The Command Line in 2004

Belorussion translation

In July 2004 I found myself sitting alone in the dark, on the enclosed deck of a ferry boat oozing between fog-shrouded islands of the Alaskan coast. The scenery was haunting, but after the first three hours, I decided to occupy myself by finally reading Neal Stephenson's essay about the command-line. Halfway through it I began crossing things out, and scribbling comments in the margin. The essay was five years old, and in dire need of a fresh perspective.

Months later, I learned that Stephenson himself was dissatisfied with the essay. He wrote that it, "is now badly obsolete and probably needs a thorough revision." An "Ask Slashdot" poll quoted him as saying, "I keep meaning to update it, but if I'm honest with myself, I have to say this is unlikely."

Though I have fleshed out my original comments into longer, more structured pieces, it is not my intention to replace or revise Neal Stephenson's original writing. His original essay is a much more cohesive and entertaining read than my notes are. (He is a Writer, after all. I consider myself a code-monkey by comparison.) In fact, my notes do not hold together unless they use the original essay as a framework, and that's why his entire essay is reproduced here, with my comments color-coded. And yes, I have sought and obtained permission from Neal to do this.

In the Beginning was the Command Line

by Neal Stephenson

About twenty years ago Jobs and Wozniak, the founders of Apple, came up with the very strange idea of selling information processing machines for use in the home. The business took off, and its founders made a lot of money and received the credit they deserved for being daring visionaries. But around the same time, Bill Gates and Paul Allen came up with an idea even stranger and more fantastical: selling computer operating systems. This was much weirder than the idea of Jobs and Wozniak. A computer at least had some sort of physical reality to it. It came in a box, you could open it up and plug it in and watch lights blink. An operating system had no tangible incarnation at all. It arrived on a disk, of course, but the disk was, in effect, nothing more than the box that the OS came in. The product itself was a very long string of ones and zeroes that, when properly installed and coddled, gave you the ability to manipulate other very long strings of ones and zeroes. Even those few who actually understood what a computer operating system was were apt to think of it as a fantastically arcane engineering prodigy, like a breeder reactor or a U-2 spy plane, and not something that could ever be (in the parlance of high-tech) "productized."

Yet now the company that Gates and Allen founded is selling operating systems like Gillette sells razor blades. New releases of operating systems are launched as if they were Hollywood blockbusters, with celebrity endorsements, talk show appearances, and world tours. The market for them is vast enough that people worry about whether it has been monopolized by one company. Even the least technically-minded people in our society now have at least a hazy idea of what operating systems do; what is more, they have strong opinions about their relative merits. It is commonly understood, even by technically unsophisticated computer users, that if you have a piece of software that works on your Macintosh, and you move it over onto a Windows machine, it will not run. That this would, in fact, be a laughable and idiotic mistake, like nailing horseshoes to the tires of a Buick.

A person who went into a coma before Microsoft was founded, and woke up now, could pick up this morning's New York Times and understand everything in it--almost:

- **Item:** the richest man in the world made his fortune from-what? Railways? Shipping? Oil? No, operating systems.
- **Item:** the Department of Justice is tackling Microsoft's supposed OS monopoly with legal tools that were invented to restrain the power of Nineteenth-Century robber barons.

- **Item:** a woman friend of mine recently told me that she'd broken off a (hitherto) stimulating exchange of e-mail with a young man. At first he had seemed like such an intelligent and interesting guy, she said, but then "he started going all PC-versus-Mac on me."

What the hell is going on here? And does the operating system business have a future, or only a past? Here is my view, which is entirely subjective; but since I have spent a fair amount of time not only using, but programming, Macintoshes, Windows machines, Linux boxes and the BeOS, perhaps it is not so ill-informed as to be completely worthless. This is a subjective essay, more review than research paper, and so it might seem unfair or biased compared to the technical reviews you can find in PC magazines. But ever since the Mac came out, our operating systems have been based on metaphors, and anything with metaphors in it is fair game as far as I'm concerned.

MGBs, TANKS, AND BATMOBILES

Around the time that Jobs, Wozniak, Gates, and Allen were dreaming up these unlikely schemes, I was a teenager living in Ames, Iowa. One of my friends' dads had an old MGB sports car rusting away in his garage. Sometimes he would actually manage to get it running and then he would take us for a spin around the block, with a memorable look of wild youthful exhiliration on his face; to his worried passengers, he was a madman, stalling and backfiring around Ames, Iowa and eating the dust of rusty Gremlins and Pintos, but in his own mind he was Dustin Hoffman tooling across the Bay Bridge with the wind in his hair.

In retrospect, this was telling me two things about people's relationship to technology. One was that romance and image go a long way towards shaping their opinions. If you doubt it (and if you have a lot of spare time on your hands) just ask anyone who owns a Macintosh and who, on those grounds, imagines him- or herself to be a member of an oppressed minority group.

The other, somewhat subtler point, was that interface is very important. Sure, the MGB was a lousy car in almost every way that counted: balky, unreliable, underpowered. But it was fun to drive. It was responsive. Every pebble on the road was felt in the bones, every nuance in the pavement transmitted instantly to the driver's hands. He could listen to the engine and tell what was wrong with it. The steering responded immediately to commands from his hands. To us passengers it was a pointless exercise in going nowhere--about as interesting as peering over someone's shoulder while he punches numbers into a spreadsheet. But to the driver it was an experience. For a short time he was extending his body and his senses into a larger realm, and doing things that he couldn't do unassisted.

The analogy between cars and operating systems is not half bad, and so let me run with it for a moment, as a way of giving an executive summary of our situation today.

Imagine a crossroads where four competing auto dealerships are situated. One of them (Microsoft) is much, much bigger than the others. It started out years ago selling three-speed bicycles (MS-DOS); these were not perfect, but they worked, and when they broke you could easily fix them.

There was a competing bicycle dealership next door (Apple) that one day began selling motorized vehicles--expensive but attractively styled cars with their innards hermetically sealed, so that how they worked was something of a mystery.

The big dealership responded by rushing a moped upgrade kit (the original Windows) onto the market. This was a Rube Goldberg contraption that, when bolted onto a three-speed bicycle, enabled it to keep up, just barely, with Apple-cars. The users had to wear goggles and were always picking bugs out of their teeth while Apple owners sped along in hermetically sealed comfort, sneering out the windows. But the Micro-mopeds were cheap, and easy to fix compared with the Apple-cars, and their market share waxed.

Eventually the big dealership came out with a full-fledged car: a colossal station wagon (Windows 95). It had all the aesthetic appeal of a Soviet worker housing block, it leaked oil and blew gaskets, and it was an enormous success. A little later, they also came out with a hulking off-road vehicle intended for industrial users (Windows NT) which was no more beautiful than the station wagon, and only a little more reliable.

Since then there has been a lot of noise and shouting, but little has changed. The smaller dealership continues to sell sleek Euro-styled sedans and to spend a lot of money on advertising campaigns. They have had GOING OUT OF BUSINESS! signs taped up in their windows for so long that they have gotten all yellow and curly. The big one keeps making bigger and bigger station wagons and ORVs.

Now, things *have* changed. The Microsoft station wagons are no longer crash prone. Their cars and off-roaders merged into one vehicle, and they've diversified with boats and planes (A Tablet OS, the XBOX). The Apple-cars are no longer hermetically sealed. The "going out of business" signs are coming down.

On the other side of the road are two competitors that have come along more recently.

One of them (Be, Inc.) is selling fully operational Batmobiles (the BeOS). They are more beautiful and stylish even than the Euro-sedans, better designed, more technologically advanced, and at least as reliable as anything else on the market--and yet cheaper than the others.

BeOS is a beautiful operating system, but whether we're UNIX, Linux, BeOs, or Apple users - we're still all basically PWN3D by Microsoft. Five years since this essay hasn't changed that.

With one exception, that is: Linux, which is right next door, and which is not a business at all. It's a bunch of RVs, yurts, tepees, and geodesic domes set up in a field and organized by consensus. The people who live there are making tanks. These are not old-fashioned, cast-iron Soviet tanks; these are more like the M1 tanks of the U.S. Army, made of space-age materials and jammed with sophisticated technology from one end to the other. But they are better than Army tanks. They've been modified in such a way that they never, ever break down, are light and maneuverable enough to use on ordinary streets, and use no more fuel than a subcompact car. These tanks are being cranked out, on the spot, at a terrific pace, and a vast number of them are lined up along the edge of the road with keys in the ignition. Anyone who wants can simply climb into one and drive it away for free.

New users of Linux are almost always exposed to it through a member of the userbase, insuring that they have at least one person on-hand who can answer their inevitable questions and undo their horrible mistakes. The above is a romanticized description of the Linux experience, because it implies that the ubiquitous Linux veteran is not a factor. Unfortunately, Linux was not designed for end-to-end ease of use -- in that respect, it was not "designed" at all.

Today, the above description is actually what it would be like if OS X were released as open-source, ran flawlessly on all equipment, and was renamed Linux. And yes, now *my* personal bias is exposed. Better to expose it early and up front.

Customers come to this crossroads in throngs, day and night. Ninety percent of them go straight to the biggest dealership and buy station wagons or off-road vehicles. They do not even look at the other dealerships.

Of the remaining ten percent, most go and buy a sleek Euro-sedan, pausing only to turn up their noses at the philistines going to buy the station wagons and ORVs. If they even notice the people on the opposite side of the road, selling the cheaper, technically superior vehicles, these customers deride them cranks and half-wits.

The Batmobile outlet sells a few vehicles to the occasional car nut who wants a second vehicle to go with his station wagon, but seems to accept, at least for now, that it's a fringe player.

The group giving away the free tanks only stays alive because it is staffed by volunteers, who are lined up at the edge of the street with bullhorns, trying to draw customers' attention to this incredible situation. A typical conversation goes something like this:

Hacker with bullhorn: "Save your money! Accept one of our free tanks! It is invulnerable, and can drive across rocks and swamps at ninety miles an hour while getting a hundred miles to the gallon!"

Prospective station wagon buyer: "I know what you say is true...but...er...I don't know how to maintain a tank!"

Bullhorn: "You don't know how to maintain a station wagon either!"

Buyer: "But this dealership has mechanics on staff. If something goes wrong with my station wagon, I can take a day off work, bring it here, and pay them to work on it while I sit in the waiting room for hours, listening to elevator music."

Bullhorn: "But if you accept one of our free tanks we will send volunteers to your house to fix it for free while you sleep!"

Buyer: "Stay away from my house, you freak!"

Bullhorn: "But..."

Buyer: "Can't you see that everyone is buying station wagons?"

This is a metaphor with legs. However, it has one flaw that needs addressing: Windows and Linux are software, and Apple is a hardware company. This problem can be solved like many other problems are solved in the computer industry: By adding monkeys.

No, seriously. It works like this. Computer hardware has changed immeasurably in the last 30 years, and nowadays everything we do must be guided by an operating system. To illustrate that situtation with cars, I could say that all modern cars are so fancy and complicated that each one sold comes with a chauffeur who will do the driving for you.

For example, if you buy an Apple sedan, you also receive a little monkey in a snappy blue suit. Your personal X-Monkey (as the company calls him) is the ideal driver of your Apple sedan. He knows where everything is, feeds and washes himself, drives defensively, and will even tune up the car for you. X-Monkey will accept precise instructions like, "forward 10 feet, right 20 degrees", but he is smart enough to think on his own, so you can tell him "Drive me to a taco stand, then pick up Uncle Steve". He will also keep you out of trouble, by politely ignoring instructions like, "Run over that jogger", and "Floor it", when you're at a red light. Depending on your temperament, this could actually be a downside.

The X-Monkey comes from a line of monkeys originally bred by the military for the purpose of driving tanks. It's a good fit, because the modern Apple sedan is actually a tank in a fancy shell. The X-Monkey's only drawback is that he can *only* drive a car from Apple. Show him any other vehicle, and he won't even know how to operate the door lock.

Meanwhile, the free-thinking Linux people, displeased with genetic engineering, have created their own smart monkey chauffeurs through a massive international breeding program. Unlike the X-Monkey, the Linux Monkey is capable of driving any car, including the Apple sedan. If you could install a steering wheel on a log splitter, the Linux Monkey could drive it for you. The catch is, you have to train the Linux Monkey yourself. Fortunately there are experts everywhere who will help you out, and the Linux Monkey trains easily.

The Microsoft Gorilla, on the other hand, cannot be trained. Instead, you must keep rephrasing your directions until the MS Gorilla can comprehend them. He consumes both front seats, lowering the mileage of your car, and blocking most of your view. Though he sounds like a bad deal, MS Gorilla is actually extremely popular, because he looks impressive, drives aggressively, and keeps his mouth shut. If you speak in his limited vocabulary, he will take you Where You Want To Go Today ... especially if he can plow monkeys off the intervening road. However, if you touch anything on the dashboard, or try to haggle with him over the exact route, he may become irritated and casually drive your car into a telephone pole. People learn to not argue.

The point to this altered metaphor is that the Microsoft dealership, and the Linux collective, do not really make cars at all. All those shiny automobiles sitting on the lot and lined up on the street corner are re-branded vehicles, manufactured by other companies. However, their modern instrument panels are so confusing that they'd be useless without a chauffeur. ... And the Microsoft dealership gets a cut from the price of every vehicle that leaves their lot, piloted by the Microsoft Gorilla.

If you were so inclined, you could purchase a car from them, drive to the sidewalk, and kick the gorilla out onto the curb. The Linux Monkey can hop right in and start driving for you. Of course, Microsoft already

has your money, and what are you going to do with a spare gorilla?

Contrast this with the Apple dealership, that personally designs and assembles every Apple sedan. When a sedan leaves their lot, they pocket the whole amount. You could still kick out the X-Monkey any time, but why would you? The Linux Monkey is basically the same, without the training.

BIT-FLINGER

The connection between cars, and ways of interacting with computers, wouldn't have occurred to me at the time I was being taken for rides in that MGB. I had signed up to take a computer programming class at Ames High School. After a few introductory lectures, we students were granted admission into a tiny room containing a teletype, a telephone, and an old-fashioned modem consisting of a metal box with a pair of rubber cups on the top (note: many readers, making their way through that last sentence, probably felt an initial pang of dread that this essay was about to turn into a tedious, codgerly reminiscence about how tough we had it back in the old days; rest assured that I am actually positioning my pieces on the chessboard, as it were, in preparation to make a point about truly hip and up-to-the minute topics like Open Source Software). The teletype was exactly the same sort of machine that had been used, for decades, to send and receive telegrams. It was basically a loud typewriter that could only produce UPPERCASE LETTERS. Mounted to one side of it was a smaller machine with a long reel of paper tape on it, and a clear plastic hopper underneath.

In order to connect this device (which was not a computer at all) to the Iowa State University mainframe across town, you would pick up the phone, dial the computer's number, listen for strange noises, and then slam the handset down into the rubber cups. If your aim was true, one would wrap its neoprene lips around the earpiece and the other around the mouthpiece, consummating a kind of informational soixante-neuf. The teletype would shudder as it was possessed by the spirit of the distant mainframe, and begin to hammer out cryptic messages.

Since computer time was a scarce resource, we used a sort of batch processing technique. Before dialing the phone, we would turn on the tape puncher (a subsidiary machine bolted to the side of the teletype) and type in our programs. Each time we depressed a key, the teletype would bash out a letter on the paper in front of us, so we could read what we'd typed; but at the same time it would convert the letter into a set of eight binary digits, or bits, and punch a corresponding pattern of holes across the width of a paper tape. The tiny disks of paper knocked out of the tape would flutter down into the clear plastic hopper, which would slowly fill up what can only be described as actual bits. On the last day of the school year, the smartest kid in the class (not me) jumped out from behind his desk and flung several quarts of these bits over the head of our teacher, like confetti, as a sort of semi-affectionate practical joke. The image of this man sitting there, gripped in the opening stages of an atavistic fight-or-flight reaction, with millions of bits (megabytes) sifting down out of his hair and into his nostrils and mouth, his face gradually turning purple as he built up to an explosion, is the single most memorable scene from my formal education.

Anyway, it will have been obvious that my interaction with the computer was of an extremely formal nature, being sharply divided up into different phases, viz.: (1) sitting at home with paper and pencil, miles and miles from any computer, I would think very, very hard about what I wanted the computer to do, and translate my intentions into a computer language--a series of alphanumeric symbols on a page. (2) I would carry this across a sort of informational cordon sanitaire (three miles of snowdrifts) to school and type those letters into a machine--not a computer--which would convert the symbols into binary numbers and record them visibly on a tape. (3) Then, through the rubber-cup modem, I would cause those numbers to be sent to the university mainframe, which would (4) do arithmetic on them and send different numbers back to the teletype. (5) The teletype would convert these numbers back into letters and hammer them out on a page and (6) I, watching, would construe the letters as meaningful symbols.

The division of responsibilities implied by all of this is admirably clean: computers do arithmetic on bits of information. Humans construe the bits as meaningful symbols. But this distinction is now being blurred, or at least complicated, by the advent of modern operating systems that use, and frequently abuse, the power of metaphor to make computers accessible to a larger audience. Along the way--possibly because

of those metaphors, which make an operating system a sort of work of art--people start to get emotional, and grow attached to pieces of software in the way that my friend's dad did to his MGB.

People who have only interacted with computers through graphical user interfaces like the MacOS or Windows--which is to say, almost everyone who has ever used a computer--may have been startled, or at least bemused, to hear about the telegraph machine that I used to communicate with a computer in 1973. But there was, and is, a good reason for using this particular kind of technology. Human beings have various ways of communicating to each other, such as music, art, dance, and facial expressions, but some of these are more amenable than others to being expressed as strings of symbols. Written language is the easiest of all, because, of course, it consists of strings of symbols to begin with. If the symbols happen to belong to a phonetic alphabet (as opposed to, say, ideograms), converting them into bits is a trivial procedure, and one that was nailed, technologically, in the early nineteenth century, with the introduction of Morse code and other forms of telegraphy.

We had a human/computer interface a hundred years before we had computers. When computers came into being around the time of the Second World War, humans, quite naturally, communicated with them by simply grafting them on to the already-existing technologies for translating letters into bits and vice versa: teletypes and punch card machines.

These embodied two fundamentally different approaches to computing. When you were using cards, you'd punch a whole stack of them and run them through the reader all at once, which was called batch processing. You could also do batch processing with a teletype, as I have already described, by using the paper tape reader, and we were certainly encouraged to use this approach when I was in high school. But--though efforts were made to keep us unaware of this--the teletype could do something that the card reader could not. On the teletype, once the modem link was established, you could just type in a line and hit the return key. The teletype would send that line to the computer, which might or might not respond with some lines of its own, which the teletype would hammer out--producing, over time, a transcript of your exchange with the machine. This way of doing it did not even have a name at the time, but when, much later, an alternative became available, it was retroactively dubbed the Command Line Interface.

When I moved on to college, I did my computing in large, stifling rooms where scores of students would sit in front of slightly updated versions of the same machines and write computer programs: these used dot-matrix printing mechanisms, but were (from the computer's point of view) identical to the old teletypes. By that point, computers were better at time-sharing--that is, mainframes were still mainframes, but they were better at communicating with a large number of terminals at once. Consequently, it was no longer necessary to use batch processing. Card readers were shoved out into hallways and boiler rooms, and batch processing became a nerds-only kind of thing, and consequently took on a certain eldritch flavor among those of us who even knew it existed. We were all off the Batch, and on the Command Line, interface now--my very first shift in operating system paradigms, if only I'd known it.

A huge stack of accordion-fold paper sat on the floor underneath each one of these glorified teletypes, and miles of paper shuddered through their platens. Almost all of this paper was thrown away or recycled without ever having been touched by ink--an ecological atrocity so glaring that those machines soon replaced by video terminals--so-called "glass teletypes"--which were quieter and didn't waste paper. Again, though, from the computer's point of view these were indistinguishable from World War II-era teletype machines. In effect we still used Victorian technology to communicate with computers until about 1984, when the Macintosh was introduced with its Graphical User Interface. Even after that, the Command Line continued to exist as an underlying stratum--a sort of brainstem reflex--of many modern computer systems all through the heyday of Graphical User Interfaces, or GUIs as I will call them from now on.

As it still does, thank goodness.

GUIs

Now the first job that any coder needs to do when writing a new piece of software is to figure out how to take the information that is being worked with (in a graphics program, an image; in a spreadsheet, a grid of numbers) and turn it into a linear string of bytes. These strings of bytes are commonly called files

or (somewhat more hiply) streams. They are to telegrams what modern humans are to Cro-Magnon man, which is to say the same thing under a different name. All that you see on your computer screen--your Tomb Raider, your digitized voice mail messages, faxes, and word processing documents written in thirty-seven different typefaces--is still, from the computer's point of view, just like telegrams, except much longer, and demanding of more arithmetic.

The quickest way to get a taste of this is to fire up your web browser, visit a site, and then select the View/Document Source menu item. You will get a bunch of computer code that looks something like this:

```
<HTML>

<HEAD>

<TITLE>Welcome to the Avon Books Homepage</TITLE>
</HEAD>

<MAP NAME="left0199">
<AREA SHAPE="rect" COORDS="16,56,111,67" HREF="/bard/">
<AREA SHAPE="rect" COORDS="14,77,111,89" HREF="/eos/">
<AREA SHAPE="rect" COORDS="17,98,112,110" HREF="/twilight/">
<AREA SHAPE="rect" COORDS="18,119,112,131"
HREF="/avon_user/category.html?category_id=271">
<AREA SHAPE="rect" COORDS="19,140,112,152"
HREF="http://www.goners.com/">
<AREA SHAPE="rect" COORDS="18,161,111,173"
HREF="http://www.spikebooks.com/">
<AREA SHAPE="rect" COORDS="2,181,112,195"
HREF="/avon_user/category.html?category_id=277">
<AREA SHAPE="rect" COORDS="9,203,112,216" HREF="/chathamisland/">
<AREA SHAPE="rect" COORDS="7,223,112,236"
HREF="/avon_user/search.html">
</MAP>

<BODY TEXT="#478CFF" LINK="#FFFFFF" VLINK="#000000" ALINK="#478CFF"
BGCOLOR="#003399">
<TABLE BORDER="0" WIDTH="600" CELLPADDING="0" CELLSPACING="0">

<TR VALIGN=TOP>

<TD ROWSPAN="3">
<A HREF="/cgi-bin/imagemap/maps/left.gif.map"><IMG SRC="/avon/images/home/nav/left0199.gif"
WIDTH="113" HEIGHT="280" BORDER="0" USEMAP="#left0199"></A></TD><TD ROWSPAN="3">
<IMG SRC="/avon/images/home/homepagejan98/2ndleft.gif" WIDTH="144" HEIGHT="280"
BORDER="0"></TD><TD><A HREF="/avon/about.html"><IMG
SRC="/avon/images/home/homepagejan98/aboutavon.gif" ALT="About Avon Books" WIDTH="199"
HEIGHT="44" BORDER="0"></A></TD><TD ROWSPAN="3"><A HREF="/avon/fiction/guides.html">
<IMG SRC="/avon/images/home/feb98/right1.gif" ALT="Reading Groups" WIDTH="165" HEIGHT="121"
BORDER="0"></A><BR><A HREF="/avon/feature/feb99/crook.html"><IMG
SRC="/avon/images/home/feb99/crook_text.gif" ALT="The Crook Factory" WIDTH="165" HEIGHT="96"
BORDER="0"></A><BR><A HREF="http://apps.hearstnewmedia.com/cgi-
bin/gx.cgi/AppLogic+APPSSURVEYS Questionnaire?domain_id=182&survey_id=541"><IMG
SRC="/avon/images/home/feb99/env_text.gif" ALT="The Envelope Please" WIDTH="165" HEIGHT="63"
BORDER="0"></A></TD> </TR>

<TR VALIGN=TOP><TD><IMG SRC="/avon/images/home/feb98/main.gif" WIDTH="199" HEIGHT="182"
```

BORDER="0"></TD></TR><TR VALIGN=TOP><TD><A HREF="/avon/feature/jan99/sitchin.html"><IMG SRC="/avon/images/home/jan99/sitchin_text.gif" WIDTH="199" HEIGHT="54" BORDER="0"></A></TD></TR><TR VALIGN=TOP><TD COLSPAN="4"><IMG SRC="/avon/images/home/jan99/avon_bottom_beau.gif" WIDTH="622" HEIGHT="179" BORDER="0" USEMAP="#bottom"></TD></TR><TR><TD ALIGN=CENTER VALIGN=TOP COLSPAN="4"><FONT SIZE="2" FACE="ARIAL,COURIER"><PRE>

</PRE><A HREF="/avon/ordering.html">How to order</A> | <A HREF="/avon/faq.html#manu">How to submit a Manuscript</A> | <A HREF="mailto:avonweb@hearst.com">Contact us</A> | <A HREF="/avon/policy.html">Privacy Policy</A></FONT>

<P>
</FONT></TD>

</TR>

</TABLE>

</BODY>

</HTML>

This crud is called HTML (HyperText Markup Language) and it is basically a very simple programming language instructing your web browser how to draw a page on a screen. Anyone can learn HTML and many people do. The important thing is that no matter what splendid multimedia web pages they might represent, HTML files are just telegrams.

When Ronald Reagan was a radio announcer, he used to call baseball games by reading the terse descriptions that trickled in over the telegraph wire and were printed out on a paper tape. He would sit there, all by himself in a padded room with a microphone, and the paper tape would eke out of the machine and crawl over the palm of his hand printed with cryptic abbreviations. If the count went to three and two, Reagan would describe the scene as he saw it in his mind's eye: "The brawny left-hander steps out of the batter's box to wipe the sweat from his brow. The umpire steps forward to sweep the dirt from home plate." and so on. When the cryptogram on the paper tape announced a base hit, he would whack the edge of the table with a pencil, creating a little sound effect, and describe the arc of the ball as if he could actually see it. His listeners, many of whom presumably thought that Reagan was actually at the ballpark watching the game, would reconstruct the scene in their minds according to his descriptions.

This is exactly how the World Wide Web works: the HTML files are the pithy description on the paper tape, and your Web browser is Ronald Reagan. The same is true of Graphical User Interfaces in general.

So an OS is a stack of metaphors and abstractions that stands between you and the telegrams, and embodying various tricks the programmer used to convert the information you're working with--be it images, e-mail messages, movies, or word processing documents--into the necklaces of bytes that are the only things computers know how to work with. When we used actual telegraph equipment (teletypes) or their higher-tech substitutes ("glass teletypes," or the MS-DOS command line) to work with our computers, we were very close to the bottom of that stack. When we use most modern operating systems, though, our interaction with the machine is heavily mediated. Everything we do is interpreted and translated time and again as it works its way down through all of the metaphors and abstractions.

This conveys a good impression of how complex our operating systems have become, but it also implies that the hardware itself, the physical object that we call a computer, has changed relatively little.

Computer-like circuitry has burrowed its way into almost every kind of device, in every arena of human progress. The microchip embedded in a rice cooker, for example, could potentially respond to a command-line interface. We could wire a teletype to most of the appliances we use in a day. But when someone says, "Use the computer!", the image that pops into our collective imagination is relatively well defined.

Consider the garden-variety machine from Dell. We pry open the cardboard box, wrestle the styrofoam jaws apart, and dump the thing on a tabletop. "There!" we say. "A brand-new computer!" But by the historical definition, we're actually looking at dozens of computers. A heap of computers. A computing collective.

Here's an abridged list of items inside the Dell machine that could be computers in their own right:

- The CPU
- The cache manager inside the CPU
- The on-board sound chip
- The USB controller
- The power-management system
- The network controller
- The graphics CPU
- The hard-drive controller
- The microcontroller inside the hard drive
- The settings-management CPU inside the display
- The key encoder inside the keyboard

Clearly a PC has become something more than a teletype. It has not just evolved from its ancestors, but actually *contains* multiple copies of its ancestors. If we wanted to treat a modern PC like a teletype, we would have to ignore all but one of these embedded computers, and throw the mouse, printer, disk drives, speakers, and game controllers in the trash can.

But we don't call our new Dell machine a "computing collective". We consider it one device, with a singular name. And our concept of an "Operating System" has evolved right along, neatly obscuring this complexity. The ways that we interact with a computer have always been simple -- punch buttons, move lumps on tabletops, occasionally shout and kick parts of it -- and any claim we had to a non-conceptual "closeness" with the processing itself died out around the time mechanical relays were replaced with transistors. That's why Mr. Stephenson is invoking the metaphor of *telegrams*. He is describing the quality of our relationship to the *digital information* that we wish to manipulate.

But that means that the crucial separation here is not between the computer (hardware) and the Operating System (software). Those are so deeply integrated these days that they have effectively merged into the same beast. The crucial division is between ourselves, and our information. And how that division is elucidated by the computer, with hardware and software working in tandem, really determines how useful the computer is to us. In other words, it's all about User Interface, and even though "In the Beginning, There Was the Command Line", it's also true that In The Beginning, Information Took Fewer Forms.

> The Macintosh OS was a revolution in both the good and bad senses of that word. Obviously it was true that command line interfaces were not for everyone, and that it would be a good thing to make computers more accessible to a less technical audience--if not for altruistic reasons, then because those sorts of people constituted an incomparably vaster market. It was clear the the Mac's engineers saw a whole new country stretching out before them; you could almost hear them muttering, "Wow! We don't have to be bound by files as linear streams of bytes anymore, vive la revolution, let's see how far we can take this!" No command line interface was available on the Macintosh; you talked to it with the mouse, or not at all. This was a statement of sorts, a credential of revolutionary purity. It seemed that the designers of the Mac intended to sweep Command Line Interfaces into the dustbin of history.

Though Apple and Microsoft still design their operating system as though the common user should never see a command line, they have both learned that providing access to one is important. More on that later.

> My own personal love affair with the Macintosh began in the spring of 1984 in a computer store in Cedar Rapids, Iowa, when a friend of mine--coincidentally, the son of the MGB owner--showed me a Macintosh running MacPaint, the revolutionary drawing program. It ended in July of 1995 when I tried to save a big important file on my Macintosh Powerbook and instead instead of doing so, it annihilated the

data so thoroughly that two different disk crash utility programs were unable to find any trace that it had ever existed. During the intervening ten years, I had a passion for the MacOS that seemed righteous and reasonable at the time but in retrospect strikes me as being exactly the same sort of goofy infatuation that my friend's dad had with his car.

The introduction of the Mac triggered a sort of holy war in the computer world. Were GUIs a brilliant design innovation that made computers more human-centered and therefore accessible to the masses, leading us toward an unprecedented revolution in human society, or an insulting bit of audiovisual gimcrackery dreamed up by flaky Bay Area hacker types that stripped computers of their power and flexibility and turned the noble and serious work of computing into a childish video game?

This debate actually seems more interesting to me today than it did in the mid-1980s. But people more or less stopped debating it when Microsoft endorsed the idea of GUIs by coming out with the first Windows. At this point, command-line partisans were relegated to the status of silly old grouches, and a new conflict was touched off, between users of MacOS and users of Windows.

There was plenty to argue about. The first Macintoshes looked different from other PCs even when they were turned off: they consisted of one box containing both CPU (the part of the computer that does arithmetic on bits) and monitor screen. This was billed, at the time, as a philosophical statement of sorts: Apple wanted to make the personal computer into an appliance, like a toaster. But it also reflected the purely technical demands of running a graphical user interface. In a GUI machine, the chips that draw things on the screen have to be integrated with the computer's central processing unit, or CPU, to a far greater extent than is the case with command-line interfaces, which until recently didn't even know that they weren't just talking to teletypes.

This distinction was of a technical and abstract nature, but it became clearer when the machine crashed (it is commonly the case with technologies that you can get the best insight about how they work by watching them fail). When everything went to hell and the CPU began spewing out random bits, the result, on a CLI machine, was lines and lines of perfectly formed but random characters on the screen--known to cognoscenti as "going Cyrillic." But to the MacOS, the screen was not a teletype, but a place to put graphics; the image on the screen was a bitmap, a literal rendering of the contents of a particular portion of the computer's memory. When the computer crashed and wrote gibberish into the bitmap, the result was something that looked vaguely like static on a broken television set--a "snow crash."

And even after the introduction of Windows, the underlying differences endured; when a Windows machine got into trouble, the old command-line interface would fall down over the GUI like an asbestos fire curtain sealing off the proscenium of a burning opera. When a Macintosh got into trouble it presented you with a cartoon of a bomb, which was funny the first time you saw it.

That bomb also became the subject of myriad parodies, too. Though it's general meaning was obvious, it was really the least informative error message possible, and came to symbolize the condescending nature of the GUI. Bomb icons showed up on computer screens in comics, cartoons, and feature films, often paired with a ridiculous upbeat "DOINK!" noise.

And these were by no means superficial differences. The reversion of Windows to a CLI when it was in distress proved to Mac partisans that Windows was nothing more than a cheap facade, like a garish afghan flung over a rotted-out sofa. They were disturbed and annoyed by the sense that lurking underneath Windows' ostensibly user-friendly interface was--literally--a subtext.

Those early mac people must be spinning in their cubicles now.

For their part, Windows fans might have made the sour observation that all computers, even Macintoshes, were built on that same subtext, and that the refusal of Mac owners to admit that fact to themselves seemed to signal a willingness, almost an eagerness, to be duped.

Anyway, a Macintosh had to switch individual bits in the memory chips on the video card, and it had to do it very fast, and in arbitrarily complicated patterns. Nowadays this is cheap and easy, but in the technological regime that prevailed in the early 1980s, the only realistic way to do it was to build the motherboard (which contained the CPU) and the video system (which contained the memory that was

mapped onto the screen) as a tightly integrated whole--hence the single, hermetically sealed case that made the Macintosh so distinctive.

When Windows came out, it was conspicuous for its ugliness, and its current successors, Windows 95 and Windows NT, are not things that people would pay money to look at either. Microsoft's complete disregard for aesthetics gave all of us Mac-lovers plenty of opportunities to look down our noses at them. That Windows looked an awful lot like a direct ripoff of MacOS gave us a burning sense of moral outrage to go with it. Among people who really knew and appreciated computers (hackers, in Steven Levy's non-pejorative sense of that word) and in a few other niches such as professional musicians, graphic artists and schoolteachers, the Macintosh, for a while, was simply the computer. It was seen as not only a superb piece of engineering, but an embodiment of certain ideals about the use of technology to benefit mankind, while Windows was seen as a pathetically clumsy imitation and a sinister world domination plot rolled into one. So very early, a pattern had been established that endures to this day: people dislike Microsoft, which is okay; but they dislike it for reasons that are poorly considered, and in the end, self-defeating.

Tell that to the small-business IT guy who plugs the network cable into the back of his freshly installed Windows XP box, only to have it infected with a virus in less than *20 seconds*. Or the publishing house that spent ten thousand dollars upgrading Word, only to discover that their documents now looked like garbage to every editor and author they worked with. "Poorly considered and self-defeating" could just as easily describe the actions of a Microsoft *customer*.

CLASS STRUGGLE ON THE DESKTOP

Now that the Third Rail has been firmly grasped, it is worth reviewing some basic facts here: like any other publicly traded, for-profit corporation, Microsoft has, in effect, borrowed a bunch of money from some people (its stockholders) in order to be in the bit business. As an officer of that corporation, Bill Gates has one responsibility only, which is to maximize return on investment. He has done this incredibly well. Any actions taken in the world by Microsoft-any software released by them, for example--are basically epiphenomena, which can't be interpreted or understood except insofar as they reflect Bill Gates's execution of his one and only responsibility.

It follows that if Microsoft sells goods that are aesthetically unappealing, or that don't work very well, it does not mean that they are (respectively) philistines or half-wits. It is because Microsoft's excellent management has figured out that they can make more money for their stockholders by releasing stuff with obvious, known imperfections than they can by making it beautiful or bug-free. This is annoying, but (in the end) not half so annoying as watching Apple inscrutably and relentlessly destroy itself.

VERY TRUE.

Hostility towards Microsoft is not difficult to find on the Net, and it blends two strains: resentful people who feel Microsoft is too powerful, and disdainful people who think it's tacky. This is all strongly reminiscent of the heyday of Communism and Socialism, when the bourgeoisie were hated from both ends: by the proles, because they had all the money, and by the intelligentsia, because of their tendency to spend it on lawn ornaments. Microsoft is the very embodiment of modern high-tech prosperity--it is, in a word, bourgeois--and so it attracts all of the same gripes.

The opening "splash screen" for Microsoft Word 6.0 summed it up pretty neatly: when you started up the program you were treated to a picture of an expensive enamel pen lying across a couple of sheets of fancy-looking handmade writing paper. It was obviously a bid to make the software look classy, and it might have worked for some, but it failed for me, because the pen was a ballpoint, and I'm a fountain pen man. If Apple had done it, they would've used a Mont Blanc fountain pen, or maybe a Chinese calligraphy brush. And I doubt that this was an accident. Recently I spent a while re-installing Windows NT on one of my home computers, and many times had to double-click on the "Control Panel" icon. For reasons that are difficult to fathom, this icon consists of a picture of a clawhammer and a chisel or screwdriver resting on top of a file folder.

One could easily cite this as an example of how a lack of competition resulted in a lack of innovation. There was no reason not to think up a more sensible icon for the Control Panel. Yet for a *dozen years*, it remained a hammer and a chisel, until Microsoft replaced it with ... what's this ... a pencil and a checkbox? As before, users are expected to understand what it means through indoctrination alone. It conveys no information to a newcomer, and while that lack of information may seem trivial at first, an entire operating system littered with equally arbitrary symbols and widgets can steer the novice to a psychotic episode.

> These aesthetic gaffes give one an almost uncontrollable urge to make fun of Microsoft, but again, it is all beside the point--if Microsoft had done focus group testing of possible alternative graphics, they probably would have found that the average mid-level office worker associated fountain pens with effete upper management toffs and was more comfortable with ballpoints. Likewise, the regular guys, the balding dads of the world who probably bear the brunt of setting up and maintaining home computers, can probably relate better to a picture of a clawhammer--while perhaps harboring fantasies of taking a real one to their balky computers.
>
> This is the only way I can explain certain peculiar facts about the current market for operating systems, such as that ninety percent of all customers continue to buy station wagons off the Microsoft lot while free tanks are there for the taking, right across the street.

We also have to consider a little thing called ... advertising. And perhaps, aggressively pursued contracts. Embrace-and-extend, FUD, and free seminars (Punch and Pie!).

> A string of ones and zeroes was not a difficult thing for Bill Gates to distribute, one he'd thought of the idea. The hard part was selling it--reassuring customers that they were actually getting something in return for their money.
>
> Anyone who has ever bought a piece of software in a store has had the curiously deflating experience of taking the bright shrink-wrapped box home, tearing it open, finding that it's 95 percent air, throwing away all the little cards, party favors, and bits of trash, and loading the disk into the computer. The end result (after you've lost the disk) is nothing except some images on a computer screen, and some capabilities that weren't there before. Sometimes you don't even have that--you have a string of error messages instead. But your money is definitely gone. Now we are almost accustomed to this, but twenty years ago it was a very dicey business proposition. Bill Gates made it work anyway. He didn't make it work by selling the best software or offering the cheapest price. Instead he somehow got people to believe that they were receiving something in exchange for their money.
>
> The streets of every city in the world are filled with those hulking, rattling station wagons. Anyone who doesn't own one feels a little weird, and wonders, in spite of himself, whether it might not be time to cease resistance and buy one; anyone who does, feels confident that he has acquired some meaningful possession, even on those days when the vehicle is up on a lift in an auto repair shop.

That worldwide demographic of computer owners usually got Windows when it came preinstalled on their system. They think of the OS and the equipment as the same package, and that package has just as much physical presence to a middle-class consumer as any other home appliance. In the specific case of software purchased later, people were already well accustomed to the idea of paying money for ephemeral data on disposable media. Back then, the media were known as cassettes. (Of the VCR, Audio, and 8-Track kind.)

> All of this is perfectly congruent with membership in the bourgeoisie, which is as much a mental, as a material state. And it explains why Microsoft is regularly attacked, on the Net, from both sides. People who are inclined to feel poor and oppressed construe everything Microsoft does as some sinister Orwellian plot. People who like to think of themselves as intelligent and informed technology users are driven crazy by the clunkiness of Windows.
>
> Nothing is more annoying to sophisticated people to see someone who is rich enough to know better being tacky--unless it is to realize, a moment later, that they probably know they are tacky and they simply don't care and they are going to go on being tacky, and rich, and happy, forever. Microsoft therefore bears the same relationship to the Silicon Valley elite as the Beverly Hillbillies did to their fussy banker, Mr.

Drysdale--who is irritated not so much by the fact that the Clampetts moved to his neighborhood as by the knowledge that, when Jethro is seventy years old, he's still going to be talking like a hillbilly and wearing bib overalls, and he's still going to be a lot richer than Mr. Drysdale.

Even the hardware that Windows ran on, when compared to the machines put out by Apple, looked like white-trash stuff, and still mostly does. The reason was that Apple was and is a hardware company, while Microsoft was and is a software company. Apple therefore had a monopoly on hardware that could run MacOS, whereas Windows-compatible hardware came out of a free market. The free market seems to have decided that people will not pay for cool-looking computers; PC hardware makers who hire designers to make their stuff look distinctive get their clocks cleaned by Taiwanese clone makers punching out boxes that look as if they belong on cinderblocks in front of someone's trailer. But Apple could make their hardware as pretty as they wanted to and simply pass the higher prices on to their besotted consumers, like me. Only last week (I am writing this sentence in early Jan. 1999) the technology sections of all the newspapers were filled with adulatory press coverage of how Apple had released the iMac in several happenin' new colors like Blueberry and Tangerine.

Yeeech, the iMac. Worst mouse ever. On the other hand, I've got one presently acting as a company-wide mailserver in the basement.

Stephenson's description of Apple is painful, because it's accurate. In the late 90's the company had little going for it except the fading halo of previous successes. The iMac was seen by some as a return to the early principles of tight integration that made the Macintosh a hit in the first place, but the happy-go-lucky ad campaign embarrassed die-hard users just as surely as it attracted new ones. In retrospect, it's tempting to assume that the iMac was released mostly as a stopgap product to gain time and money for addressing the real problem: OS 9.

But the big question remains the same as ever: "Why are the Apple users besotted? What has hypnotized them so thoroughly? Are Apple devices really so pleasurable to use?"

Apple has always insisted on having a hardware monopoly, except for a brief period in the mid-1990s when they allowed clone-makers to compete with them, before subsequently putting them out of business. Macintosh hardware was, consequently, expensive. You didn't open it up and fool around with it because doing so would void the warranty. In fact the first Mac was specifically designed to be difficult to open--you needed a kit of exotic tools, which you could buy through little ads that began to appear in the back pages of magazines a few months after the Mac came out on the market. These ads always had a certain disreputable air about them, like pitches for lock-picking tools in the backs of lurid detective magazines.

This monopolistic policy can be explained in at least three different ways.

THE CHARITABLE EXPLANATION is that the hardware monopoly policy reflected a drive on Apple's part to provide a seamless, unified blending of hardware, operating system, and software. There is something to this. It is hard enough to make an OS that works well on one specific piece of hardware, designed and tested by engineers who work down the hallway from you, in the same company. Making an OS to work on arbitrary pieces of hardware, cranked out by rabidly entrepeneurial clonemakers on the other side of the International Date Line, is very difficult, and accounts for much of the troubles people have using Windows.

THE FINANCIAL EXPLANATION is that Apple, unlike Microsoft, is and always has been a hardware company. It simply depends on revenue from selling hardware, and cannot exist without it.

THE NOT-SO-CHARITABLE EXPLANATION has to do with Apple's corporate culture, which is rooted in Bay Area Baby Boomdom.

Oh boy, here we go. We've discarded the obvious and sensible answer, to pursue politics. I already got enough of this with the last Presidential election cycle.

Now, since I'm going to talk for a moment about culture, full disclosure is probably in order, to protect myself against allegations of conflict of interest and ethical turpitude: (1) Geographically I am a Seattleite,

> of a Saturnine temperament, and inclined to take a sour view of the Dionysian Bay Area, just as they tend to be annoyed and appalled by us. (2) Chronologically I am a post-Baby Boomer. I feel that way, at least, because I never experienced the fun and exciting parts of the whole Boomer scene--just spent a lot of time dutifully chuckling at Boomers' maddeningly pointless anecdotes about just how stoned they got on various occasions, and politely fielding their assertions about how great their music was. But even from this remove it was possible to glean certain patterns, and one that recurred as regularly as an urban legend was the one about how someone would move into a commune populated by sandal-wearing, peace-sign flashing flower children, and eventually discover that, underneath this facade, the guys who ran it were actually control freaks; and that, as living in a commune, where much lip service was paid to ideals of peace, love and harmony, had deprived them of normal, socially approved outlets for their control-freakdom, it tended to come out in other, invariably more sinister, ways.
>
> Applying this to the case of Apple Computer will be left as an exercise for the reader, and not a very difficult exercise.
>
> It is a bit unsettling, at first, to think of Apple as a control freak, because it is completely at odds with their corporate image. Weren't these the guys who aired the famous Super Bowl ads showing suited, blindfolded executives marching like lemmings off a cliff? Isn't this the company that even now runs ads picturing the Dalai Lama (except in Hong Kong) and Einstein and other offbeat rebels?

You gotta love advertising. While this campaign is at least as tacky as the Microsoft ads featuring office workers flying around on wires, an observer should be reminded of one big difference: The products that Apple rolls out on the red carpet represent *actual innovation*.

I don't consider the Tablet PC to be in the same league, either. The Tablet PC is essentially a big fat Apple Newton for the modern age, allowing a whole new generation of modern programs to be functionally inaccessible to their hapless user.

> It is indeed the same company, and the fact that they have been able to plant this image of themselves as creative and rebellious free-thinkers in the minds of so many intelligent and media-hardened skeptics really gives one pause. It is testimony to the insidious power of expensive slick ad campaigns and, perhaps, to a certain amount of wishful thinking in the minds of people who fall for them. It also raises the question of why Microsoft is so bad at PR, when the history of Apple demonstrates that, by writing large checks to good ad agencies, you can plant a corporate image in the minds of intelligent people that is completely at odds with reality.

Yes, slick ad campaigns are the enemy of truth. But Apple owners are not fervent because they've been pumped up by an advertising blitz, they're fervent because the product itself has actually enhanced their life and work. They're not dazzled before they buy it, especially in this age of "switchers", they become happy afterwards.

> (The answer, for people who don't like Damoclean questions, is that since Microsoft has won the hearts and minds of the silent majority--the bourgeoisie--they don't give a damn about having a slick image, any more then Dick Nixon did. "I want to believe,"--the mantra that Fox Mulder has pinned to his office wall in The X-Files--applies in different ways to these two companies; Mac partisans want to believe in the image of Apple purveyed in those ads, and in the notion that Macs are somehow fundamentally different from other computers, while Windows people want to believe that they are getting something for their money, engaging in a respectable business transaction).

Not just any business transaction, but a *cost-effective* one. That immediately removed Apple from the list, since their hardware was so much more expensive up front, and (at the time) didn't interact well with other computing platforms.

> In any event, as of 1987, both MacOS and Windows were out on the market, running on hardware platforms that were radically different from each other--not only in the sense that MacOS used Motorola CPU chips while Windows used Intel, but in the sense--then overlooked, but in the long run, vastly more

significant--that the Apple hardware business was a rigid monopoly and the Windows side was a churning free-for-all.

But the full ramifications of this did not become clear until very recently--in fact, they are still unfolding, in remarkably strange ways, as I'll explain when we get to Linux. The upshot is that millions of people got accustomed to using GUIs in one form or another. By doing so, they made Apple/Microsoft a lot of money. The fortunes of many people have become bound up with the ability of these companies to continue selling products whose salability is very much open to question.

HONEY-POT, TAR-PIT, WHATEVER

When Gates and Allen invented the idea of selling software, they ran into criticism from both hackers and sober-sided businesspeople. Hackers understood that software was just information, and objected to the idea of selling it. These objections were partly moral. The hackers were coming out of the scientific and academic world where it is imperative to make the results of one's work freely available to the public. They were also partly practical; how can you sell something that can be easily copied? Businesspeople, who are polar opposites of hackers in so many ways, had objections of their own. Accustomed to selling toasters and insurance policies, they naturally had a difficult time understanding how a long collection of ones and zeroes could constitute a salable product.

Obviously Microsoft prevailed over these objections, and so did Apple. But the objections still exist. The most hackerish of all the hackers, the Ur-hacker as it were, was and is Richard Stallman, who became so annoyed with the evil practice of selling software that, in 1984 (the same year that the Macintosh went on sale) he went off and founded something called the Free Software Foundation, which commenced work on something called GNU. Gnu is an acronym for Gnu's Not Unix, but this is a joke in more ways than one, because GNU most certainly IS Unix,. Because of trademark concerns ("Unix" is trademarked by AT&T) they simply could not claim that it was Unix, and so, just to be extra safe, they claimed that it wasn't. Notwithstanding the incomparable talent and drive possessed by Mr. Stallman and other GNU adherents, their project to build a free Unix to compete against Microsoft and Apple's OSes was a little bit like trying to dig a subway system with a teaspoon. Until, that is, the advent of Linux, which I will get to later.

But the basic idea of re-creating an operating system from scratch was perfectly sound and completely doable. It has been done many times. It is inherent in the very nature of operating systems.

Operating systems are not strictly necessary. There is no reason why a sufficiently dedicated coder could not start from nothing with every project and write fresh code to handle such basic, low-level operations as controlling the read/write heads on the disk drives and lighting up pixels on the screen. The very first computers had to be programmed in this way. But since nearly every program needs to carry out those same basic operations, this approach would lead to vast duplication of effort.

Nothing is more disagreeable to the hacker than duplication of effort. The first and most important mental habit that people develop when they learn how to write computer programs is to generalize, generalize, generalize. To make their code as modular and flexible as possible, breaking large problems down into small subroutines that can be used over and over again in different contexts. Consequently, the development of operating systems, despite being technically unnecessary, was inevitable. Because at its heart, an operating system is nothing more than a library containing the most commonly used code, written once (and hopefully written well) and then made available to every coder who needs it.

So a proprietary, closed, secret operating system is a contradiction in terms. It goes against the whole point of having an operating system. And it is impossible to keep them secret anyway. The source code--the original lines of text written by the programmers--can be kept secret. But an OS as a whole is a collection of small subroutines that do very specific, very clearly defined jobs. Exactly what those subroutines do has to be made public, quite explicitly and exactly, or else the OS is completely useless to programmers; they can't make use of those subroutines if they don't have a complete and perfect understanding of what the subroutines do.

The only thing that isn't made public is exactly how the subroutines do what they do. But once you know what a subroutine does, it's generally quite easy (if you are a hacker) to write one of your own that does exactly the same thing. It might take a while, and it is tedious and unrewarding, but in most cases it's not really hard.

What's hard, in hacking as in fiction, is not writing; it's deciding what to write. And the vendors of commercial OSes have already decided, and published their decisions.

This has been generally understood for a long time. MS-DOS was duplicated, functionally, by a rival product, written from scratch, called ProDOS, that did all of the same things in pretty much the same way. In other words, another company was able to write code that did all of the same things as MS-DOS and sell it at a profit. If you are using the Linux OS, you can get a free program called WINE which is a windows emulator; that is, you can open up a window on your desktop that runs windows programs. It means that a completely functional Windows OS has been recreated inside of Unix, like a ship in a bottle. And Unix itself, which is vastly more sophisticated than MS-DOS, has been built up from scratch many times over. Versions of it are sold by Sun, Hewlett-Packard, AT&T, Silicon Graphics, IBM, and others.

People have, in other words, been re-writing basic OS code for so long that all of the technology that constituted an "operating system" in the traditional (pre-GUI) sense of that phrase is now so cheap and common that it's literally free. Not only could Gates and Allen not sell MS-DOS today, they could not even give it away, because much more powerful OSes are already being given away. Even the original Windows (which was the only windows until 1995) has become worthless, in that there is no point in owning something that can be emulated inside of Linux--which is, itself, free.

In this way the OS business is very different from, say, the car business. Even an old rundown car has some value. You can use it for making runs to the dump, or strip it for parts. It is the fate of manufactured goods to slowly and gently depreciate as they get old and have to compete against more modern products.

But it is the fate of operating systems to become free.

Microsoft is a great software applications company. Applications--such as Microsoft Word--are an area where innovation brings real, direct, tangible benefits to users. The innovations might be new technology straight from the research department, or they might be in the category of bells and whistles, but in any event they are frequently useful and they seem to make users happy. And Microsoft is in the process of becoming a great research company. But Microsoft is not such a great operating systems company. And this is not necessarily because their operating systems are all that bad from a purely technological standpoint. Microsoft's OSes do have their problems, sure, but they are vastly better than they used to be, and they are adequate for most people.

Why, then, do I say that Microsoft is not such a great operating systems company? Because the very nature of operating systems is such that it is senseless for them to be developed and owned by a specific company. It's a thankless job to begin with. Applications create possibilities for millions of credulous users, whereas OSes impose limitations on thousands of grumpy coders, and so OS-makers will forever be on the shit-list of anyone who counts for anything in the high-tech world. Applications get used by people whose big problem is understanding all of their features, whereas OSes get hacked by coders who are annoyed by their limitations. The OS business has been good to Microsoft only insofar as it has given them the money they needed to launch a really good applications software business and to hire a lot of smart researchers. Now it really ought to be jettisoned, like a spent booster stage from a rocket. The big question is whether Microsoft is capable of doing this. Or is it addicted to OS sales in the same way as Apple is to selling hardware?

Keep in mind that Apple's ability to monopolize its own hardware supply was once cited, by learned observers, as a great advantage over Microsoft. At the time, it seemed to place them in a much stronger position. In the end, it nearly killed them, and may kill them yet. The problem, for Apple, was that most of the world's computer users ended up owning cheaper hardware. But cheap hardware couldn't run MacOS, and so these people switched to Windows.

> Replace "hardware" with "operating systems," and "Apple" with "Microsoft" and you can see the same thing about to happen all over again. Microsoft dominates the OS market, which makes them money and seems like a great idea for now. But cheaper and better OSes are available, and they are growingly popular in parts of the world that are not so saturated with computers as the US. Ten years from now, most of the world's computer users may end up owning these cheaper OSes. But these OSes do not, for the time being, run any Microsoft applications, and so these people will use something else.

This situation may yet come to pass, but from a different angle. Any shift in OS preference in the desktop computer market is already dwarfed by the seismic shifting of computing tasks onto *other devices*, like cellphones, PDAs, media players, in-dash consoles, game systems, video recorders, et cetera, ... all interconnected. The desktop computer remains the ideal centerpiece for a lot of this interaction, but is increasingly non-essential for it to occur.

Microsoft knows this, and has been trying to head it off at the pass for a decade or more. They developed the XBOX as a crowbar for the game-console market. They developed Windows CE as a lanyard into the PDA market. The Tablet PC seemed to pop in from nowhere. It's evident to the casual observer that Microsoft knows it's in a shrinking space, and is trying hard to shimmy out.

Their more successful moves have been to diversify directly, by purchasing other business models, not just other OS markets. MSNBC for example. Now they need to open a chain of hot-dog stands.

> To put it more directly: every time someone decides to use a non-Microsoft OS, Microsoft's OS division, obviously, loses a customer. But, as things stand now, Microsoft's applications division loses a customer too. This is not such a big deal as long as almost everyone uses Microsoft OSes. But as soon as Windows' market share begins to slip, the math starts to look pretty dismal for the people in Redmond.
>
> This argument could be countered by saying that Microsoft could simply re-compile its applications to run under other OSes. But this strategy goes against most normal corporate instincts. Again the case of Apple is instructive. When things started to go south for Apple, they should have ported their OS to cheap PC hardware. But they didn't. Instead, they tried to make the most of their brilliant hardware, adding new features and expanding the product line. But this only had the effect of making their OS more dependent on these special hardware features, which made it worse for them in the end.

This is based on a common misunderstanding that Apple and Microsoft are in the same business. As I state several times in these notes, Apple and Microsoft are not. Comparing them is like comparing a restaurant supply store to a restaurant, or a muffler shop to a limo service. Their markets only intersect, and a good move for one might be disaster for the other.

Why would Apple want to switch from making $100 off the sale of a computer, to $10 off the sale of an OS? Their market- and mind-share would have to instantly increase by *ten times* just to *break even* on that move. Linux is downloadable for free -- why would any company deliberately compete with that? Even Microsoft is bailing out into other markets, as fast as it can.

> Likewise, when Microsoft's position in the OS world is threatened, their corporate instincts will tell them to pile more new features into their operating systems, and then re-jigger their software applications to exploit those special features. But this will only have the effect of making their applications dependent on an OS with declining market share, and make it worse for them in the end.
>
> The operating system market is a death-trap, a tar-pit, a slough of despond. There are only two reasons to invest in Apple and Microsoft. (1) each of these companies is in what we would call a co-dependency relationship with their customers. The customers Want To Believe, and Apple and Microsoft know how to give them what they want. (2) each company works very hard to add new features to their OSes, which works to secure customer loyalty, at least for a little while.
>
> Accordingly, most of the remainder of this essay will be about those two topics.

THE TECHNOSPHERE

are maddeningly unstable, and that there's no telling what weird, seemingly inconsequential event might cause the system to shift into a radically different configuration.

Like say, the release of the iPod.

To put it another way, Microsoft can be confident that Thomas Penfield Jackson will not hand down an order that the brains of everyone in the developed world are to be summarily re-programmed. But there's no way to predict when people will decide, en masse, to re-program their own brains. This might explain some of Microsoft's behavior, such as their policy of keeping eerily large reserves of cash sitting around, and the extreme anxiety that they display whenever something like Java comes along.

I have never seen the inside of the building at Microsoft where the top executives hang out, but I have this fantasy that in the hallways, at regular intervals, big red alarm boxes are bolted to the wall. Each contains a large red button protected by a windowpane. A metal hammer dangles on a chain next to it. Above is a big sign reading: IN THE EVENT OF A CRASH IN MARKET SHARE, BREAK GLASS.

What happens when someone shatters the glass and hits the button, I don't know, but it sure would be interesting to find out. One imagines banks collapsing all over the world as Microsoft withdraws its cash reserves, and shrink-wrapped pallet-loads of hundred-dollar bills dropping from the skies. No doubt, Microsoft has a plan. But what I would really like to know is whether, at some level, their programmers might heave a big sigh of relief if the burden of writing the One Universal Interface to Everything were suddenly lifted from their shoulders.

THE RIGHT PINKY OF GOD

I'm not going to touch this section, except to refer the interested reader to additional reading material on the formation of the universe.

For example, determining the initial wrinkles of space-time by listening to the "overtones" of the Big Bang.

In his book The Life of the Cosmos, which everyone should read, Lee Smolin gives the best description I've ever read of how our universe emerged from an uncannily precise balancing of different fundamental constants. The mass of the proton, the strength of gravity, the range of the weak nuclear force, and a few dozen other fundamental constants completely determine what sort of universe will emerge from a Big Bang. If these values had been even slightly different, the universe would have been a vast ocean of tepid gas or a hot knot of plasma or some other basically uninteresting thing--a dud, in other words. The only way to get a universe that's not a dud--that has stars, heavy elements, planets, and life--is to get the basic numbers just right. If there were some machine, somewhere, that could spit out universes with randomly chosen values for their fundamental constants, then for every universe like ours it would produce 10^229 duds.

Though I haven't sat down and run the numbers on it, to me this seems comparable to the probability of making a Unix computer do something useful by logging into a tty and typing in command lines when you have forgotten all of the little options and keywords. Every time your right pinky slams that ENTER key, you are making another try. In some cases the operating system does nothing. In other cases it wipes out all of your files. In most cases it just gives you an error message. In other words, you get many duds. But sometimes, if you have it all just right, the computer grinds away for a while and then produces something like emacs. It actually generates complexity, which is Smolin's criterion for interestingness.

Not only that, but it's beginning to look as if, once you get below a certain size--way below the level of quarks, down into the realm of string theory--the universe can't be described very well by physics as it has been practiced since the days of Newton. If you look at a small enough scale, you see processes that look almost computational in nature.

I think that the message is very clear here: somewhere outside of and beyond our universe is an operating system, coded up over incalculable spans of time by some kind of hacker-demiurge. The cosmic operating system uses a command-line interface. It runs on something like a teletype, with lots of noise and heat; punched-out bits flutter down into its hopper like drifting stars. The demiurge sits at his

teletype, pounding out one command line after another, specifying the values of fundamental constants of physics:

universe -G 6.672e-11 -e 1.602e-19 -h 6.626e-34 -protonmass 1.673e-27....

and when he's finished typing out the command line, his right pinky hesitates above the ENTER key for an aeon or two, wondering what's going to happen; then down it comes--and the WHACK you hear is another Big Bang.

Now THAT is a cool operating system, and if such a thing were actually made available on the Internet (for free, of course) every hacker in the world would download it right away and then stay up all night long messing with it, spitting out universes right and left. Most of them would be pretty dull universes but some of them would be simply amazing. Because what those hackers would be aiming for would be much more ambitious than a universe that had a few stars and galaxies in it. Any run-of-the-mill hacker would be able to do that. No, the way to gain a towering reputation on the Internet would be to get so good at tweaking your command line that your universes would spontaneously develop life. And once the way to do that became common knowledge, those hackers would move on, trying to make their universes develop the right kind of life, trying to find the one change in the Nth decimal place of some physical constant that would give us an Earth in which, say, Hitler had been accepted into art school after all, and had ended up his days as a street artist with cranky political opinions.

Even if that fantasy came true, though, most users (including myself, on certain days) wouldn't want to bother learning to use all of those arcane commands, and struggling with all of the failures; a few dud universes can really clutter up your basement. After we'd spent a while pounding out command lines and hitting that ENTER key and spawning dull, failed universes, we would start to long for an OS that would go all the way to the opposite extreme: an OS that had the power to do everything--to live our life for us. In this OS, all of the possible decisions we could ever want to make would have been anticipated by clever programmers, and condensed into a series of dialog boxes. By clicking on radio buttons we could choose from among mutually exclusive choices (HETEROSEXUAL/HOMOSEXUAL). Columns of check boxes would enable us to select the things that we wanted in our life (GET MARRIED/WRITE GREAT AMERICAN NOVEL) and for more complicated options we could fill in little text boxes (NUMBER OF DAUGHTERS: NUMBER OF SONS:).

Even this user interface would begin to look awfully complicated after a while, with so many choices, and so many hidden interactions between choices. It could become damn near unmanageable--the blinking twelve problem all over again. The people who brought us this operating system would have to provide templates and wizards, giving us a few default lives that we could use as starting places for designing our own. Chances are that these default lives would actually look pretty damn good to most people, good enough, anyway, that they'd be reluctant to tear them open and mess around with them for fear of making them worse. So after a few releases the software would begin to look even simpler: you would boot it up and it would present you with a dialog box with a single large button in the middle labeled: LIVE. Once you had clicked that button, your life would begin. If anything got out of whack, or failed to meet your expectations, you could complain about it to Microsoft's Customer Support Department. If you got a flack on the line, he or she would tell you that your life was actually fine, that there was not a thing wrong with it, and in any event it would be a lot better after the next upgrade was rolled out. But if you persisted, and identified yourself as Advanced, you might get through to an actual engineer.

What would the engineer say, after you had explained your problem, and enumerated all of the dissatisfactions in your life? He would probably tell you that life is a very hard and complicated thing; that no interface can change that; that anyone who believes otherwise is a sucker; and that if you don't like having choices made for you, you should start making your own.

Only computer hackers who have delved into the inconvenient guts of an operating system that turns their home computer into a giant pile of tinker-toys are truly liberated from the cruel manipulation of their slipshod corporate taskmasters. You know what I think makes an operating system truly great? Its ability to help me GET MY WORK DONE.