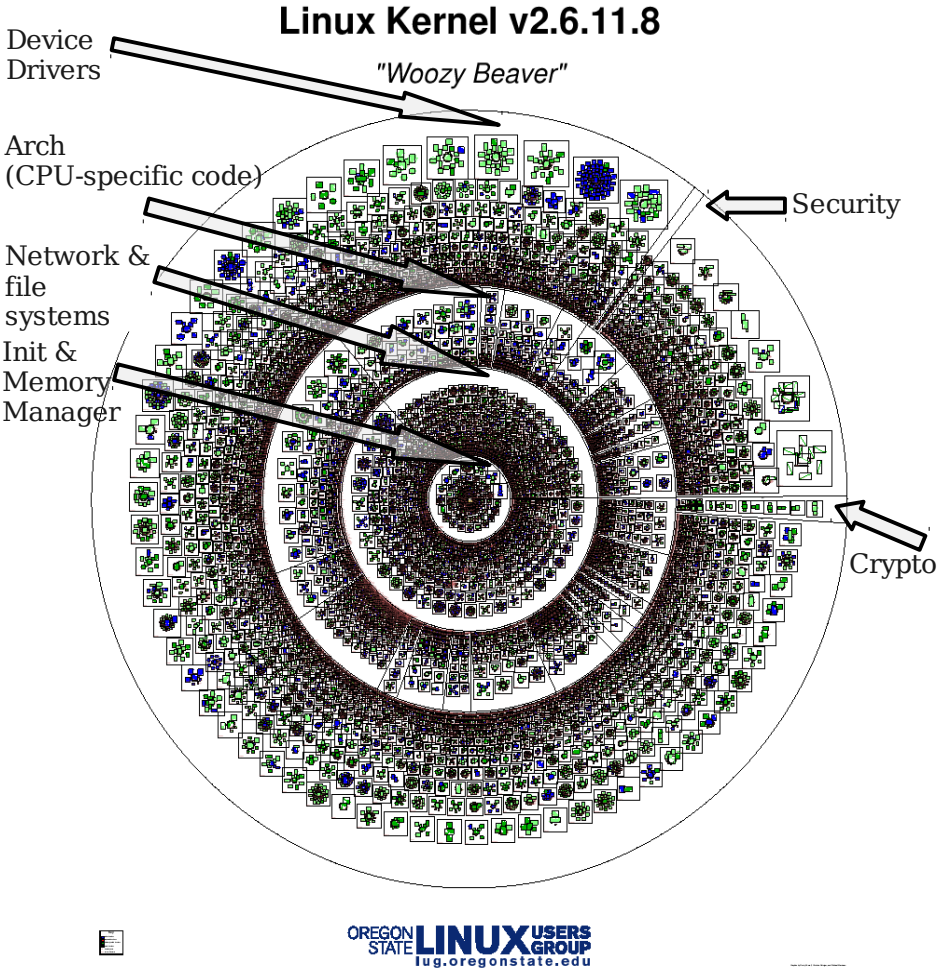
...

Linux Kernel Superiority

Here are the reasons Linux is superior to the Windows kernel:

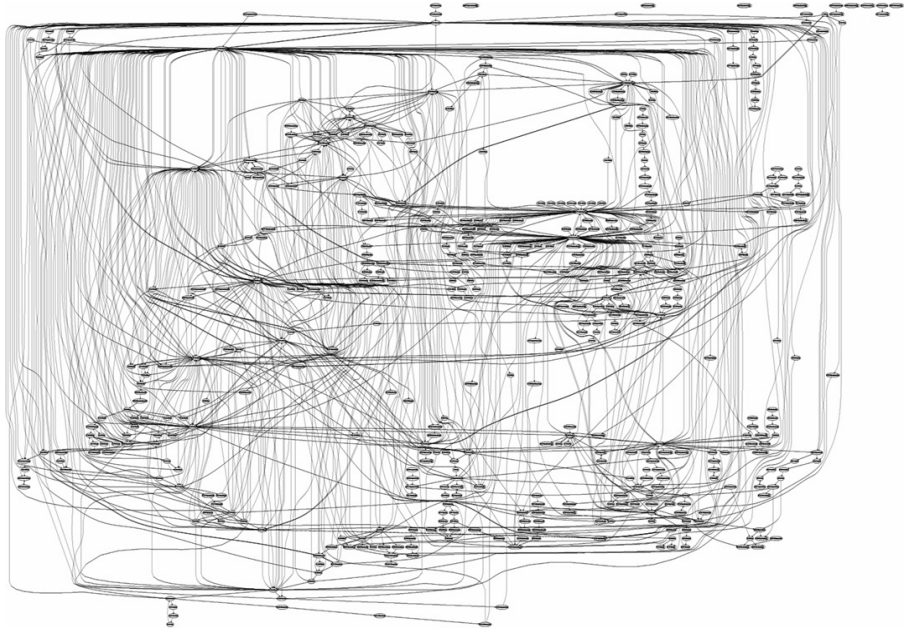
1. Refactored Code (Reliability)

Here is a diagram of the Linux kernel:

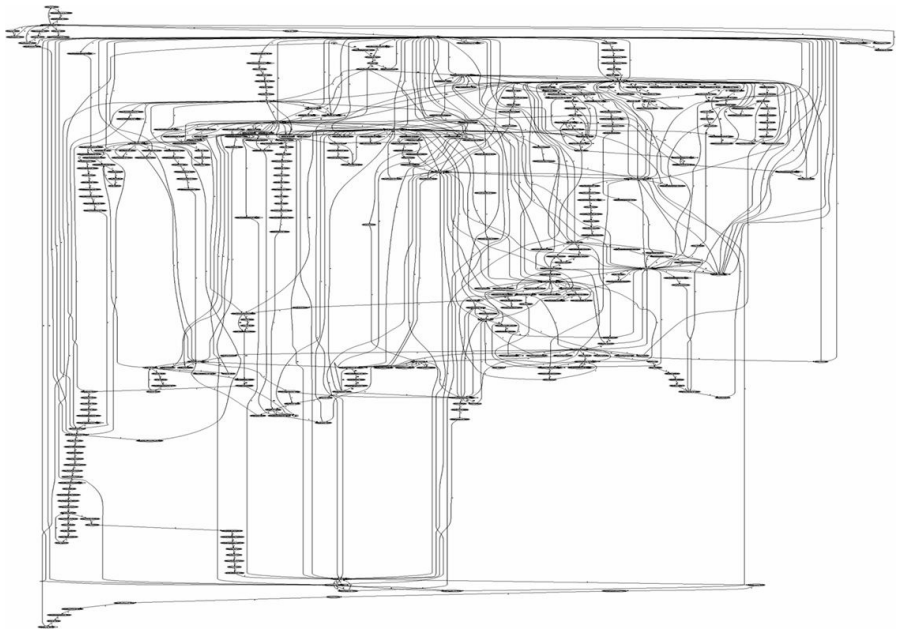


...

Here is a graph of all the function calls into the OS required to return a simple web request. These pictures demonstrate a visual difference in complexity between free and proprietary software:



System call graph in Microsoft's proprietary web server, IIS.



System call graph to return a picture in the free web server Apache.

AI and Google

> The future is open source everything.
>
> —Linus Torvalds

> That knowledge has become *the* resource, rather than *a* resource, is what makes our society post-capitalist.
>
> —Peter Drucker, 1993

Imagine 1,000 people, broken up into groups of five, working on two hundred separate encyclopedias, versus that same number of people working on one encyclopedia? Which one will be the best? This sounds like a silly analogy when described in the context of an encyclopedia, but it is exactly what is going on in artificial intelligence (AI) research today.[1] Some say free software doesn't work in theory, but it does work in practice. In truth, it “works” in proportion to the number of people who are working together, and their collective efficiency.

In early drafts of this book, I had positioned this chapter after the one explaining economic and legal issues around free software. However, I now believe it is important to discuss artificial intelligence separately and first, because AI is the holy-grail of computing, and the reason we haven't solved AI is that there are no free software codebases that have gained critical mass. Far more than enough people are out there, but they are usually working in teams of one or two people, or proprietary codebases.

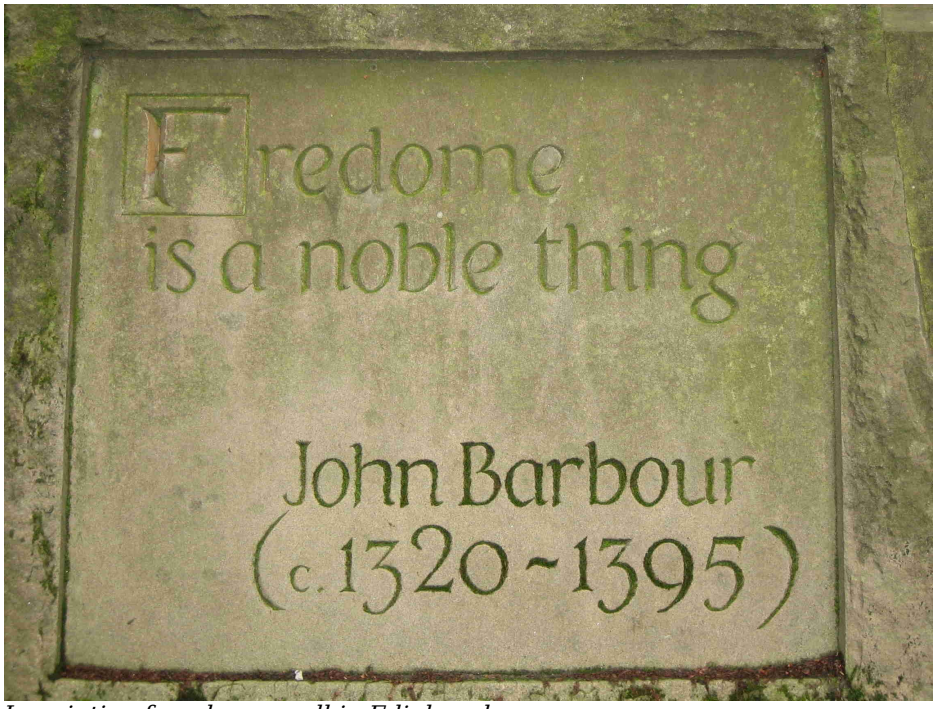
...

1 One website documents 60 pieces of source code that perform Fourier transformations, which is an important software building block. The situation is the same for neural networks, computer vision, and many other advanced technologies.

Free Software

If you have an apple and I have an apple and we exchange these apples then you and I will still each have one apple. But if you have an idea and I have an idea and we exchange these ideas, then each of us will have two ideas.

—George Bernard Shaw



Inscription found on a wall in Edinburgh.

Much of man's existence until the late 20th century involved an economy focused primarily on the manipulation of scarce, unmalleable atoms. Copyright law was created to protect writers from the publishers. In a digital world, we can all be creators and publishers, so we need to revisit many fundamental questions, from the means we use to protect ideas, to the ways in which we purchase them.

This and the next chapter discusses details of free software, copyright and patent laws, but let's start by remembering that a shift towards a presumption of free digital information is actually a moral question. The Internet makes transmitting knowledge essentially free, so as free software evangelist Eben Moglen asks: “If you could feed the world for free, would you? Likewise, if you could provide

every child access to a library of human knowledge they would never outgrow, would you?” It is the Internet that makes this question possible to ask, and necessary to answer.

...

The OS Battle

> Free software works well in a complex environment. Maybe nobody at all understands the big picture, but evolution doesn't require global understanding, it just requires small local improvements and an open market ("survival of the fittest").
>
> —Linus Torvalds

> I've been a big proponent of Microsoft Windows Vista over the past few months, even going so far as loading it onto most of my computers and spending hours tweaking and optimizing it. So why, nine months after launch, am I so frustrated? The litany of what doesn't work and what still frustrates me stretches on endlessly.
>
> Take sleep mode, for example. Vista promised a new low-power sleep mode that would save energy yet enable nearly instantaneous resume. Poppycock. The brand-new dual-core system I built a few months ago totters off to sleep but never returns. I have to cold-start it to bring it back. This after replacing virtually every driver inside.
>
> Take my media center PC, for example. It's supposed to serve up photos, videos, and music. Instead, it often simply drops off the network for absolutely no reason.
>
> I could go on and on about the lack of drivers, the bizarre wake-up rituals, the strange and nonreproducible system quirks, and more. But I won't bore you with the details.
>
> —Jim Louderback, Editor in Chief of PC Magazine.

Windows revenue of $16 billion is larger than the GDP of many countries, and it is this dominance that provides technical and financial competitive advantages that fuel Microsoft's success everywhere else. The battle for the desktop computing OS is an epic struggle seldom seen in the history of business, and the success of Linux on the desktop will reinvigorate the PC as a tool much more powerful than a word processor and a web browser.

To a Linux distribution, the kernel is just the software that makes all the other software run, and is one of the thousands of components they integrate. The Linux kernel by itself will not defeat Windows — it requires an entire distribution. Therefore, it is worth analyzing the state of the OS market.

There are many producers of Linux distributions, but in the PC world the four most important teams are: Red Hat, Novell, Debian, and an upstart created in 2004, Ubuntu. There are hundreds of other Linux distributors, but most of them are merely using the big

four's efforts and tweaking them further for more specialized markets. Each of these four will be discussed over the next few pages, but it is worth mentioning a name that isn't on the list, IBM.

...

Apple

> After Woz hooked his haywire rig up to the living-room TV, he turned it on, and there on the screen I saw a crude Breakout game in full color! Now I was really amazed. This was much better than the crude color graphics from the Cromemco Dazzler. ... “How do you like that?” said Jobs, smiling. “We're going to dump the Apple I and only work on the Apple II.” “Steve,” I said, “if you do that you will never sell another computer. You promised BASIC for the Apple I, and most dealers haven't sold the boards they bought from you. If you come out with an improved Model II they will be stuck. Put it on the back burner until you deliver on your promises.”
>
> —Stan Velt, former Editor-in-chief, *Computer Shopper*

Apple's iPod and iPhone may be sexy and profitable, but these small devices are specialized in function, so there isn't a lot to say about them. An iPod is busy when playing music, whereas when your computer plays music, it uses less than 1% of its computing power, which is not even noticeable.

For most of Apple's existence, they never really got the idea of the relationship between market share and a developer community. For example, Macs have historically not been allowed in enterprises because no one added the necessary features and applications — because it never got the requisite market share to make anyone want to bother.

Microsoft understood the virtuous cycle between users and developers, and knew that making it easy to build applications would make Microsoft's ecosystems successful. Bill Gates brags that Microsoft has ten times as many partners as Apple, and tools like Visual Basic and FrontPage were important reasons why.

This internal focus that has limited the Mac's potential marketshare is now playing itself out with their new devices. Symptoms of this mindset are noticeable in the most basic scenarios: you cannot drag and drop music on and off an iPod, as you can with a digital camera. Even if you could copy over your files, unless it is in one of the few formats Apple can be bothered to support, you would still not be able to play it.

...

Software

> Nobody has ever had more contempt for customers than Steve Jobs.
>
> —Eben Moglen

In a better iWorld, Apple would have used Linux instead of BSD.

With less of an internal focus, Apple could even have done a few things to make their OS more compatible with Windows, to increase sales and to better tempt Windows users into switching. There is plenty of free software out there to enable interoperation with Windows technologies. Out of the box, Windows Media is treated by the Mac like a text file; there is code out there to fix this, but Apple doesn't provide it. It's as if supporting Windows Media is a concession that weakens Apple. We aren't even talking about having music-creating software support that format — simply the ability to play these files!

Their iChat program doesn't support MSN messenger, though it does support AOL and Yahoo. Given the unnecessary hurdles Apple has created for Windows users, it is not surprising that their market share remains so low. There is even a free Win32 implementation known as WINE that would allow Windows software to run on the Mac; yet another way to storm Microsoft's beachfront that Apple hasn't adopted.

...

Apple has been good for Linux because it has forced software vendors to think about writing cross-platform code. By using free software like the BSD Kernel, Apple is more of a part of the free software community than before, although not by much. Apple's web browser Safari is based on the engine WebKit, which Apple derived from the free KHTML, but they worked for a year on the fork before attempting to integrate their changes back, and KHTML developers said that their relationship with Apple was a “bitter failure.”

...

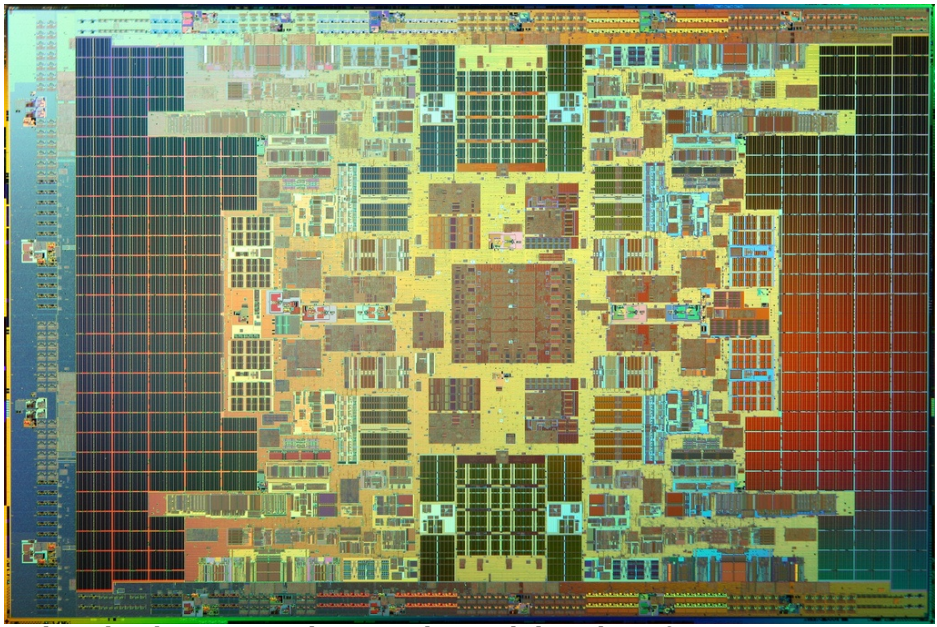
Tools

You can tell a craftsman by his tools.

—Socrates

The major cause of the software crisis is that the machines have become several orders of magnitude more powerful! To put it quite bluntly: as long as there were no machines, programming was no problem at all; when we had a few weak computers, programming became a mild problem, and now we have gigantic computers, programming has become an equally gigantic problem.

—Edsger Dijkstra, 1972



Today's hardware is much more advanced than the software.

While cooperation amongst our computer scientists is important, the tools we use are as well, as they can either facilitate progress, or impede it. The purpose of this chapter is to explore the biggest technical reason why people distrust computer technology today. Many of the problems that frustrate users, such as crashes, security violations, code bloat, and the slow pace of progress, are directly related to the software used to create our software.

In fact, the last hardware bug anyone remembers was the Intel floating point division bug in 1994 — nearly all the rest are software bugs.[1]

The problem is this: the vast majority of today's code is written in C, a programming language created in the early 1970s, or C++, created in the late 1970s. Computers always execute machine language, but programming these 1s and 0s are inconvenient in the extreme, so programmers create high-level languages and compilers that convert meaningful statements to machine code. We are asking the same tools used to program the distant computer ancestors to also program our iPods, laptops, and supercomputers, none of which were even conceived of back then.

Imagine building a modern car with the tools of Henry Ford.

...

1 In fairness to Intel, that bug happened on average once in every nine million division operations, and would return results that were off by less than 0.000061. Testing a 64-bit processor's math capabilities involves 2^{128}, or 10^{38} test cases! Intel does continuously release errata lists about their processors. For example, initializing a computer is a very complicated process with plenty of room for ambiguity, yet because user data isn't even in memory yet, there is no risk of a hardware bug causing a newsworthy problem.

The Java Mess

> I think everybody hates Java as a desktop thing. I see Java mentioned a lot lately, but all of the mentions within the last year have been of Java as a server language, not as a desktop language. If you go back a year and a half, everybody was talking about Java on the desktop. They aren't anymore. It's dead. And once you're dead on the desktop, my personal opinion is you're dead. If servers are everything you have, just forget it. Why do you think Sun, HP — everybody — is nervous about Microsoft? It's not because they make great servers. It's because they control the desktop. Once you control the desktop, you control the servers.
>
> It's no longer something that will revolutionize the industry. It could have revolutionized the industry if it was on the desktop, but I don't see that happening anymore. I hope I'm wrong. Really. I just don't think I am.
>
> —Linus Torvalds, 1998

> A company should build a process that systematically looks at every product, every service, every process, every policy, every market with the question, “If we weren't doing this already, knowing what we now know, would we start it, would we go into it?” If not, how quickly can we get out?
>
> —Peter Drucker

```
class JavaProgram
{
    public static void main(String args[])
    {
        int counter = 1; //Add up all the integers from 1 thru 10
        int sum = 0;     //Store value in variable sum

        while (counter <= 10)
        {
            sum = sum + counter;
            counter = counter + 1;
        }
        System.out.println("Numbers 1-10 add to: " + sum);
    }
}
```

Java is relatively elegant, and should have replaced C and C++. The highlighted portions show the very few places that would need changing to port to C#.

In 1995, Sun Microsystems created a next-generation programming language called Java, something that could have been the most significant part of their legacy, more important than their Sparc processor, Solaris (their flavor of Unix), or anything else. Java

had the potential to change the software industry by becoming the next generation C/C++, but now I think it is a dying language. (Java is similar in name to Javascript, the programming language of web browsers, and both are similar to C, but the languages and runtimes are incompatible and different.)

At first glance, Java looks quite like C, and very much like Microsoft's C#. Sun's Java adopted much of the syntax and semantics of C but added object orientation and many other language innovations, including garbage collection, and they made no effort to be 100% backwards compatible like C++ did.

While Java has many important technical advancements, it has achieved only a small fraction of the universal status it should have. Java should have replaced C and C++, languages desperate to be taken out back and shot! However, Java did not, and one of the biggest reasons why software is in shambles today is because Sun repeatedly screwed the pooch with Java.

...

Challenges for Free Software

> “Bill doesn't really want to review your spec,” a colleague told me. “He just wants to make sure you've got it under control. His standard MO is to ask harder and harder questions until you admit that you don't know, and then he can yell at you for being unprepared. Nobody was really sure what happens if you answer the hardest question he can come up with, because it's never happened before.”
>
> Watching nonprogrammers trying to run software companies is like watching someone who doesn't know how to surf trying to surf. Even if he has great advisers standing on the shore telling him what to do, he still falls off the board again and again. The cult of the MBA likes to believe that you can run organizations that do things that you don't understand. But often, you can't.
>
> —Joel Spolsky

> The mode by which the inevitable is reached is effort.
>
> —Felix Frankfurter, US Supreme Court Justice

Free software has been around since 1985, and yet has only 1% marketshare on the desktop today. Free software has tremendous potential, but the community needs to execute better to win. In fact, until free software succeeds on the desktop, its last and biggest challenge, many will continue to question whether it is even viable.

...

PC Hardware

> We love Linux, and we're doing our best to support the Linux community. we see the Linux desktop as a customer-driven activity. If customers want it, well, Dell will give it to them.
>
> —Michael Dell

Michael Dell's quote demonstrates that he doesn't consider the situation where people aren't running Linux on his hardware because it doesn't work! I have a Dell Vostro laptop that doesn't have a Linux driver for wireless Internet. If I can't take the computer to a coffee shop and surf the web, it is useless. I've thought about putting a bullet through the laptop and mailing it back. Even laptops by Dell that ship with Linux still contain proprietary drivers, drivers that aren't in the kernel, and so forth so it seems clear that

Dell management doesn't understand Linux yet. Dell also inexplicably has 30 models of laptops, each with 30 options, so it might just be a general case of corporate cranial-rectal inversion.

Overall, PC hardware support in Linux is generally in good shape, and has improved a lot in the years since I first started using it. I believe it mostly requires that we continue to press on, working through the chicken and egg issue where hardware vendors are reluctant to support Linux until it has more users, but users won't run Linux if it doesn't fully support their hardware.

I singled out IBM in the OS chapter, but many other hardware companies are under-investing in free software even though public statements by their VIPs suggest that they believe in it. For example, Intel claims to be a strong support of Linux, but is doing only a decent job in its support of Linux drivers.[1] An Intel engineer told me at a Linux conference that their Linux efforts are just 1% of the manpower that their Windows efforts receive. Doubling their Linux development team would cost less than .1% of their total R&D. Intel is under-investing in Linux not because they can't afford to increase costs by .1%, but because they're suffering from Stockholm syndrome![2]

...

1 My Intel wireless card resets and frequently re-associates or doesn't always resume properly. (These problems appear to now be fixed in 2009, but it took years.) In general, Intel's video drivers are considered slow, buggy, and behind: Intel added Linux driver support for TV-out years after the hardware was released! There are recent news reports that Intel's GMA 500 drivers are “a bloody mess.”

2 Wikipedia: “Stockholm syndrome is a psychological response sometimes seen in an abducted hostage, in which the hostage shows signs of loyalty to the hostage-taker, regardless of the danger in which they have been placed. Loyalty to a more powerful abuser — in spite of the danger that this loyalty puts the victim in — is common among victims of domestic abuse, battered partners and child abuse. In many instances the victims choose to remain loyal to their abuser, and choose not to leave him or her, even when they are offered a safe placement in foster homes.”

The Desktop

> The *only strategy* in getting people to switch to your product is to *eliminate barriers*. Imagine that it's 1991. The dominant spreadsheet, with 100% market share, is Lotus 123. You're the product manager for Microsoft Excel. Ask yourself: what are the barriers to switching? What keeps users from becoming Excel customers tomorrow? Think of these barriers as an obstacle course that people have to run before you can count them as your customers. If you start out with a field of 1000 runners, about half of them will trip on the tires; half of the survivors won't be strong enough to jump the wall; half of *those* survivors will fall off the rope ladder into the mud, and so on, until only 1 or 2 people actually overcome all the hurdles. With 8 or 9 barriers, *everybody* will have one non-negotiable deal killer.
>
> This calculus means that eliminating barriers to switching is the most important thing you have to do if you want to take over an existing market, because eliminating *just one barrier* will likely *double* your sales. Eliminate two barriers, and you'll double your sales again. Microsoft looked at the list of Lotus 123 barriers and worked on *all of them*:

Barrier	Solution
They have to know about Excel and know that it's better	Advertise Excel, send out demo disks, and tour the country showing it off
They have to buy Excel	Offer a special discount for former 123 users to switch to Excel
They have to buy Windows to run Excel	Make a runtime version of Windows which ships free with Excel
They have to convert their existing spreadsheets from 123 to Excel	Give Excel the capability to read 123 spreadsheets
They have to rewrite their keyboard macros which won't run in Excel	Give Excel the capability to run 123 macros
They have to learn a new user interface	Give Excel the ability to understand Lotus keystrokes, in case you were used to the old way of doing things
They need a faster computer with more memory	Wait for Moore's law to solve the problem of computer power

> And it worked pretty well. By incessant pounding on eliminating barriers, they slowly pried some market share away from Lotus.
>
> —Joel Spolsky, *Joel on Software*

Free software developers have had a long row to hoe. As proprietary software has been the dominant model for decades, many geeks have had to live with one foot in each world; Linux program-

mers today are often forced to use Macs, Windows and other proprietary software.

The good news is that a Linux PC does a reasonably good job of interoperating with proprietary hardware and software, especially Microsoft technology. Today, Linux supports the Microsoft Office file format, Windows Media file formats and protocols, MSN Instant Messenger protocols, Microsoft file systems, C#, and much more. The implementations aren't perfect, but just the most popular 95% of a standard is good enough for most. In fact, by tackling the easy and more important portions, the codebases are smaller and simpler than their Microsoft counterparts.

...

Standards & Web

From: Bill Gates
Sent: Saturday, December 5, 1998
To: Bob Muglia, Jon DeVann, Steven Sinofsky
Subject: Office rendering

One thing we have got to change in our strategy - allowing Office documents to be rendered very well by other people's browsers is one of the most destructive things we could do to the company.

We have to stop putting any effort into this and make sure that Office documents very well depends on PROPRIETARY IE capabilities.

Anything else is suicide for our platform. This is a case where Office has to avoid doing something to destroy Windows.

I would be glad to explain at a greater length.

Likewise this love of the standard DAV in Office/Exchange is a huge problem. I would also like to make sure people understand this as well.

To use the Internet, you need software that supports two big standards: TCP/HTTP and HTML. There is no “HTML” standard competing with an “HTMM” standard, as the idea is silly on its face, yet such redundancies exist in many other areas in the world of bits today. When you can't agree on a file format, your ability to exchange information goes from 1 to 0.

...

Digital Audio

It is the mess of proprietary standards and patent restrictions that are impeding the progress to digital audio. Proprietary software companies have been pushing their standards down our throat. If you insert a music CD into a Windows computer, it wants to rip the audio into WMA, a proprietary format which the Mac OS doesn't support out of the box. If you insert that CD into a Mac, it rips the music into AAC, a format that Windows doesn't support by default.

I'm not sure what should be done about this colossal mess. We should keep trying to pick a standard format for audio, like JPEG has become for still images. The best candidates are OGG and MP3. MP3 is an old format and while it is not considered state of the art, it is efficient enough. MP3's primary problem is that there are a number of companies with patent claims against it. If the industry

cannot agree to end the patent hassles, it should adopt OGG or some other fully free format and tell those MP3 license owners to take a long walk off a short pier.

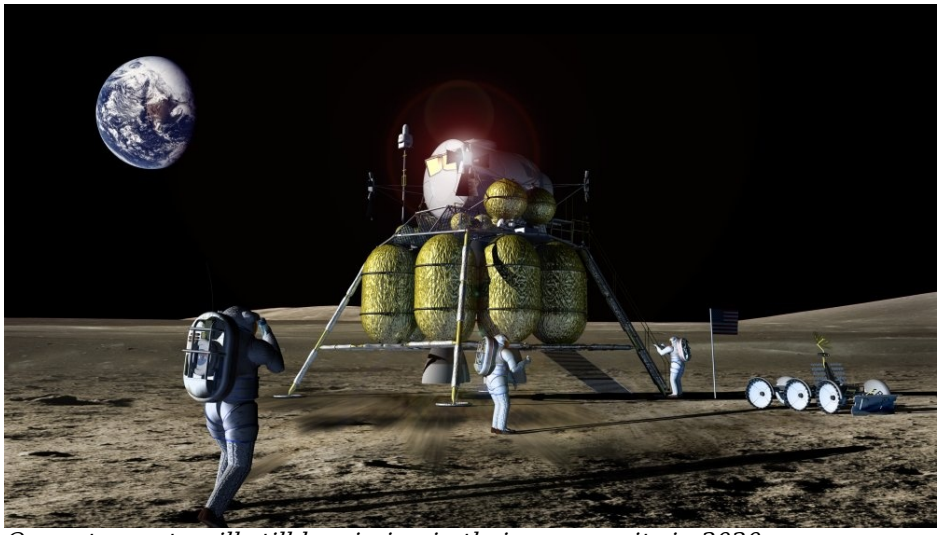
If we could finally agree on a free digital music format, we could finally have digital music — which will also open up many possibilities of a richer connection between artist and consumer. Whatever format is chosen, we need to define the number of bits per second required to achieve transparency.[1] We also need to come up with a standard streaming protocol and video format, but I won't even get into that here; we must crawl before we can walk.

…

1 Transparency is defined as audio that is of such high quality it cannot be distinguished from a CD. Transparency *should* mean passing the aforementioned listening test, plus when you convert it to other transparent codecs and back, even 100 times, the quality is not diminished. To achieve that we do not need to go to lossless compression, which is five to seven times bigger.

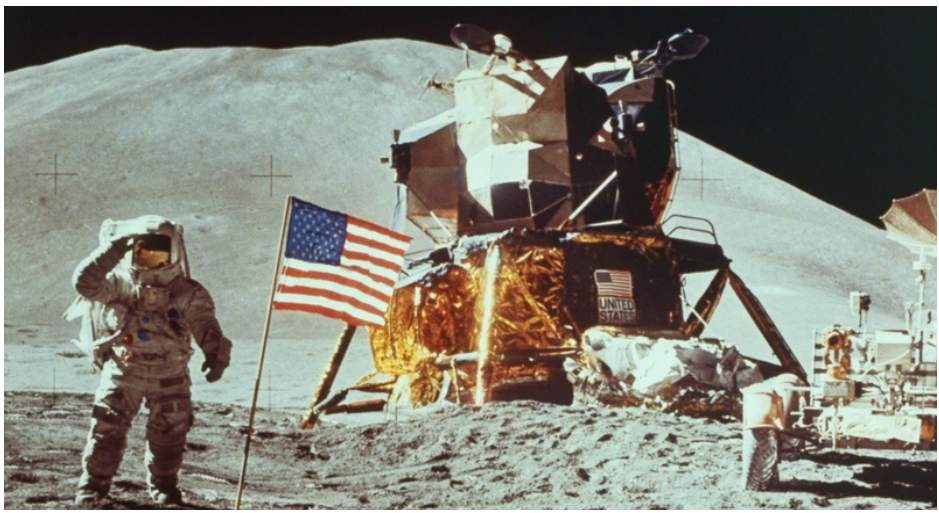
Space Elevator in 7

If NASA follows through on its 2004 vision to retire the Space Shuttle and go back to rockets, and go to the moon again, this is NASA's own imagery of what we will be looking at on DrudgeReport.com in 2020.



Our astronauts will still be pissing in their space suits in 2020.

According to NASA, the above is what we will see in 2020, but if you squint your eyes, it looks just like 1969:



All this was done without things we would call computers.

Only a government bureaucracy can make such little progress in 50 years and consider it business as usual.

...

If you want to know why we have not been back to the moon for decades, here is an analogy:

What would taking delivery of this car cost you?
A Californian buys a car made in Japan. The car is shipped in its own car carrier. The car is off-loaded in the port of Los Angeles. The freighter is then sunk.

The latest in propulsion technology is electrical ion drives which accelerate atoms 20 times faster than chemical rockets, which mean you need *much* less fuel. The inefficiency of our current chemical rockets is what is preventing man from colonizing space. Our simple modern rockets might be cheaper than our complicated old Space Shuttle, but it will still cost thousands of dollars per pound to get to LEO, a fancy acronym for 200 miles away. Working on chemical rockets today is the technological equivalent of polishing a dusty turd, yet this is what our esteemed NASA is doing.

Excerpt notes: I hope this gives you a sampling of what is in the book. I believe there is much food for thought. I worked very hard on it for 2 years, a number of the ideas are not widely accepted, even in the free software community, and most of it should be news to my former Microsoft co-workers!

This document is free, but please send people to:

http://keithcu.com/SoftwareWarsExcerpt.pdf *to download their own copy as I may update this document, I'm trying to keep track of the number of copies out there, and do other experiments in copyright.*

keithcu.com/SoftwareWars

-Keith

keithcu@gmail.com

Table of Contents