Emulation

Bridges over troubled waters

Matt Barton

The term *emulation* means to either equal or exceed something or someone else. As computer jargon, however, *emulation* means recreating another computer or console's operating system on another system; e.g., recreating a Nintendo Entertainment System on your Sega Dreamcast so you can boot up a *Super Metroid* ROM, or playing classic arcade games like *Ms. Pac-Man* or *Omega Race* on your Gameboy Advance SP. Certainly, neither Nintendo nor Sega ever meant for their systems to be used for such purposes.

The obvious utility of emulation software is that it allows users better compatibility and more freedom. The obvious threat of the software is that it allows users better compatibility and more freedom.

This article is not a "how-to" or a guide for emulation. If you want to learn how to emulate a NES on your Dreamcast, there are better sources of information. What I'm concerned with here is why we, as promoters of free software and concerned citizens in general, should care about these issues. As I will endeavour to show, emulation is not just about playing the latest PS2 title on your PC or even allowing new gamers access to older titles. It's about the freedom to run the software you want on the hardware you want.

The threat of emulation

As we discover so often in the computer industry, emulation involves a clash between private and public interests. It is in the best interest of most software and hardware corporations to "lock-in" users. If you want to play a Nintendo game, Nintendo wants you to buy a Nintendo Entertainment System. If you want to play *Metroid* on your GBA, Nintendo expects you to pony up $20 for the "authorized" version they're selling for the GBA even if you bought and still own the original cartridge for your NES. (It is quite possible to emulate the original on your GBA.) If you want to run Amiga Software, you'll need an Amiga, and a company named Cloanto is ready to sue you if you don't pay them for the privilege of emulating the system on your PC. Microsoft's strategies to shut out GNU/Linux users or impose Internet Explorer on web surfers are well known. Emulation represents a threat to corporations who thrive on the exclusivity of their software and hardware.

What's the big deal? Who cares if you're not free to emulate other systems on your PC or console? Probably the most obvious reason I care is convenience. As someone who enjoys researching game and software history, it's neither convenient nor economically viable for me to own all the hardware I need to run more than a few systems. To use an analogy that Larry Lessig uses for a different purpose, imagine if the highway system were built so that only a particular make of car could drive on a particular kind of road. With such a system in place, you'd need three different automobiles to get from California to Florida and two more to get to New York. If one car could "emulate" the rest and get people around without the need for separate vehicles, everyone would want one no matter the "threat" such a car would pose to the other manufacturers. Sure, perhaps

a Toyota would perform better than a Mazda on a "Toyota Highway", but as long as I can emulate my way to Bourbon Street in my Miata, I'll deal with it.

Emulation has been with computers since the dawn of computing, and it certainly hasn't always carried the stigma it does today. In 1965, IBM released its new IBM 360 line. The only problem was that IBM's customers would rather live without the bells and whistles of the 360s if they could continue using the same software they'd paid so much to develop and spent so much time getting familiar with. IBM anticipated their fears and made the switch a no-brainer. Along with their 360s came an emulation program by Larry Moss (http://www.zophar.net/articles/art_14-2.html) that allowed users to run software written for IBM's older 7070 systems. IBM realized quite rightly that users would be reluctant to upgrade their computers if it meant reprogramming or purchasing a whole new suite of customized software. Moss's emulator made switching less costly, and was thus quite popular with IBM's clientele. Moss's emulator was IBM's trump card, and they won big with it.

If every emulator was as helpful to a company's bottom line as Moss's was for IBM, there would be no "threat" of emulation. Every corporation would welcome them and probably develop emulators themselves. As long as emulation software is programmed "in house" and kept under corporate control, it's a jewel in their crown. The problem is that it is just as easy for a competitor to take advantage of emulation software, which can become a powerful weapon indeed if a company is relying on the exclusivity of its software to maintain its market share. This was certainly the case with Colecovision's "Atari 2600 adapter" released in 1982, which gave owners of Colecovisions the freedom to select from two formerly competing libraries of games. There's no good reason to buy one of Atari's game consoles if you can play all the Atari games on your Colecovision—*and* play those near pixel-perfect Colecovision arcade conversions you've been hearing so much about.

There is also a concern that emulation may be bad for current hardware; ensuring backwards-compatibility may limit a system or drive up manufacturing costs. Commodore made no serious effort to promote C-64 emulation when it released its newer Amiga line and didn't even include a $5\frac{1}{4}$" floppy on any of its new machines—a decidedly different strategy than it had taken with the C-128, which could switch between three different modes: C-128, C-64, and C/PM. Sony's decision to incorporate backwards-compatibility with the PS1 in its PS2 was seen by some as a concession to consumers, but by others as a rather meek strategy to buff up the game library for their new console (a few hundred PS1 games looks a lot better than 3 or 4 rushed PS2 titles on release day).

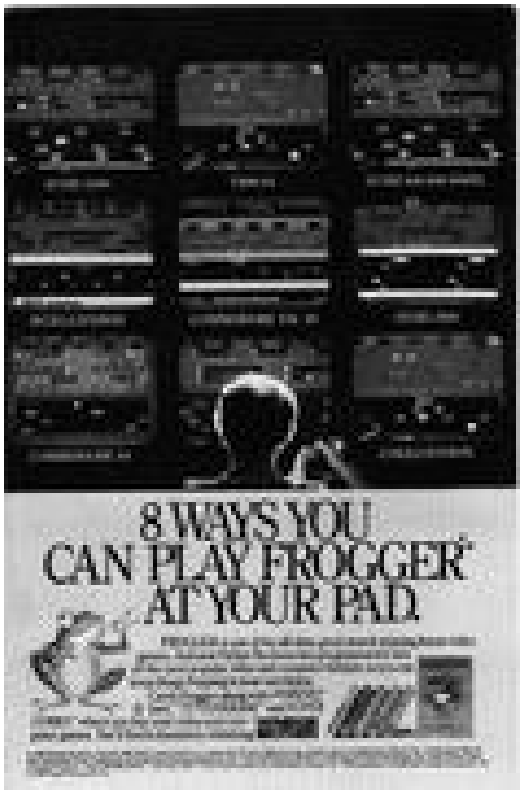
Another economic issue is that users may not want to buy new accessories like joysticks or mice or new versions of old software if they can use the ones they purchased for another system. Console manufacturers want to do everything they can to increase "customer loyalty" by forcing people to purchase proprietary accessories that will only work with their systems. Those of us who can still remember unplugging our Atari 2600 joysticks and plugging them into our Commodore 64s (and later our Amigas) are just too spoiled to tolerate the dozens of incompatible plugs and ports on later consoles.

Yet we'll still buy a console if it has "the game" we can't live without. Console manufacturers and operating system developers know that if a very popular software program is only available for one operating system—the "killer app"—then consumers will purchase it regardless of the sophistication of another machine. No one knows this better than Nintendo, who have milked their various "franchises" to the point of absurdity (what's next, *Yoshi's Love Match Timber Sports?*). Consumers will also be drawn to computing platforms with the most software. In the 80s, consumers were often bewildered by the dozens of incompatible computer systems on the market. Computers like the Commodore 64, Tandy TRS-80, and Apple II competed savagely with each other and divided consumers, who often became fiercely loyal to whatever brand they purchased. Companies who dared to manufacture "clones" of these proprietary machines, like Franklin, were taken to court and driven out of town. The money was assumed to be in proprietary hardware—an assumption which Microsoft wisely did not share. The war continued into the "16-bit" era, and it is still easy to find diehards still clinging to their proprietary Amiga or Atari computers despite their alleged obsolescence. When you're asked to learn another operating system from scratch, it's amazing how superior your old one feels.

Game consoles underwent a similar turf war in the 80s. Seemingly every major department store was offering its own "television game". Few families could afford more

than one system, so the games that children played were limited by whether their living room was home to a Colecovision, Intellivision, Atari 2600, Adventurevision, Astrocade, or Odyssey II—to name a few. Each system had its special features, quirks, and a game library that distinguished it from the competition. Unlike the personal computer market, the hardware manufacturers produced nearly all of the software "in house" or under exclusive agreements and even tried to hide the identity of their programmers to protect against competition from other console manufacturers. When four programmers (including David Crane, later developer of *Pitfall*) left Atari to begin producing unauthorized games for the 2600 in 1979, Atari promptly sued to protect its monopoly (a strategy that would be later repeated by Nintendo against Atari, who'd try the same thing). The stakes were high and Atari was too short-sighted to see the advantage of third-party developers in helping it to sell consoles. Atari eventually lost in 1982, the same year Colecovision released its "Atari 2600 Adapter", which allowed Colecovision owners to play any of the Atari 2600's titles. Atari sued and lost yet again. 1982 was a year when consumer interests mattered to our elected officials, but in 1983 came the great videogame crash. So it goes.

80s *Frogger* ad showing screenshots from 8 systems. From Tomorrow's Heroes website



Whenever there is an abundance of platforms, developers have to port their programs to a wide variety of machines if they want to make the most profit. A glance at the ads in many comic books of the 80s reveals a montage of screenshots of the same game across systems. Though these ads are intended to sell games rather than game systems, it is clear enough which systems have the best graphics. Thus, hardware manufacturers could no longer offer inferior products in the hopes that their exclusive software line up would make the difference. Now they had to court not only consumers but developers—if they could not force developers to release a title only for their system, then perhaps they could coax them with more sophisticated hardware. In a trend that continues to this day, the popularity of any given game console relies at least as much on its third-party developers than the sophistication of its hardware. Software developers want to reach the widest number of consumers; hardware developers want to limit the availability of software to their own systems.

The three parties involved: hardware developers, software developers, and users; each with their own interests, most of which conflict with one or the other. It's fun to think about the changes that would come about if any one of these groups dictated the terms. Hardware developers want to make software exclusive to their products so that users who want the software must invest in their hardware. They make their products as incompatible with the competition as they can. Software developers want to reach the maximum number of consumers with the least amount of compatibility issues, so they push towards generic *hardware* standards while closing up *software* standards. Meanwhile, users want open hardware and software, since the market rewards consumers when the maximum number of competitors is allowed to participate—a situation that arises in periods when monopolies do not exist. Ultimately, a user doesn't so much want an "Apple"; a user wants a certain set of hardware and software features that are "Apple's" only by virtue of patents, copyrights, and trademarks. A consumer would like to buy a trackball or joystick without having to worry about whether it'll work on a particular machine.

If the world of software development was truly autonomous from the world of hardware development, emulation would

not be an issue. The reason it remains a problem is that hardware manufacturers (and the developers of non-free operating systems like *Windows* or *Tiger*) have proven themselves willing to enter into lucrative licensing agreements with the makers of popular games and applications time and time again. Sony and Nintendo fought hard for the *Final Fantasy* series, and it's not unthinkable to suggest that Square's decision to go with Sony ended Nintendo's stranglehold on the console market of the 90s. Similar examples from personal computing are also plentiful. Clearly the Commodore Amiga would have died off years before if NewTek's *Video Toaster* was also available for IBM-compatibles or Apple Macintoshes. Apple defended its graphical operating system fiercely, even suing Microsoft for daring to call its "Trashcan" feature by the same name. Microsoft, the unabashed king of operating system lock-in, helped to destroy hardware vendor lock-in by convincing IBM to use Microsoft's operating system on their computers instead of developing their own. Once Microsoft had its foot in the door, it was all too easy to encourage clone-makers and third-party developers to swear their fealty to the new standard. The software industry was now wagging the dog, and IBM's OS/2 failed to reverse the situation. Clones running DOS and later Windows won out because a platform that can do all things adequately trumps any other that can do fewer things better.

It's this last fact that brings us to the threat emulation software poses to the software industry. Compared to the threat emulation poses to their livelihoods, unauthorized copying and distribution are the flies distracting them from the lion. If I can successfully emulate a propriety operating environment on a free machine, then there is no reason to own the proprietary system, which will always be more limited than the superior one capable of emulating it. If I can play Nintendo DS games on my Sony PSP, but not vice versa, then the choice is obvious which system I should purchase. If a generic portable like the GP32 (http://en.wikipedia.org/wiki/GP32) can emulate both, I won't even bother with the proprietary models. Likewise, if my powerful PC can emulate all old and modern consoles flawlessly (or even improve on their performance), then I would be foolish to invest in a game console—to paraphrase Will Shatner's old slogan for Commodore, "Why buy a system when you can have an emulator?". We shouldn't wonder why Nintendo has taken to experimenting with control

Commodore Vic-20 ad featuring Will Shatner.

and interface and introducing bizarre "innovations" in these areas: they are worried about emulation.

Is Nintendo right to worry? Perhaps. In 1999, an emulator for the 3-year old Nintendo 64 was released—UltraHLE. UltraHLE demonstrated that it was possible to successfully emulate *modern* consoles at a playable frame rate on reasonably equipped PCs. Emulators would soon follow for other modern consoles. To put it crassly, the emulation scene "hit the fan" when UltraHLE debuted and lost whatever innocence it had hitherto purported to possess. Emulation software could now be used to copy and distribute games currently on the market—it wasn't just about obsolete systems anymore.

Many emulation enthusiasts have long claimed that they are not interested in "piracy" or promoting the unauthorized distribution of commercially viable programs. A typical enthusiast might claim that, though technically illegal, it is ethically justifiable to swap old titles that are no longer being sold commercially, but doing the same with modern games is wrong. If I can't buy *Gorf*, then what's wrong with downloading the ROM? Such an attitude is untenable. We cannot at once recognize software as both private (legitimizing pro-

prietary software) and public property (legitimizing "abandonware"). The law makes no distinction between copying and distributing old or new software, since no software is in the public domain unless its copyright owners officially have officially declared it so.

Nevertheless, the economic threat emulation poses to modern consoles seems negligible. A PC powerful enough to emulate any of the "Big 3" would be exponentially more expensive than simply buying the console. As long as there is a significant price gap between the cost of a given console and a machine with enough power to emulate it, the threat is marginal. Ostensibly, someone with thousands to spend on computer hardware would view modern console emulation more as a curiosity than a practical replacement for his game console. We talk of "hundreds" spent on console gaming, but "thousands" spent on computer gaming. There is no threat here.

The challenge of emulation

Let us hubristically assume for the moment that a free operating system had been built that could, for a reasonable investment in readily available hardware, not only emulate Microsoft *Windows* but actually run it faster and more efficiently than is currently possible. If it is possible today to emulate a Commodore 64 or Amiga at exponentially faster speeds than those computers were capable of reaching, it seems possible to do the same for the modern PC on a significantly more advanced machine. I say "hubristically", because of course such a feat would entail some fundamental discovery of computer engineering far more exciting than the mere execution of a Microsoft application. Emulating another system is difficult because it requires a system to not only support its own environment but also recreate another within it. There is an order of magnitude difference here in terms of complexity. In short, there is an exponential relation between the complexity of system X and system Y's ability to emulate it.

Consider a simple example. Say that you've been chosen to play the role of a famous surgeon in an upcoming documentary. To prepare adequately for the role, you'll need to not only learn your lines, but study the surgeon's actual techniques and thus offer a convincing performance. You can imagine being asked to spend several days observing surgeons at the local hospital, and then spending time in front of a mirror attempting to "get it right". Anyone who has spent any time with professional actors knows how much hard work goes into preparing for a serious role. We might compare this type of acting to "simulation", or a program known as a "port". It's not *really* the software you wanted, but it fools you into thinking it is as long as you don't peer under the hood.

However, you haven't been asked to merely *act* like a surgeon, but to actually *be* one. In the film you will actually perform a real surgery—but after it's all over, you can go back to being an actor and prepare for your next film, perhaps one in which you will not only play a nuclear engineer but *be* one. No human actor could possibly undertake such tasks—by the time they'd been through medical school and all the intermediate steps required to be a surgeon, they'd be too old and otherwise preoccupied to worry about a silly film!

While not perfect, this example does give some idea of what a system is asked to do when attempting to emulate other systems. We're asking our operating system to not only be itself, but also, at the same time, to be something else. With this in mind, it's easy to see why the "emulation scene" is mostly concerned with antiquated consoles and computers and why it takes a powerhouse of a PC to emulate even systems like the Amiga.

Emulation programs for antiquated systems are available for both computers and consoles. Perhaps more importantly, the "ROMs" of games for these systems are widely available on the internet. "ROM" is the "street name", if you will, for a "ROM Image Definition". Obviously, you can't just plug an NES cartridge into your computer (another advantage of the cartridge system for the game industry). However, hackers have long ago found ways to stream the data from a cartridge and store it as a binary file called a "ROM image". This ROM can then be loaded into an emulator. Of course, it didn't take long for these ROMs to start showing up online, where emulation enthusiasts can download them for free.

Andrew Wolan, manager of the popular emulation website *EmulationZone.org*, argues that the internet has "helped speed the [emulation] process by collaborating efforts worldwide", bringing emulation from the workshops of a few hardcore hackers to the mainstream. Sega's defunct Dreamcast system is home to a thriving emulation scene, as are most modern consoles (even the portables). GNU/Linux

GP32 portable gaming device. Pic from Wikipedia.

is home to countless emulation packages—indeed, many of the most popular Windows emulators are ports of programs created for GNU/Linux. The Multiple Arcade Machine Emulator, MAME, is available for a wide variety of computers and consoles. MAME is an especially impressive achievement in emulation because arcade machines were often even more "proprietary" than game consoles, because they only needed to run one game. The hardware could thus be built especially to support that one game. Thanks to MAME, those lucky enough to own a powerful PC have access to nearly every arcade game ever created and can emulate classic computer systems at speeds never dreamt of by their designers. Never before in history have so many programs been available for a single platform. How many of us have our hard drives loaded with thousands if not tens of thousands of games for dozens of systems; some of which we will never have the time to try once, much less play through?

The future of emulation and free software

Perhaps the most significant implication of emulation software is that it offers users more platform independence than they have ever enjoyed before. Armies of hackers and hordes of enthusiasts have used the internet to accomplish what the industry would never have given them—the freedom to run the programs they want on the platforms they want. If I want to play *Donkey Kong Country* on my PS2, I can. If I want to run an Amiga desktop on top of XP, I can. The choice is no longer up to hardware and software developers. Eventually, it seems likely that these boundaries will blur to the point where the term "platform" is no longer applicable. Generic devices will run generic software, and the equivalent of the operating system will simply be a "smart" emulator that will automatically configure itself to accommodate a program. Of course, such ideas are nothing new; Java is intended to fill much the same niche, though it has not caught on as well as Sun had hoped.

Perhaps the problem is that we have yet to see a significant "open source" or generic movement in hardware development to equal the achievements of software programmers. The reason for such a lack is not hard to see. While some PC components are generic and easy to find cheap (mice and keyboards, for instance), designing and manufacturing sophisticated hardware like CPUs and graphic cards is a much more expensive process than software development—only major corporations like Intel have the necessary funds. Most of the components in our modern PCs are not fundamentally different from IBM's original model, and modern consoles seem to be moving towards the same architecture. Though many of us are impressed with the gains hardware has made over the decades (particularly in graphics, processor speed, and so on), we are talking about innovations to an existing model rather than the invention of new models. How many refinements will we make to our steam engines before someone discovers the potential of gasoline and catapults the industry into the next age? Indulge me for a moment. The kind of "super computer" that will be capable of the high-end emulation I have in mind is not possible without radical new (and open) hardware. Such a computer would have the power to emulate modern systems like GNU/Linux and Windows while at the same time offering exciting new applications previously impossible. If high-end PCs struggle to reach speeds of 3 gigahertz, this machine will run at terahertz. Furthermore, it will not be tied to any particular manufacturer of hardware or software, but remain open so that manufacturers will compete and the market will work to lower production costs. Eventually this machine's emulation possibilities would destroy the commercial viability of any platform-specific development model and all software would be platform independent by necessity—there would be no platforms!

Such dreams are often derided by cynics, who are always on hand to offer compelling arguments or observations about

why the status quo is here to stay and true change is impossible. George Airy, an important statesman for science during Babbage's day, convinced most of Britain that mechanical devices for computation could *never* excel the speed or accuracy of human calculators working with slide rules and tables. If there is one thing that studying the history of science has taught me, it's that "impossible" simply means we haven't figured out a way to do it yet. Often enough, what holds us back from solving a problem is that we keep asking the wrong questions or making the wrong assumptions. We assume, for instance, that tomorrow's computers will be faster because the CPUs will be faster, not that the very architecture of the PC will be rebuilt or that the thing we now call a "computer" will not be a grey box but built piecemeal into the other devices we'll carry around with us. We need to be listening to revolutionaries on the frontier, not Intel and Microsoft's corporate speak.

Those of us who are eager to not only push the computer industry forward, but to maximize its potential to make citizens of consumers, need to stay focused on emulation. We need to add our concerns about emulation to the attacks we make on tyrannical laws like the DMCA, which make reverse-engineering (a critical aspect of emulation programming) a federal crime. We need to build our free operating systems and open architectures so that they not only run the best of free software, but are also able to emulate whatever the proprietary world has to offer—for once a proprietary system is successfully and cheaply emulated on a free platform, revolution is inevitable.

Copyright information

About the author

Matt Barton is an English professor at St. Cloud State University in Minnesota. He is an advocate of free software, wikis, and the Creative Commons. He also studies and writes about videogames.