Case No. 16-1808

IN THE UNITED STATES COURT OF APPEALS
FOR THE FOURTH CIRCUIT

SAS INSTITUTE, INC.,

Plaintiff-Appellant,

v.

WORLD PROGRAMMING LIMITED,

Defendant-Appellee.

On Appeal from the United States District Court
for the Eastern District of North Carolina at Raleigh
The Honorable Louise W. Flanagan, US District Court Judge
Case No. 5:10-cv-00025-FL

BRIEF OF AMICUS CURIAE ELECTRONIC FRONTIER FOUNDATION IN SUPPORT OF APPELLEE AND AFFIRMANCE

Mitchell L. Stoltz
Kit Walsh
Michael Barclay
ELECTRONIC FRONTIER FOUNDATION
815 Eddy Street
San Francisco, California 94109
(415) 436-9333

Attorneys for Amicus Curiae
ELECTRONIC FRONTIER FOUNDATION

CORPORATE DISCLOSURE STATEMENT

Pursuant to Rule 26.1 of the Federal Rules of Appellate Procedure, *amicus curiae* Electronic Frontier Foundation states that it does not have a parent corporation, and that no publicly held corporation owns 10% or more of the stock of *amicus*.

TABLE OF CONTENTS

TABLE OF AUTHORITIES

Cases

Statutes

Other Authorities

STATEMENT OF IDENTITY AND INTEREST OF AMICUS CURIAE[1]

The Electronic Frontier Foundation (EFF) is a nonprofit civil liberties organization that works to ensure that new technologies advance the freedom of its users rather than compromising it. EFF has over 36,000 dues-paying members, and represents the interests of computer scientists and users of technology who have relied on decades of custom and precedent that recognized the importance of interoperability. That custom and precedent has also permitted software engineers to reimplement the functionality of interfaces like interpreter/compilers and methods of operation embodied in software. The public depends on and expects the inherent openness of interfaces to support innovation by startups and incumbents alike. The Court should decline SAS's invitation to undermine those settled expectations.

INTRODUCTION AND SUMMARY OF ARGUMENT

For decades, computer scientists have relied on the freedom to reimplement software functions to enable rapid innovation in computer technology. For decades, circuit courts have supported that reliance, concluding that Section 102(b) of the Copyright Act protects a programmer's source code as

[1] No counsel for a party authored this brief in whole or in part, and no such counsel or party made a monetary contribution intended to fund the preparation or submission of this brief. No person other than the amicus curiae, or its counsel, made a monetary contribution intended to fund its preparation or submission. Both parties have consented to the filing of this brief.
Websites cited in this brief were last visited on February 15, 2017.

creative expression, but does not cover the processes, systems, and methods of operation that code may employ to interface with other software. The district court correctly followed that precedent and rejected SAS's claim that copyright law grants it a monopoly over the functional input and output behavior of its interpreter/compiler.

If SAS's view had been accepted at the birth of modern computing, many important technologies would never have existed or succeeded. For example, the widespread availability of diverse, cheap, and customizable personal computers owes its existence to the lack of copyright on the specification for IBM's Basic Input/Output System (BIOS) for the PC. And interoperability was essential to many modern computing developments, including those of operating systems such as UNIX, programming languages such as "C," the Internet's network protocols, and cloud computing.

Today, open, uncopyrightable interfaces continue to spur the creation and adoption of new technologies. When programmers can freely reimplement or reverse engineer functional interfaces without obtaining a costly license or risking a lawsuit, they can create compatible software that the interface's original creator might never have envisioned or had the resources to develop. Moreover, compatible software helps enable people to switch platforms and services freely, and to find software that meets their needs regardless of what

browser or operating system they use. Without the compatibility enabled by the open nature of interfaces, customers could be forced to leave their data and programs behind when they switch to a new service.

The freedom to reimplement software functionality also helps developers rescue "orphan" software or data—systems that are no longer supported by their creators. When a company stops supporting a computer platform or service, the ability to freely reimplement interfaces protects the communities that rely on that software. Government entities and non-profits are especially susceptible to the orphan programs problem as they often cannot afford to upgrade and are left using legacy technologies for years or decades.

SAS seeks an unprecedented and dangerous power over the future of innovation. Software creators would have veto rights over any developer who wants to create a compatible program—regardless of whether they copy any literal code from the original implementation. That, in turn, would upset the settled business practices that have enabled the American computer industry to flourish, and choke off many of the system's benefits to customers.

We urge this Court to protect software innovation by affirming the district court's conclusion that WPL did not copy any copyrightable element of SAS's software.

ARGUMENT

I. DECADES OF CONSISTENT COURT OPINIONS SET SETTLED EXPECTATIONS THAT COPYRIGHT LAW ALLOWS FOR INTEROPERABILITY

A. Copyright Law Favors Interoperability by Excluding Functionality from the Scope of Its Limited Monopoly

Decades of progress and innovation in computer technology have depended upon the open nature of functional specifications, such as the functionality of an interpreter/compiler that allows a human being to input "scripts" written in a programming language, and generate particular computer behavior such as rendering a graph, executing computations, or storing data in memory. The free and open use of such specifications and programming languages has been both routine and essential in the computer industry since its beginning, and that use depended, in turn, on the sensible assumption that functional behavior of software is uncopyrightable.

That assumption is deeply grounded in caselaw. SAS makes claims that are remarkably similar to those rejected by the Supreme Court over 100 years ago in *Baker v. Selden*, 101 U.S. 99 (1880). Selden's treatise described several accounting methods, and contained special ledger formats tied to those methods. He claimed that his copyright in the treatise allowed him to exclude others from using those ledger formats and the methods described in the text. *Id.* at 101. The Supreme Court rejected this claim:

> [T]here is a clear distinction between the book, as such, and the art which it is intended to illustrate. To give to the author of the book an exclusive property in the art described therein, when no examination of its novelty has ever been officially made, would be a surprise and a fraud upon the public. That is the province of letters-patent, not of copyright. The claim to an invention or discovery of an art or manufacture must be subjected to the examination of the Patent Office before an exclusive right therein can be obtained; and it can only be secured by a patent from the government.

Id. at 102. The ledger formats were a part of the uncopyrightable system of accounting described in Selden's work. They were not copyrightable expression.

Applying that principle to software, the influential *Lotus* case recognized that menu hierarchies that control functional capabilities are a method of operation, and thus uncopyrightable under 17 U.S.C. § 102(b). *Lotus Dev. Corp. v. Borland Int'l, Inc.*, 49 F.3d 807, 815 (1st Cir. 1995), *aff'd by an equally divided Court*, 516 U.S. 233 (1996); *see also MiTek Holdings, Inc. v. Arce Engineering Co.*, 89 F.3d 1548, 1556-57 (11th Cir. 1996) (input structure of menu was uncopyrightable system under §102(b)). Programmers and developers relied on such rulings for the proposition that functional interfaces, mapping inputs to particular behaviors like the menu hierarchy in *Lotus*, may not be copyrighted under § 102(b).

Similarly, the Ninth Circuit explained that the "functional requirements for compatibility" between computer programs "are not protected by copyright."

Sega Enters., Ltd., v. Accolade, Inc., 977 F.2d 1510, 1522 (9th Cir. 1992); *see also Sony Computer Ent'mt, Inc. v. Connectix Corp.*, 203 F.3d 596, 599–600 (9th Cir. 2000) (describing Sony's PlayStation BIOS as a "system interface procedure[]" that Connectix was entitled to reimplement under § 102(b)). In both cases, competitors reverse engineered the functional input/output profiles of video game software to allow original games to be run on existing video game consoles. The competitors needed to understand what inputs generated corresponding outputs in order to create competing works, and copyright law permitted them to copy these elements.

Courts across the country have followed suit, repeatedly holding that the use of functional elements and formats does not infringe copyright. *E.g., Bateman v. Mnemonics, Inc.*, 79 F.3d 1532, 1547 (11th Cir. 1996) (need for functional compatibility can negate infringement claim); *Gates Rubber Co. v. Bando Chem. Indus., Ltd.,* 9 F.3d 823, 843–44 (10th Cir. 1993) (functionality and processes not copyrightable); *Baystate Technologies, Inc. v. Bentley Systems, Inc.*, 946 F. Supp. 1079, 1087–88 (D. Mass. 1996), *appeal docketed*, Nos. 97–1077, 97–1227 (1st Cir. Jan. 17, 1997, Feb. 28, 1997) (data structures not protected by copyright); *Mitel, Inc. v. Iqtel, Inc.*, 896 F. Supp. 1050 (D. Colo. 1995) (telephone input command codes were an uncopyrightable "procedure, process, system, and method of operation"), *aff'd on different*

grounds, 124 F.3d 1366 (10th Cir. 1997).

B. *Oracle v. Google* Was Wrongly Decided and Threatens Innovation and Competition

The Federal Circuit's decision in *Oracle v. Google*, 750 F.3d 1339 (Fed. Cir. 2014), was a disturbing outlier from modern interoperability precedent and a misstatement of the law of the Ninth Circuit. The Federal Circuit asserted that "an original work - even one that serves a function - is entitled to copyright protection as long as the author had multiple ways to express the underlying idea." *Id.* at 1367. This assertion resurrected reasoning from *Whelan Associates, Inc. v. Jaslow Dental Laboratory, Inc.*, 797 F.2d 1222 (3d Cir. 1986), an early case that was broadly rejected within a decade of its issuance. *Whelan* held that "the purpose or function of a utilitarian work would be the work's idea, and everything that is not necessary to that purpose or function would be part of the expression of the idea." *Id*. at 1236.

The Ninth Circuit itself noted in *Sega* that "[t]he *Whelan* rule . . . has been widely—and soundly—criticized as simplistic and overbroad," 977 F.2d at 1525. The Ninth Circuit implicitly rejected this reasoning again (after *Oracle*) in *Bikram's Yoga College of India, L.P. v. Evolation Yoga, LLC*, 803 F.3d 1032 (9th Cir. 2015).

Noting the industry's long-standing reliance on cases mentioned above, one commenter explained why the Federal Circuit decision was so harmful:

> By ruling that interoperability is relevant only to fair use, and not to protectability, Judge O'Malley would require every developer to perform a fair use analysis before developing an interoperable product. This would place U.S. programmers at a competitive disadvantage to developers in other jurisdictions that recognized that copyright does not protect program elements necessary for interoperability.

Jonathan Band, *The Federal Circuit's Poorly Reasoned Decision in Oracle v. Google*, DisCo (May 9, 2014).[2] Others pointed out flaws in the Federal Circuit's reasoning, suggesting it reflected a "fundamental lack of understanding of how software works." Timothy B. Lee, *The Court That Created the Patent Troll Mess Is Screwing Up Copyright Too*, Vox (May 9, 2014).[3] One computer scientist analogized the computer industry's economy to an engine, and stated that "Oracle is replacing the engine oil with glue." Kin Lane, *Where Will Your API Stand In The Oracle v Google API Copyright Debate?,* API Evangelist (May 10, 2014).[4] Another technologist criticized the Federal Circuit for "creating a new class of copyright plaintiffs intent on using perceived structural similarities in all manner of software products as grounds to demand copyright royalties." John Villasenor, *How Much Copyright Protection Should Source Code Get? A New Court Ruling Reshapes The Landscape*, Forbes (May 19,

---

[2] Available at: http://www.project-disco.org/intellectual-property/050914-the-federal-circuits-poorly-reasoned-decision-in-oracle-v-google.

[3] Available at: http://www.vox.com/2014/5/9/5699960/this-court-decision-is-a-disaster-for-the-software-industry.

[4] Available at: http://apievangelist.com/2014/05/10/where-will-your-api-stand-in-the-oracle-v-google-api-copyright-debate/.

2014).[5]

EFF urges the court to reject SAS's invitation to embrace the Federal Circuit's widely criticized and insupportable approach.

II. INTERFACES, INCLUDING COMPILERS AND INTERPRETERS, PROVIDE BASIC METHODS OF OPERATION AMONG COMPUTERS, PROGRAMS, AND HUMANS

Considered in light of the overwhelming weight of case law, SAS's copyright claim is both insupportable and dangerous. The technology that is the basis of the copyright claim is nothing more than a basic and purely functional interface. Treating it as a form of copyrightable expression would undermine both the letter and the purpose of copyright.

A. The SAS Compiler/Interpreter Is an Interface to Translate Input Values into Specific Outputs

The SAS compiler/interpreter is an interface for humans to operate computers using computer code. A human writes code in the SAS language and then inputs it to the compiler/interpreter, which then processes the code according to a method to generate particular computer behavior.

On the input side, a programmer specifies a certain input format to tell the computer how to read in data. For instance, SAS supports input of calendar dates in different lengths, such as 02/22/2017 versus 02/22/17. The programmer

[5] Available at: http://www.forbes.com/sites/johnvillasenor/2014/05/19/how-much-copyright-protection-should-source-code-get-a-new-court-ruling-reshapes-the-landscape/.

specifies the input format so that the compiler/interpreter correctly interprets the data to be processed. (For example, without the correct input format specification, the computer might interpret "02/22/17" as being in the year 17 A.D., or perhaps 1917, instead of 2017.) The programmer likewise specifies via the SAS Language what the computer should do with the data once it is input. SAS, *Introduction to SAS Informats and Formats,*[6] 2-3.

The SAS compiler/interpreter can then generate a variety of computer behavior. The programmer can command the computer to execute certain mathematical computations, ranging from simple addition to statistical analyses. It can display text and numbers in a variety of formats. For example, commands may be given to display numbers either in raw form, with commas, or with dollar signs (i.e., 2000; 2,000; or $2,000). Such output formats for displaying numerical data have been commonplace for decades in the computer context and centuries in the field of mathematical notation. *Id.* at 4-5. The compiler/interpreter is also capable of rendering graphs describing the input data.

To be clear, the parties agree that WPL *did not copy* the SAS software code making up the compiler/interpreter. Instead, WPL authored its own code to achieve the same functionality, mapping the same input values to the same

[6] Available at: http://support.sas.com/publishing/pubcat/chaps/59498.pdf.

computer operations. Thus, if is "copied" anything, it was purely functional material: ideas, not expression.

B. Copyright Protection for Interfaces Like the SAS Interpreter/Compiler Would Stifle Competition and Innovation

There is a good reason activities like WPL's do not violate copyright: any other rule would hinder copyright's core purpose of promoting creativity and innovation. Software programmers must comply with technological programming restrictions to make sure that programs fed into a compiler/interpreter will work in a known and certain manner. This is not just for convenience. A compiler/interpreter has to understand the *same* functional names and give them the same meaning as another in order to know that programs will operate in the intended way.

If the legal system required that each compiler/interpreter used a different vocabulary because of "design choices," the *functional operations* specified in a software script would not work at all with another compiler/interpreter, and the copyright monopoly would impermissibly extend to that functionality. To a computer, the names *themselves* are methods of operation under § 102(b); the compiler/interpreter uses those inputs to determine what operations the computer shall execute. *See Lotus*, 49 F.3d at 815 ("the Lotus menu command hierarchy is an uncopyrightable 'method of operation'" because it "provides the means by which users control and operate Lotus 1–2–3").

In the SAS input format example above, the computer will get the correct year only if the exact input format is specified and known. The existence of "multiple ways to express" a format (as *Oracle* would put it) is irrelevant—using the wrong format will mean that the program will not understand the correct year, and will not work.

Copyright protection for the functionality of the SAS interpreter/compiler would also impermissibly lock in the universe of computer programs written by end users in SAS language. Once written, those programs can only operate in different systems if the compiler/interpreter functionality is the same; if not, there are huge quantities of programs that are either locked in to the original platform, or are useless. As Judge Boudin put it, "it is hard to see why customers who have learned the Lotus menu and devised macros for it should remain captives of Lotus because of an investment in learning made by the users and not by Lotus." *Lotus*, 49 F.3d at 821 (Boudin, J., concurring). SAS does not own its customers' programs, and should not be allowed to obtain quasi-ownership in them under the guise of copyright protection in the compiler/interpreter needed for those programs to function. Such end user programs are widespread in many of the following examples of functional interfaces.

III. RE-IMPLEMENTATION OF INTERFACES WAS ESSENTIAL TO THE DEVELOPMENT OF MODERN COMPUTERS AND THE INTERNET

The development of modern computing relied upon input/output functionality being free of copyright restrictions. The ability to re-implement such functional interfaces has directly led to the rise of home computers, portable computers, cloud computing, and programming methods that allow the same program to work on different operating systems. SAS's copyright theories would have squelched these innovations, contrary to the fundamental purpose of copyright law.

A. The BIOS of the Original IBM-Compatible PC

In 1981, IBM released its first home computer, the PC. Charles H. Ferguson & Charles R. Morris, *Computer Wars: The Fall of IBM and the Future of Global Technology* at 27–28 (1994). Unlike prior offerings, the IBM PC had an open design. Thanks to that design, add-on innovation in PC software and hardware peripherals flourished. *Id*. at 28–29. To use IBM-exclusive software like the popular spreadsheet program Lotus 1-2-3, however, users initially had to buy IBM computers. *Id*. Although other manufacturers could run the same MS-DOS operating system that IBM used, many best-selling programs required complete hardware and Basic Input/Output System (BIOS) firmware[7]

[7] Firmware is software stored in read-only memory that stays intact even when a computer is switched off. *Microsoft Dictionary* (2005) at 357. Firmware holds

compatibility as well. Thus, the IBM model was the de facto standard. *Id*. at 51–53; *see*, e.g., Stephen Satchell, *The Corona ATP is Faster than the IBM PC AT, But it Has Flaws*, InfoWorld (Jan. 1986), at 50 (using Microsoft Flight Simulator and Lotus 1-2-3 to test PC compatibility).

In order to create a computer that was truly competitive with the IBM PC, other manufacturers needed to duplicate the functionality of IBM's BIOS firmware. *See* Ferguson, *supra*, at 52-53. To avoid exposing themselves to copyright liability, Phoenix, Compaq, and other hardware manufacturers assembled two "teams." *Id.*; Van Lindberg, *Intellectual Property and Open Source: A Practical Guide to Protecting Code* 240–41 (2008). The first "team" analyzed the IBM BIOS and wrote functional specifications about the software's structure, sequence, and organization. Matthew Schwartz, *Reverse Engineering*, Computerworld (November 12, 2001).[8] These functional specifications were passed to the "clean" teams of programmers who had never seen the BIOS source code. Van Lindberg, *supra*. The clean teams created new software from scratch using the interface specifications needed to interact successfully with the IBM PCs: the BIOS interface, including its structure, sequence, and

---

basic pieces of software in a computer, like startup routines and the interface that allows the operating system to interact with the computer hardware. *See generally* Jeff Tyson, *How BIOS Works*, HowStuffWorks, http://computer.howstuffworks.com/bios1.htm.

[8] Available at: http://www.computerworld.com/article/2585652/app-development/reverse-engineering.html.

organization. *Id*.

Once these firms developed their own BIOS firmware, they were able to produce cheaper, faster IBM-compatible computers, and market innovations like the first portable PC. Ferguson, *supra*, at 53–55; *see also Compaq Computer Corporation: Portable Computer*, Encyclopedia Britannica.[9] With more computers and customers now available to them, software developers began to write and distribute more software than ever, innovating with new features and functionality and competing directly on price. The age of home computing began in earnest.

Key to that development was the fact that IBM owned the copyright on the BIOS source code, but could not claim a monopoly on the system of commands the operating system used to communicate with that code. Thus, Compaq and Phoenix were entitled to reimplement the BIOS interface as long as they did not copy any of IBM's code. *Cf. Lotus*, 49 F.3d at 810 (holding that the menu structure and commands of Lotus's interface comprised an uncopyrightable system or method of operation under § 102(b), and that Borland was free to reimplement them). If the law had been otherwise, IBM's ownership of the BIOS code would have given it the ability to stifle competitive innovation, to the detriment of the public.

[9] *See* https://www.britannica.com/topic/Compaq-Computer-Corporation.

B. Major Modern Operating Systems Reimplement the Groundbreaking UNIX Interface

Many popular operating systems today reimplement the interface of one of the earliest operating systems, UNIX. Developed at AT&T Bell labs and launched in 1969, UNIX is widely regarded as the first modern operating system. Heather J. Meeker, *The Open Source Alternative* at 3–4 (2008). It ran on large mainframe and minicomputers owned by corporations, universities, and the government. *Id.*

At the time, AT&T developed UNIX, however, the company was operating under a 1956 consent decree (the result of an antitrust suit) that forbade it from monetizing any project outside of telecommunications and special federal contracts. *Milestones in AT&T History*, ATT.com.[10] Thus, to comply with the decree, AT&T licensed UNIX source code to any interested party for a nominal fee. Meeker, *supra*, at 4. Thanks in part to that open license, computer scientists embraced UNIX, making it the dominant operating system of its day. *Id.* Programmers shared their source code and programming innovations freely, developing and releasing new versions of the operating system. *Id.*

The original versions of UNIX became obsolete as the computers that ran them changed, but the UNIX platform could always return in new forms because

[10] *See* http://www.corp.att.com/history/milestones.html.

AT&T's copyright in the UNIX code didn't extend to its programming interface (the functional mapping of commands to computer behavior). Software developers dissatisfied with available operating systems such as MS-DOS, Windows, and Apple's system, along with UNIX users, reimplemented the UNIX interface to run on a PC.[11] Meeker, *supra*, at 6.

Because the functional programming interface was open, it took a minimal amount of work to make pre-existing software run on subsequent systems. For example, some developers wanted to create a new operating system that would run software made for UNIX, but was also free of AT&T's (or anyone's) intellectual property, specifically a system comprising only free software. *Id.* The GNU project, together with the Finnish programmer Linus Torvalds, produced the Linux operating system, which shares the UNIX programming interface, but uses entirely original code. *Id.*

Today, Linux is widely used throughout the computer industry. Tens of millions of servers run Linux. Steven J. Vaughan-Nichols, *How Many Linux Users Are There (Really)?* Linux Planet (Feb. 18, 2009).[12] Thirty-seven per cent

[11] MS-DOS itself reimplemented the programming interface of an earlier operating system, CP/M. *Paterson v. Little, Brown & Co.*, 502 F. Supp. 2d 1124, 1128 (W.D. Wash. 2007).

[12] Available at: http://www.linuxplanet.com/linuxplanet/reports/6671/1.

of Web servers run on Linux.[13] As of November 1, 2016 Linux was used by 498 of the 500 fastest supercomputers.[14] As of January 2017, Android (which uses the Linux kernel) had a 63.99% total share of mobile and tablet operating systems.[15] Countless Internet-based services from Facebook to ATMs rely on Linux-based high-speed networking systems. Vaughan-Nichols, *supra*.

The varied implementations of UNIX are textbook examples of the importance of § 102(b) to innovation and competition. Thanks to the prevailing interpretation of that provision, innovators could provide their own code behind a UNIX interface, letting customers adopt the right one to fit their needs.

C. The C Programming Language Became Universal Because of Its Uncopyrightable Interface

One of the most important contributions of open interface specifications to computer science was enabling software written in one programming language to run on any operating system by allowing interpreter/compilers to be written for each new operating system and yield the same functional behavior.

---

[13] *See Usage of operating systems for websites*, W3Techs, available at: http://w3techs.com/technologies/overview/operating_system/all; *Usage statistics and market share of Unix for websites*, W3Techs, available at: https://w3techs.com/technologies/details/os-linux/all/all.

[14] *See Operating System Development Over Time*, TOP500 Supercomputer List, http://www.top500.org/statistics/overtime/, which collects data on the 500 most powerful commercially available computer systems (select "Operating System Family" and "Systems Share," then click on "Submit.")

[15] Netmarketshare, Mobile/Tablet Operating System Market Share (October 2014), available at: http://marketshare.hitslink.com/operating-system-market-share.aspx?qprid=8&qpcustomd=1&qptimeframe=M.

The evolution of "C" is a textbook example. Dennis Ritchie, one of the computer scientists who invented UNIX, also invented a new language, called "C," in which to code it. Brian W. Kernighan & Dennis M. Richie, *The C Programming Language*, at ix (1978). Programs written in C use the C standard library to execute their functions and operate the computer on which they run—including tasks as basic as opening and closing files. Once programmers learn C, they can write code that will run on any operating system that can provide a reimplementation of the C standard library.

Today, those operating systems are legion. The C Standard Library has been reimplemented countless times to allow different operating systems to work with programs written in C. For example, Microsoft reimplemented the C Standard Library for Windows as part of the Microsoft C Run-Time Library. *C Run-Time Libraries*, Microsoft Developer Network.[16] Google's reimplementation of the same for Android is called Bionic. *The Native Android API*, Mobile Pearls.[17] Another significant reimplementation was the GNU C Library, which was essential to the GNU Project's effort to create a free UNIX-compatible operating system. *The GNU C Library (glibc)*, The GNU Project.[18]

Limiting the ability to reimplement the C Standard Library would have

[16] Available at: http://msdn.microsoft.com/en-us/library/abx4dbyh(v=vs.80).aspx.
[17] Available at: http://mobilepearls.com/labs/native-android-api/.
[18] Available at: http://www.gnu.org/software/libc/.

severely limited the range of systems on which C programs could run. Each operating system would require a new, incompatible version of the language.

Thus, interface copyright would turn universal programming languages like C into narrow dialects, usable only on a specific operating system. Many innovative software projects would be restricted to a single operating system, or simply never get off the ground. Old programs could become obsolete whenever a new operating system came into use, and new operating systems would be unable to take advantage of the thousands of existing C programs.

D. Treating Interfaces as Copyrightable Would Undermine the Industry Standards for Cloud Computing

Modern cloud computing providers, like Amazon Web Services, rely on a reimplementation of one of the oldest interpreter/compilers: the IBM PC BIOS. Cloud computing allows users to rent space and processing power on distant servers, accessible from anywhere in the world via the Internet. *What is Cloud Computing?*, Amazon Web Services.[19] At their core, cloud computing clusters act as "virtual machines"—imitations of small computers being run on huge servers. *See id.*; *see also* Margaret Rouse, *Definition: Virtual Machine (VM)*, SearchServerVirtualization (Oct. 2011).[20] Virtual machines "call" (or invoke) the functions of the BIOS just like physical computers, but they have no

[19] Available at: http://aws.amazon.com/what-is-aws.
[20] Available at: http://searchservervirtualization.techtarget.com/definition/virtual-machine.

individual physical hardware. *See id.* Instead, a reimplementation of the BIOS allows the server to execute the instructions of all the virtual machines running on it. *See id.*

Cloud computing providers use a similar functional interface to govern how their users can interact with their services. *Amazon Web Services (AWS) and Eucalyptus Partner to Bring Additional Compatibility Between AWS and On-Premises IT Environments*, Eucalyptus (March 22, 2012).[21] For example, most providers rely on Amazon's cloud services application programming interface to allow users to control and operate the cloud computers that they rent. Because the functional interface is not restricted by copyright, companies like CloudStack and Eucalyptus can compete with Amazon to provide the best implementation of it. Businesses that employ cloud services can also write or commission their own proprietary software to perform operations on cloud servers. *Business Applications*, Amazon Web Services.[22] In addition, since major cloud service providers like Amazon, Eucalyptus, and CloudStack use the same functional specifications, their customers can easily switch from one cloud service to another. Steven J. Vaughan-Nichols, *OpenStack vs. CloudStack: The*

[21] Available at: http://www.itbriefcase.net/amazon-web-services-aws-and-eucalyptus-partner.
[22] Available at: http://aws.amazon.com/business-applications.

Beginning of the Open-Source Cloud Wars, ZDNet (Apr. 12, 2012).[23] Software developers can write programs capable of interacting with the above three cloud services, creating new ways for users to access and manipulate information spread out across the Internet.

By contrast, if copyright allowed Amazon to monopolize the functional aspects of its cloud storage interface, Amazon would be able to use that power to lock in its users and cripple new competitors. Because businesses use custom software built around the cloud service provider's interface, switching to a cloud service provider with a different one would require rewriting their cloud software. Given the cost and disruption of doing so, few businesses would be willing to leave their cloud service provider, meaning late entrants in the cloud service market would be hard-pressed to build a customer base. The ultimate result: less choice, less innovation.

Cloud services demonstrate how interfaces link the past and the future of computing. Cloud services exist because their creators could build on the openness of the BIOS and other functional interfaces. As discussed above, IBM enforced copyright on the BIOS *source code* and would certainly have used copyright to control reimplementations of the BIOS functionality if the law allowed. It did not, which meant the BIOS functional mapping between input

[23] Available at: http://www.zdnet.com/blog/open-source/openstack-vs-cloudstack-the-beginning-of-the-open-source-cloud-wars/10763.

commands and output behavior became a kind of shared resource. Using this shared resource, cloud service providers created a new service with a new functional interface that is quickly becoming a compatibility standard in their field. Only time will reveal what new innovations will take advantage of widespread, compatible cloud services.

E. Uncopyrightable Interfaces Spur the Creation of Software That Otherwise Would Not Be Written

When programmers can freely employ any interface without obtaining a costly license or risking a lawsuit, they can create compatible software that the interface's original creator might never have envisioned or had the resources to develop. Copyright in the functionality of interfaces would discourage this innovation by creating potential liability for the mere act of writing a compatible program.

One straightforward and common reason to reimplement another programmer's interface is to make a program compatible with a different application or platform. Small companies and volunteer groups often undertake such projects, but heavy licensing fees or the threat of copyright litigation would hinder this work.

Reimplemented interfaces benefit users as well as developers. Wine is a compatibility layer that reimplements the Windows program interface so Windows programs can run on UNIX-based operating systems like Linux and

Mac OS X. *About Wine*, WineHQ.[24] Millions of people use Wine to make their favorite Windows programs work on other operating systems. *Id.* Wine is free and open source software, and volunteers write much of its code. *Id.* Microsoft has no agreement and no contact with the Wine project. Scott Swigart & Sean Campbell, *Interview with Alexandre Juliard, Head of the Wine Project/CTO of CodeWeavers*, How Software is Built (Sept. 8, 2008).[25] In fact, Microsoft has interfered with Wine users' ability to update their software. Ingrid Marson, *Microsoft Admits Targeting Wine Users,* ZDnet (Feb. 25, 2005).[26] If Microsoft could assert copyright on the Windows functional mapping between input commands and computer behaviors, Microsoft could demand licensing fees from Wine, or sue for damages. Either outcome might lead to the Wine project shutting down permanently, preventing its users from running software they have legally purchased or licensed on their own computers.

In the context of supercomputers, it is often necessary to reimplement an interface in order to make new hardware compatible with existing software. Supercomputers typically have unusual, custom-built hardware reflecting both

---

[24] Available at: http://www.winehq.org/about/.
[25] Available at: http://web.archive.org/web/20130719141118/http://howsoftwareisbuilt.com/2008/09/09/interview-with-Alexandre-Julliard-Head-of-the-Wine-Project-CTO-of-CodeWeavers/.
[26] Available at: http://www.zdnet.com/microsoft-admits-targeting-wine-users-3039189180/.

their purpose and the state of the art in computer design at the time of their manufacture. For supercomputers to operate effectively, they need software written specially for their hardware architecture. *See* National Research Council, *The Future of Supercomputing: An Interim Report* 4, 17 (2003). Supercomputer vendors create tailored interpreter/compiler interfaces like the Basic Linear Algebra Subprograms (BLAS) library so that scientists and mathematicians can use standard input formats to write code for their research and experiments. *See BLAS Frequently Asked Questions*, Netlib (Jul. 25, 2005);[27] IBM, *Basic Linear Algebra Subprograms Library Programmer's Guide and API Reference* (2008).[28]

A shared interface is particularly important for supercomputers, because time on these machines is often limited. *See e.g.*, *Scheduling Policies and Limits*, Ohio Supercomputer Center.[29]Supercomputer users must write and test their programs on smaller computers and only use the supercomputer when they wish to run the program for research or experimental purposes. Therefore, programs must work on both the smaller computer and the supercomputer, even though the two systems have different hardware and demands. BLAS and

[27] Available at: http://www.netlib.org/blas/faq.html.
[28] Available at: http://webpages.uncc.edu/~apanday/documents/BLAS_Prog_Guide_API_v3.0.0.3.pdf.
[29] Available at: https://www.osc.edu/supercomputing/batch-processing-at-osc/scheduling-policies-and-limits.

similar interfaces provide a compatibility standard that makes that possible.

If every supercomputer vendor had its own proprietary input and output formats, then research groups also would find themselves "locked in" to the vendor they worked with at the start of their project. A research project's codebase (all the code they've written) is a significant investment, tied to the formats chosen at the start. *See* National Research Council, *supra*, at 21. Moving to an incompatible new supercomputer could mean losing that investment. *Id.* Thus, supercomputer users would be unable to switch vendors to escape poor service or gain access to new technology without making their codebase useless. New vendors with improved supercomputer technology would be unable to attract customers, making it harder to successfully bring hardware innovations to market.

F. Copyright in Interfaces Would Create an "Orphan Software" Problem

Programmers frequently need to reimplement interpreter/compilers in order to access data or other resources trapped in obsolete software. Software creators go bankrupt or stop supporting their creations for many reasons, and rights in software may change hands many times when startups are acquired or divisions of companies spin off or shut down. Over time, "orphan" software often becomes incompatible with modern computers and other software, particularly as platforms change. For owners of that software, reimplementing

the software's interpreter/compiler may be the only realistic way to reclaim the time and resources they have invested in it. But the copyright owner may no longer be identifiable, much less available to authorize the creation of derivative works.

Section 102(b) helps solve this problem. When a copyright owner goes missing, it is difficult to make derivative works of the code. However, a program's interface specifications are part of its system or method of operation, rather than part of its copyrightable expression. By keeping interface specifications free of copyright, Congress allowed other developers to easily build compatible systems. An orphan program's original implementation may be lost, obsolete, or inoperable, but any developer is free to build a new compatible program. It would subvert § 102(b)'s purposes to allow copyright claims to prevent an entire community of users and third-party developers from switching easily to another service.

The history of the social bookmarking site Delicious is a case in point. Delicious was a popular site where users could post links to interesting content that they found around the web. Bobbie Johnson, *Oh, Delicious—Where Did It All Go So Wrong?*, GigaOm (Sept. 28, 2011).[30] People used a variety of third-party applications that ran on the Delicious functional interface to read and post

[30] Available at: http://gigaom.com/2011/09/28/oh-delicious-where-did-it-all-go-so-wrong.

information on Delicious. *A Tour of Pinboard*, Pinboard.[31] Yahoo! bought Delicious and slowly phased out its development, losing Delicious users along the way. Bobbie Johnson, *supra.* As the size of the community diminished, so did the usefulness of Delicious. Kristina Dell, *Entrepreneurs Who Go It Alone—By Choice*, Time (Oct. 24, 2011).[32] Eventually Yahoo! sold Delicious, and many users decided to find a new place to go. *Id.*

A new social bookmarking site, Pinboard, offered itself up as a haven for former Delicious users. *Id.* By reimplementing the functional mapping of the Delicious interface, Pinboard allowed users to keep using their Delicious-based software, but with Pinboard instead. Pinboard, *supra*. Pinboard was created by one man, Maciej Ceglowski, in his spare time. Dell, *supra*. If Ceglowski had to pay for an expensive license or risk copyright liability in order to reimplement the functional aspects of the Delicious interface, he probably wouldn't have gone ahead with the project. The Delicious-based software would have become useless.

As the above indicates, the open nature of functional interfaces protects the investments of users in a platform or service as much as those of software developers. Some of the best uses of Twitter, a massively popular way to

[31] Available at: http://pinboard.in/tour#api.
[32] Available at: http://content.time.com/time/specials/packages/article/0,28804,2094921_2094923_2094924,00.html.

communicate with the world, have come from add-on applications that interoperate with the Twitter programming interface to provide additional services that dramatically increases its value to users. For example, Sickweather tracks Twitter and Facebook for people posting about being sick, and maps their comments so users can find out what illnesses are going around in their area. *How It Works*, Sickweather.[33] Flipboard lets users access all their social networks and regular news sources together. *Flipboard*, Flipboard.[34] Third-party Twitter clients, programs that display Twitter feeds in different, user-friendly ways, are especially popular. Erick Schonfeld, *The Top 21 Twitter Clients*, TechCrunch (Feb. 19, 2009).[35]

Left to its own devices, Twitter probably could not have created and implemented all of these services. In fact, Twitter hasn't built native apps for some hardware devices, such as Microsoft's Surface tablet. John McDermott, *App Developers Shun Microsoft's Surface*, Ad Age (Dec. 4, 2012).[36] If Twitter were to go out of business or stop supporting its programming interfaces, many users would lose access to their favorite applications. If this happened, a new competitor could come into the market and support those applications by

[33] Available at: http://www.sickweather.com/how.
[34] Available at: http://flipboard.com.
[35] Available at: http://techcrunch.com/2009/02/19/the-top-21-twitter-clients-according-to-twitstat.
[36] Available at: http://adage.com/article/digital/app-developers-shun-microsoft-s-surface/238602/.

reimplementing the functionality of the Twitter interface.

Another example involves Google's "Reader" service, which Google shut down effective July 1, 2013. If the Google reader programming interface was copyrightable, users of Google Reader would have been stranded. It is not, and that is one reason a subsequent service called Feedly initially attracted over 500,000 users by offering a compatible service. Salvador Rodriguez, *Google Reader's demise means big gains for Feedly*, Los Angeles Times (March 18, 2013).[37] Two months later, Feedly's user base "swelled to seven million." David Pogue, *Google's Aggregator Gives Way to an Heir*, N.Y. Times (May 8, 2013).[38]

[37] Available at: http://www.latimes.com/business/technology/la-fi-tn-google-reader-demise-feedly-20130318,0,3173230.story.
[38] Available at: https://www.nytimes.com/2013/05/09/technology/personaltech/three-ways-feedly-outdoes-the-vanishing-google-reader.html?_r=1&.

CONCLUSION

The Court should affirm the district court's conclusion that WPL did not copy any copyrightable element of SAS software.

February 22, 2017

Respectfully submitted,

ELECTRONIC FRONTIER FOUNDATION

By: /s/ Mitchell L. Stoltz
Mitchell L. Stoltz

Kit Walsh
Michael Barclay
ELECTRONIC FRONTIER FOUNDATION
815 Eddy Street
San Francisco, CA 94109-7701
Tel: (415) 436-9333
Fax: (415) 436-9993
mitch@eff.org

CERTIFICATE OF COMPLIANCE

Pursuant to Fed. R. App. P. 32(a)(7)(C), I certify as follows:

1. This Brief of Amicus Curiae in Support of Appellee complies with the type-volume limitation of Fed. R. App. P. 32(a)(7)(B) because this brief contains 6,271 words, excluding the parts of the brief exempted by Fed. R. App. P. 32(a)(7)(B)(iii); and

2. This brief complies with the typeface requirements of Fed. R. App. P. 32(a)(5) and the type style requirements of Fed. R. App. P. 32(a)(6) because this brief has been prepared in a proportionally spaced typeface using Microsoft Word 2011, the word processing system used to prepare the brief, in 14 point font in Times New Roman font.

Dated: February 22, 2017

/s/ Mitchell L. Stoltz

Mitchell L. Stoltz

ELECTRONIC FRONTIER FOUNDATION
815 Eddy Street
San Francisco, CA 94109
Telephone: (415) 436-9333
Facsimile: (415) 436-9993
mitch@eff.org

Counsel for Amicus Curiae
Electronic Frontier Foundation

CERTIFICATE OF SERVICE

I hereby certify that I electronically filed the foregoing with the Clerk of the Court for the United States Court of Appeals for the Fourth Circuit by using the appellate CM/ECF system on February 22, 2017.

I certify that all participants in the case are registered CM/ECF users and that service will be accomplished by the appellate CM/ECF system.

Dated: February 22, 2017

/s/ Mitchell L. Stoltz

Mitchell L. Stoltz

ELECTRONIC FRONTIER FOUNDATION
815 Eddy Street
San Francisco, CA 94109
Telephone: (415) 436-9333
Facsimile: (415) 436-9993
mitch@eff.org

Counsel for Amicus Curiae Electronic Frontier Foundation