The John Marshall Journal of Information Technology & Privacy Law

Volume 11
Issue 1 *Computer/Law Journal - Winter 1991*

Article 5

Winter 1991

Can a Computer Language be Copyrighted? The State of Confusion in Computer Copyright Law, 11 Computer L.J. 97 (1991)

Steve Posner

CAN A COMPUTER LANGUAGE BE COPYRIGHTED? THE STATE OF CONFUSION IN COMPUTER COPYRIGHT LAW*

*Steve Posner***

TABLE OF CONTENTS

* This paper was awarded first prize at St. John's University School of Law in the 1990 Nathan Burkan Memorial Competition sponsored by the American Society of Composers, Authors and Publishers.

** Mr. Posner received a B.A. in English from Syracuse University and was a business reporter in Seattle, Washington. He earned his M.B.A. in Computer Applications and Information Systems from New York University and became a computer programmer, consultant and systems manager before entering St. John's University School of Law in 1988. He currently is a law student at St. John's University and is employed by the New York City Law Department.

I. THE ATTEMPT TO COPYRIGHT A COMPUTER LANGUAGE: A FOCUS FOR INQUIRY

Thousands of jobs and the future of the computer industry will be affected when a court decides whether a computer language can be copyrighted. The issue, never before litigated, arises in *Ashton-Tate Corp. v. Fox Software, Inc.*,[1] a suit now pending in the United States District Court, Central District of California.

Hugely popular business database programs, dBASEII and its successors, dBASEIII and dBASEIV, were created and copyrighted by Ashton-Tate in 1984, 1986 and 1988, respectively. In 1986, Fox began to market Foxbase+ as a dBASEIII "workalike." Although the issue of computer language copyright is not mentioned in the complaint, Ashton-Tate's acquiescence to public use of the dBASE language is posited as an affirmative defense in Fox's answer,[2] and attorneys for both plaintiff[3] and defendant[4] say the issue will be litigated. Edward Esber, Chairman and Chief Executive Officer of Ashton-Tate, wrote in an open letter to the software industry, "Ashton-Tate believes that the structure and organization of the dBASE programs are protectible under copyright law, including the screens, menus, basic file structures and dBASE language embodied in these products."[5]

This paper focuses on the copyrightability of computer languages for two reasons: First, if a language is, indeed, held copyrightable, this author believes the result would be disastrous. Second, the copyright issue can serve as a metaphor around which an inquiry into the confusion dominating computer copyright law can be organized. This paper seeks to develop a rational and functional analytical framework consistent with existing copyright law. The call of Sholkoff,[6] among others, for new computer copyright legislation is also briefly examined.

---

1. No. 88-6837 (C.D. Cal. filed Nov. 18, 1988). (Editor's Note: On December 13, 1990, the United States District Court, Central District of California held that Ashton-Tate's copyrights on dBASE software programs were invalid due to failure to disclose material information to the United States Copyright Office. However, the court reversed its decision in a ruling dated April 18, 1991, and granted Ashton-Tate copyright protection for the dBASE software programs. L.A. Times, Apr. 24, 1991, at D1, col. 5.

2. Answer for Fox Software, Inc., Ashton-Tate Corp. v. Fox Software, Inc., No. 88-6837 (C.D. Cal. filed Nov. 18, 1988).

3. Conversation with Allen Grogan of Blanc Gilburne Williams & Johnston, Attorneys for Plaintiff, Los Angeles (Oct. 11, 1989) (notes on file with author).

4. Conversation with Thomas A. Lewry of Brooks & Kushman, Attorneys for Defendant, Southfield, Mich. (October 14, 1989) (notes on file with author).

5. E. Esber, An Open Letter to the Software Industry (1988).

6. Note, *Breaking the Mold: Forging a New and Comprehensive Standard of Protection for Computer Software*, 8 COMPUTER/L.J. 389 (1988).

II. POTENTIAL RAMIFICATIONS FOR THE COMPUTER INDUSTRY

At present, approximately 100,000 programmers[7] write professionally in the dBASE dialects.[8] If the court finds that Ashton-Tate owns a copyright in the dBASE language, these programmers and the users of a vast installed base of their programs will be liable for copyright infringement.[9] All future programs written in the dBASE language will be subject to licensing fees and/or royalties, which will force the price of development to increase. Companies that have developed dBASE variants might find themselves enjoined from further sale of their products if Ashton-Tate should deny licenses to their creators. Accordingly, those features that exist in such variants would disappear from the marketplace.

For purposes of this discussion, laches, acquiescence and any other legal issues beyond the nature of what can be copyrighted will be ignored. We are not concerned with the resolution of the particular case, *Ashton-Tate v. Fox*, because the issue goes beyond the dBASE language. If one language is copyrightable, are not all languages that have not been expressly released into the public domain also subject to copyright? Such possessory interests would run contrary to a major trend in the computer industry: the movement toward software compatibility.

---

7. Estimate supplied by K. Strehlo, Editor, DBMS MAGAZINE, Redwood City, California. (Nov. 12, 1989).

8. The dBASE language had its genesis in two pre-existing programs. The first was called JPLDIS and was created by the Jet Propulsion Laboratories in the mid-1970's. The second was called Vulcan; it was based on JPLDIS but greatly evolved by Wayne Ratliff, who licensed it to Ashton-Tate in 1980 and sold all rights to Ashton-Tate in 1983. Vulcan was marketed under the name dBASEII. The language as it then existed was added to or "extended" in dBASEIII. The language as used in Foxbase+ is identical to that of dBASEIII, but other programs, such as Nantucket Corporation's Clipper, have extended the language further. Presently, besides Ashton-Tate, there are at least nine different companies publishing database programs using variations of the dBASE language, known in industry vernacular as "dialects." There are also two dozen or more products such as graphics and support libraries, application/code generators, compilers and debuggers, which use the dBASE language as their common starting point and extend it in various ways.

9. A purchaser of dBASEIII+ is licensed to use the product on one machine. However, if written using a separate editor and compiled in Clipper, Force, or any of a number of products, a custom program written in dBASE will run without either the programmer or his customer ever having bought Ashton-Tate's product. The programmer can presently license his client to make as many copies as he wishes of the custom program. If the dBASE language is copyrightable as part of the dBASEIII+ program, then use of the language may be under the same license restrictions as the rest of the program and any other restrictions Ashton-Tate imposes. Ashton-Tate could license the dBASE language to be used only in the dBASE environment. All the programmers and customers currently using dBASE variant programs outside the dBASE environment may be infringing.

Exemplifying this trend is another computer language, "C." Although C was created by Dennis M. Ritchie of AT & T,[10] independent software developers made C extraordinarily popular, largely because programs written in it can run in many different computing environments. Philippe Kahn, President of Borland International, a company that develops language compilers, writes "The C language is firmly established as the favorite programming language of serious developers. The freedom, unmatched portability, and remarkable access of this language have made it the choice of computer professionals around the world."[11] Portability is a term hardly distinguishable from compatibility.[12]

The growth of the dBASE language has been similar to that of C. As dBASE and its variants became ubiquitous,[13] programmers flocked

---

10. B. KERNIGHAN & D. RITCHIE, THE C PROGRAMMING LANGUAGE (1978).

11. Kahn, *Forward* to H. SCHILDT, ADVANCED TURBO C at ix (1977); *See also*, Duncan, *Forward* to R. LAFORE, TURBO C, PROGRAMMING FOR THE IBM (1988). Mr. Duncan, a columnist for *Dr. Dobb's Journal*, a prominent trade journal, attributes the continuing development of C to "the penetration into academia that was made possible by Bell Laboratory's broadminded licensing plans for universities." AT & T created UNIX, the operating system under which C originally ran. UNIX is itself an example of the trend toward compatibility. It has grown enormously popular in the industry because it can run on a wide variety of machines from different manufacturers, and because programs written for UNIX running in a one machine environment will run in most other UNIX environments with minor alterations. In the mid-1970's, UNIX and C seemed to promise that one would soon be able to write a program in C and be able to run it, almost without alteration, in any environment. The subsequent explosion of the C language, which soon spread beyond UNIX into many operating system and machine environments, should give the reader some indication of the value placed on compatibility in the computer industry.

12. A. RALSTON & E. REILLY, ENCYCLOPEDIA OF COMPUTER SCIENCE AND ENGINEERING 253, 1156 (1983).

> PORTABILITY - For articles on related subjects see COMPATIBILITY. . . Software is said to be *portable* if it can, with reasonable effort, be made to run on computers other than the one for which it was originally written. Portable software proves its worth when computers are replaced or when the same software is run on many different computers . . .

Id. at 1156 (emphasis in original).

> COMPATIBILITY - Two compilers or language translators are said to be *compatible* if source programs written for a compiler on one computer will compile and execute successfully on the other. . .[T]he more common use of compatibility in computing is applied to two machines, two configurations, two operating systems, or two software packages with respect to the ease with which programs or data can be converted from one to the other. The term normally applied to a program to describe the ease with which it can be converted from one system to another is *portability*.

Id. at 253 (emphasis in original).

13. The dBASE language, like C, has grown beyond its original operating environment. Although originally developed for the IBM PC environment, Fox Software has developed a version that runs in the Apple Mackintosh environment. The dBASE language is more specialized than C and less generally useful; therefore its spread to other environments has been less explosive.

to it and an industry-wide skill-pool developed. Had access to either of these languages been restricted, programs written in these languages would probably have been limited to the environments approved by the language creators. Marketability of skills developed in these languages would also have been limited. Evolution of the pools of programming knowledge, and of the languages themselves, inevitably would have been slowed. If Ashton-Tate succeeds in copyrighting the dBASE language, it and new languages will be protected, making such languages less accessible and less desirable to programmers. The development and acceptance of new industry standard languages will be less likely.

A further effect could be a general slowdown in applications development, since the ability of corporate data processing managers to acquire or create new applications that would be compatible (that is, would share data and/or development environments) with their preexisting programs and data would be severely limited. Despite the decision of the Third Circuit in *Apple Computer, Inc. v. Franklin Computer Corp.*,[14] which ignored the importance of compatibility, it is a vital issue in the computer industry. Inmon writes:

> The internal development is usually done in a widely used language (COBOL, PL-1, FORTRAN, etc.). Selecting the language(s) that is used parallels the same reasoning as selection of operating systems and DBMS—an available talent pool and the existence of a wide variety of software written in the selected languages.[15]

Inmon notes that while it is not always easy to move data and programs from one machine computer to another, even when they use the same language, it is a great deal easier than converting from one language to another.[16]

Undeniably, the progress of competitive business practices and technology are dependent on both software advances and the ability to efficiently handle data. If computer language copyright limitations were imposed only in America, the competitiveness of American companies compared to those of other technological nations would suffer.

> The relationship between technology and the marketplace is a symbiotic one. On one hand technology depends on growth and acceptance in the marketplace for sustenance and the opportunity to become established. At the same time, many marketplaces depend on advances in technology for existence and the opportunity to grow. Without technological innovation entire marketplaces would never exist.[17]

---

14. Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1253 (3d Cir. 1983).

15. W. INMON, TECHNOMICS: THE ECONOMICS OF TECHNOLOGY AND THE COMPUTER INDUSTRY 33 (1986).

16. *See id.* at 35.

17. *Id.* at 2.

Retarding technological growth is not the purpose of copyright law.

III. THE PURPOSE OF COPYRIGHT: TO PROMOTE THE PROGRESS OF SCIENCE AND USEFUL ARTS

Article I, Section 8 of the United States Constitution states that the purpose of copyright is "To promote the progress of science and useful arts, by securing for a limited time to authors and inventors the exclusive right to their respective writings and discoveries." A question for the *Ashton-Tate* court and, prospectively, for Congress, is whether copyright should be granted where the effect of securing such rights to authors will retard the progress of science rather than promote it. The clear weight of authority is against retarding progress.

"The sole interest of the United States and the primary object in conferring the monopoly lie in the general benefits derived by the public from the labors of authors,"[18] wrote the U.S. Supreme Court. Nimmer, the noted copyright authority, observes that:

> The primary purpose of copyright is not to reward the author, but is rather to secure 'the general benefits derived by the public from the labors of authors . . . . [T]he public benefits from the creative activities of authors, and . . . the copyright monopoly is a necessary condition to the full realization of such creative activities.[19]

Moreover, "when technological change has rendered its literal terms ambiguous, the Copyright Act must be construed in light of this basic purpose."[20]

Copyright law has provided no stimuli to language development; this is obvious because languages have developed without being copyrighted. It is arguable that languages are developed and gain general acceptance and usefulness precisely because they are not protected.[21]

Undoubtedly, the creator of a language does invest considerable effort in his creation. Conley and Bryan write: "Regardless of the source of the elements, the author legitimately may claim that synthesizing the entire system required substantial creativity and ingenuity."[22]

However, not everything that requires originality and effort to create is copyrightable.

> In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation,

---

18. Fox Film Corp. v. Doyal, 286 U.S. 123, 127 (1932).

19. M. NIMMER, NIMMER ON COPYRIGHT § 1.03, at 1-31, 1-32 (1988).

20. Twentieth Century Music Corp. v. Aiken, 422 U.S. 151, 156 (1976).

21. Compilers, as expressions of a language, are copyrightable. The competition to build a better compiler undoubtedly does contribute to the further development of a language.

22. Conley & Bryan, *A Unifying Theory for the Litigation of Computer Software Copyright Cases*, 6 COMPUTER/L.J. 55, 62 (1985).

concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work.[23]

Policy notwithstanding, does the nature of a computer language qualify it for copyright protection?

IV. THE IDEA/EXPRESSION DICHOTOMY AND COMPUTER LANGUAGES

One of the basic questions courts ask in determining whether a work is copyrightable is whether that work or its elements are "idea" or "expression." "Ideas" are not copyrightable; "expressions" are.[24]

The idea/expression dichotomy had its genesis in *Baker v. Selden*,[25] which held that "copyright of a work on mathematical science cannot give to the author an exclusive right to the methods of operation which he propounds."[26] The focus of this case was a bookkeeping system that included blank forms with ruled lines and headings. "The mere copyright of Selden's book did not confer upon him the exclusive right to make and use account-books, ruled and arranged as designated by him and described and illustrated in said book."[27]

The court in *Mazer v. Stein*[28] held that "protection is given only to the expression of the idea—not the idea itself."[29]

What, in a particular case, is "idea" and what is "expression" is considered one of the most difficult distinctions to make in copyright law. The traditional test was put forward in *Dymow v. Bolton*[30]: "[I]f the same idea can be expressed in a plurality of totally different manners, a plurality of copyrights may result, and no infringement will exist."[31] In the more modern language of *Apple Computer, Inc. v. Franklin Computer Corp.*, the issue is "whether particular ideas and expressions have merged."[32]

However, whether such plurality or merger exists can be difficult to determine. Learned Hand wrote, "Nobody has ever been able to fix that boundary [between idea and expression], and nobody can."[33] He also noted that "decisions must therefore inevitably be ad hoc."[34]

---

23. 17 U.S.C. § 102(b) (1988).
24. *Id.*
25. 101 U.S. 99 (1879).
26. *Id.* at 103.
27. *Id.* at 107.
28. 347 U.S. 201 (1954).
29. *Id.* at 217.
30. 11 F.2d 690 (2d Cir. 1926).
31. *Id.* at 691.
32. 714 F.2d at 1253.
33. Nichols v. Universal Pictures Corp., 45 F.2d 119, 121 (2d Cir. 1930).
34. Peter Pan Fabrics, Inc. v. Martin Weiner Corp., 274 F.2d 487, 489 (2d Cir. 1960).

The author believes that when courts understand precisely what a computer language is, and then apply idea/expression analysis, they must conclude that a computer language is almost always an uncopyrightable idea.

V. THE THEORETICAL NATURE OF A COMPUTER LANGUAGE: IDEA

Webster's New Collegiate Dictionary defines a language as, *inter alia*, "a formal system of signs and symbols (as FORTRAN or a calculus in logic) including rules for the formation and transformation of admissible expressions."[35] De Remer defines the elements of a computer language: "A 'grammar' is a formal device for specifying a potentially infinite 'language' (set of strings) in a finite way."[36]

A "device" as used by De Remer and other theorists is not a tangible object that performs a task; rather, it is the theoretical construct of such a device. Part of such a grammatical device can be expressed as follows:

FIGURE 1[37]

```
32 | # The Lexicon of Context-Free Grammars.
33 |
34 | Identifier
35 |     = Upper-case-letter
36 |     | Identifier Lower-case-letter
37 |     | Identifier Digit
38 |     | Identifier Underscore
39 |     ;
40 | String        = Quote Any-characters Quote        ;
41 |
42 | Spaces
43 |     = Separator
44 |     | Spaces Separator
45 |     ;
46 | Separator
47 |     = Blank
48 |     | New-line
49 |     | Comment
50 |     ;
51 | Comment     = Sharp Blank Any-character New-line
```

What is described here is a set of rules. The rule of line 40 dictates that any group of characters beginning and ending with quotation

35. WEBSTER'S NINTH NEW COLLEGIATE DICTIONARY 672 (1988).

36. De Remer, *Review of Formalisms and Notations*, in COMPILER CONSTRUCTION, AN ADVANCED COURSE 37 (F. Bauer & J. Eickel ed. 1976).

37. *Id.* at 51.

marks will be treated as a string of characters that can be manipulated—for example, a client's name. Line 51 shows that the same group of characters preceded by a sharp sign (#) will be treated as a comment. (A comment is placed into a program's source code for the edification of any programmer reading it, but is ignored by the compiler that translates the code into machine language, and thus by the computer itself.)

Thus, a computer language is a set of rules, an example of which, in the dBASE language, is this: If a programmer writes "SET DEVICE TO PRINT," the output of the program will be directed to the printer. If the programmer breaks the rule by writing "DEVICE SET TO PRINT," the program will not work.

Different languages have different sets of rules for accomplishing tasks. For example, in MAI/Basic IV's Business Basic language, the programmer might designate the printer device as "P1." He would assign that printer to one of several "channels" representing different devices; we will use Channel (1). By writing the statement, "Print (1) X," he would direct the output of the program, the variable X, to the printer.

Having chosen a language, if the programmer writes a statement following one rule, the corresponding algorithm or process will be executed. Certain rules, such as those governing repetitive processing loops, require multiple statements. For example, from the dBASE language:

```
DO WHILE .NOT. EOF ()
. . . A SERIES OF COMMANDS . . .
ENDDO[38]
```

Furthermore, the use of some rules requires the use, at some prior place in the program, of others. A command to print a variable cannot work unless the value of the variable has previously been established.

Beyond this, however, the set of rules is not sequenced. In a programmer's manual, the creator of the language may list the various rules in some order, such as alphabetical order. But the language itself is a jumble of these rules, given order only when the programmer uses the language to write an ordered series of statements that becomes a program. The requirement of Rule B that Rule A first have been satisfied can be seen as an attribute of Rule B.

Each rule thus exists essentially in isolation.

---

38. In plain English, this means that starting at the beginning of a database, the program will subject every record in that database to the algorithms and processes called by the same series of commands, until the end of the database is reached.

What are the prospects of such a set of grammatical rules being copyrighted?

First, the law is couched in terms of languages not being copyrightable. "Just as one cannot copyright the arabic number, so one cannot copyright the Latin alphabet or the English language."[39]

Second, courts have held that rules of a game are not copyrightable. "The rules of the game are perforce in the public domain as well as the game itself."[40] It is arguable that grammatical rules are analogous.

Third, although the court in *Hartfield v. Peterson*[41] held that a compilation of 75,000 code words and phrases was copyrightable both as to the compilation as a whole and as to the individual phrases themselves,[42] a computer language is distinguishable. A computer language is not a compilation because it does not necessarily appear in any set order or arrangement. Language commands and functions can be ordered in a wide variety of ways by a programmer. The arrangement of a code book is in the order given it by the original author and no other. The potential copyrightability of individual phrases is more problematical. Conley and Bryan write: "Although any statement by Learned Hand in a copyright decision is not to be dismissed lightly, it can only be inferred that the remark is aberrant dictum. The statement was unnecessary to the decision since the defendant had copied portions of the compilation."[43] Rather than simply dismiss the dictum, it may be more useful to note that many of the elements of computer languages are not phrases so much as phrase structures. An example in the dBASE language is:

> LIST [<scope>] [<expression list>] [FOR<condition>] [WHILE <condition>] [OFF] [TO PRINT][44]:

This is the rule for writing a command that will list certain items of data residing in a database either to the screen or to a printer. The phrase itself can be written many different ways, each undoubtedly an expression. Even the descriptor of the rule, as typed above, is an expression. But the rule itself is only a rule, as the language as a whole is a set of rules. The closest analogue to such a single linguistic rule is an

---

39. West Publishing Co. v. Mead Data Cent., Inc., 616 F. Supp. 1571, 1579 (D.C. Minn. 1985), *aff'd*, 799 F.2d 1219 (4th Cir. 1985), *cert. denied*, 479 U.S. 1070 (1986).

40. Affiliated Hosp. Prods. Inc. v. Merdel Game Mfg. Co., 513 F.2d 1183, 1188 (2d Cir. 1975); *see also* M. NIMMER, *supra* note 19, § 218[H], at 2-213 n.70.

41. 91 F.2d 998 (2d Cir. 1937).

42. *Id.* at 999. ("Both the phrases, so far as they were his, and the arrangement were proper subjects of copyright.").

43. Conley & Bryan, *supra* note 22, at 106 (citation omitted). "The remark" referred to by the author is quoted in the text accompanying note 42, *supra*. *Hartfield* was not written by Learned Hand, but by Augustus N. Hand.

44. Transcription of the "LIST" help Screen from dBASE III+.

equation (which is not copyrightable). The closest analogue to the set of rules that comprises a language is a catalogue schema (which is copyrightable insofar as it indexes a set arrangement of information).[45] A language schema is distinguishable because it describes only itself; there is no separate data which it organizes and makes more valuable.

The dBASE language does contain certain elements that can be considered true "phrases." However, these are few and generally not the sort of expressions that ought to be considered original. Examples are "SET DEVICE TO PRINT," "CLEAR MEMORY," "APPEND BLANK," etc. It is doubtful that these rise even to the minimal level of originality required by copyright law.

VI. THE CONTEXT OF A COMPUTER LANGUAGE: COMPILERS AND OTHER EXPRESSIONS

While the above is a theoretical description of a human-usable computer language, this language is useful only in the context of a compiler and in the ability of the compiled program to handle data.

The compiler translates the commands entered by the user into a form to which the computer can respond. Typically, compilers are commercially sold programs which come bundled with certain features that turn the program into a development environment for that particular language. For example, Borland's Turbo C compiler is sold with its own editor and debugger.[46] Microsoft's C compiler also comes with an editor and debugger, although these programs offer differences in form, feature and performance sufficient to provide a meaningful choice between the two products.

The compiler itself consists, conceptually, of a set of modules that translates individual user-entered commands. The translation modules must exist in a set relation with one another in order that the user-entered commands may work together.

However, from compiler to compiler, the modules need not work exactly the same way. Their functions are substantially similar, otherwise there would be no consistency to the language. There are, however, very significant differences in the exact procedures executed by

45. West Publishing Co. v. Mead Data Cent., Inc., 616 F. Supp. 1571 (D.C. Minn. 1985), *aff'd*, 799 F.2d 1219 (4th Cir. 1985), *cert. denied*, 479 U.S. 1070 (1986).

46. An editor is a program that facilitates the writing of programs. It provides a screen into which program commands can be entered and certain features that make program writing easier. For example, "Brief", a commercially published editor, will remember the last several commands entered and will allow the programmer to eliminate them—a very useful feature if one has entered several commands in different places throughout a 10,000 line program and then finds that the approach was invalid. A debugger allows the programmer to trace the individual steps of a program's execution in order to find and eliminate errors. Again, a debugger can have many features.

the translation modules, which are themselves programs. Thus, in trade magazines it is common to see such comparisons as the following: "The most surprising run-time result is Turbo C's poor performance on the GRAPH test. The ellipse() and floodfill() functions perform more than twice as slowly as their Microsoft counterparts."[47]

The fundamental reason that Foxbase+ has been able to make inroads into Ashton-Tate's market is that the Foxbase+ compiler executes programs more than fifteen times faster than Ashton-Tate's dBASEIII+.[48] Additionally, while both compilers execute a core set of language rules that are recognized industry wide, Foxbase+ includes certain commands not included in dBASEIII+. Such additions are called, in the trade, "extensions." One reason that the Nantucket's Clipper compiler has made inroads into the dBASEIII+ market is that it is not only much faster than dBASEIII, but it also offers many more functions.[49]

Many types of compilers exist. Some compile an entire program and then run it. Some compile and run the program one step at a time, and are called "interpreters." Many languages require another process beyond compiling, called "linking." Linking occurs when the program is not complete in itself, but must be "linked" with pre-existing modules, already compiled, in order to run. The two ellipse() and floodfill() functions noted above are examples. The compiler comes with a "library" of such functions which, in writing the program, the user calls into operation simply by mentioning the function name. Third-party libraries which provide further extensions to languages are also sold.

The second contextual area of concern is with file structures. Most languages are designed with underlying schemes for the creation of files

---

47. Ladd, *Benchmarking Turbo C and Quick C*, DR. DOBB'S JOURNAL 89-90 (Aug. 1989). The parentheses are part of the name of the function and also serve a purpose. Items of data enclosed in the parentheses are sent to the function which then acts upon the data and returns it to the main program. Suppose, for example, we had a function called "double()", which doubles any number or numeric variable it acts upon. If, in the main program, we wrote the line "double (5)", the function would return the value 10. Obviously, most functions serve more complex purposes.

48. The following is an excerpt from Fox Software advertisement, 2 DBMS MAGAZINE No. 13, at 7 (Dec. 1989):

> "NOTHING IS FASTER.
>
> Fox Software products are famous for their unmatched execution speed. . . FoxPro is up to eight times faster than dBASE IV—more than 15 times faster than dBASEIII PLUS!"

49. S. STRALEY, PROGRAMMING IN CLIPPER 37-40 (1988). Of some 150 functions available in the Summer '87 release of Clipper, only seventy-eight are supported by dBASEIII+. Functions are provided by compiler publishers so that programmers do not have to reinvent the wheel each time they need to accomplish a common but perhaps complex task. A programmer is therefore more likely to buy a program that provides many functions.

that can then be customized to a user's or programmer's specifications. Thus, all the data required for efficient running of, for instance, a dental office can be organized and stored in a dBASE or Foxbase program.

The dBASE language was designed to create files of certain types: "data", "memo" and "index." Each type has a certain design. For example, a "data" file can consist of a number of fields, each field being a piece of information. These fields can be of type "character", "number", "date" or "memo". Control characters surrounding each field determine the size and type of each field. In every database file, irrespective of its purpose or organization, every field of type "number" will be surrounded by the same control characters. Every database file will have the same control characters indicating the beginning of that file, and the same characters designating the end. Dialects of the dBASE language vary in how flexibly they can utilize these basic file structures. For example, while Ashton-Tate's dBASEIII+ limited the user to files of no more than 128 fields, Nantucket's Clipper at least doubles that number.

A part of the basic purpose of all dBASE dialects is to access files of the specific structure described above. The language is tied to that basic file structure. If the basic file structure is copyrightable, then even if the language itself is in the public domain, it might as well be copyrighted because it can only be used in conjunction with those file structures.[50]

Such a basic file structure, prior to the entry of any user data, is merely an organization of control characters designating the beginning and end of the file, the beginning and end of each record and the beginning, end and field-type of each field.[51] Basic file structures can take various forms. Thus, the file created by a C-language compiler[52] is different from the file created by a COBOL-language or dBASE-language compiler, although the user may enter identical information into each, through different programs. In this sense, a basic file structure can be seen as an expression of the idea of holding information. However,

50. This is basically true of dBASE but not of all languages. C language programs can be written to access dBASE files and other types of files, as well. This reflects an aspect of the idea of "C": maximum flexibility. By contrast, dBASE focuses on ease of use and sacrifices flexibility to that end. The amazing speed of product evolution in the computer industry makes categorical statements dangerous. As of March, 1991, third party vendors are offering products that allow graphic images to be incorporated into dBASE language applications. A fourth type of non-standard file is thus available and the flexibility of the language is increased. Footnote 53 should be read in this light.

51. Again, one must be careful of overgeneralizing. The more recent "object-oriented" programming environments also embed or allow the programmers to embed into a data file—or "object"—instructions as how the data contained in the object may be processed, reallocating some control from the program to the data file.

52. Absent instructions to create a different type of file.

whatever information a user derives from the standard file structures limits,[53] the basic file structure is neither "original" nor "informative" in ways that case law requires for copyright.[54]

Further, both case law[55] and Nimmer[56], in treating compilations and forms, look to whether the basic structure suggests to the user a methodology or rationale for ordering his own information. The format of a basic file structure, by contrast, is altogether invisible to the typical user.[57] It suggests no ordering of user information, nor does it offer any useful information except, again, for knowledge of what it can or cannot do. For example, the basic schema of a legal reporter is <VOLUME> <SERIES NAME> <PAGE> and does not, in itself, rise to the slight originality requirements of copyright law; after all, every court reporter uses this basic schema.[58] The schema of a basic file structure goes even less far; it does not even specify the order in which the elements appear.

Thus, a usable computer language has three aspects: the theoretical set of rules that comprises the language's grammar, the compiler and the basic file structures. The rules are clearly non-copyrightable "idea." The compiler is "expression" and is copyrightable. The basic file layouts require so little originality and are so uninformative as not to be copyrightable.

To determine whether a language is copyrightable, it is important

---

53. One cannot, for example, enter a graphic image into a standard dBASE III file, or create a file structure of more than 128 fields. A programmer can determine this by examining the basic file structures, although a typical user would lack the necessary knowledge to do so. Programmers and users would be more likely to find this information in the manual.

54. Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1242-43, (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1986) ("Although some courts have stated that the meaning of Baker v. Selden is that blank forms cannot be copyrighted, this circuit, like the majority of courts that have considered the issue, has rejected this position and instead held that blank forms may be copyrighted if they are sufficiently innovative that their arrangement of information is itself informative . . . This is not to say that all blank forms or computer files are copyrightable. Only those that by their arrangement and organization convey some information can be copyrighted.").

55. Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d 1240, 1243 (3d Cir. 1983); Manpower, Inc. v. Temporary Help of Harrisburg, Inc., 246 F. Supp. 788 (E.D. Pa. 1965); Edwin K. Williams & Co. v. Edwin K. Williams & Co.—E., 542 F.2d 1053 (9th Cir. 1976).

56. M. NIMMER, *supra* note 19, at 2-201 ("Thus books intended to record the events of a baby's first year, or a record of a European trip, or any one of a number of other subjects, may evince considerable originality in suggestions of specific items of information which are to be recorded, and in the arrangement of such items.").

57. *But see supra* note 54.

58. West Publishing Co. v. Mead Data Cent., Inc., 616 F. Supp. 1571, 1579 (D.C. Minn. 1985), *aff'd* 799 F.2d 1219 (4th Cir. 1985), *cert. denied*, 479 U.S. 1070 (1986) (concerned not the use of the basic court reporter schema, but rather the pagination of cases and pages of cases within that schema).

to separate the theoretical construct—the idea of the language and its rules—from its practical expressions: manuals, descriptive writings, compilers, etc.

VII. OVERBROAD DEFINITION OF A PROGRAM'S IDEA: THE WORM IN THE *APPLE* CASES[59]

"Idea" and "expression" with regard to languages can be most usefully analyzed if one views the grammar as the idea and the available compilers as the expressions.

It is possible, however, to define "idea" less rigorously. For example, a computer language can be viewed as an instruction set that enables the user to create and manipulate screens, files, file structures and executable programs. With such a view, nearly every language becomes an expression of the idea, because nearly all languages have those capabilities.

Certainly good reason exists to avoid too detailed a definition of "idea." At least one court has cautioned that "the 'idea' of any work could always be defined in such detail that the description of the expression would add nothing to the 'idea', thus allowing a defendant to engage in all but verbatim copying."[60]

It is equally possible to define an idea in so little detail that almost any attempt to develop a new work based upon that idea will run afoul of an earlier copyright.

A practical example of failure to appreciate this problem exists in *Apple Computer, Inc. v. Franklin Computer Corp.*:[61]

> Franklin claims that whether or not the programs can be rewritten, there are a limited "number of ways to arrange operating systems to enable a computer to run the vast body of Apple-compatible software," Brief of Appellee at 20. This claim has no pertinence to either the idea/expression dichotomy or merger. The idea which may merge with the expression, thus making the copyright unavailable, is the idea which is the subject of the expression. *The idea of one of the operating system programs is, for example, how to translate source code into object code.* If other methods of expressing that idea are not foreclosed as a practical matter, then there is no merger. *Franklin may wish to achieve total compatibility with independently developed application association programs written for the Apple II, but that is a commercial and competitive objective which does not enter into the somewhat metaphysical issue of whether particular ideas and expressions have*

59. Not to be confused with WORM, the acronym referring to Write Once Read Many-Times storage devices.

60. Midway Mfg. Co. v. Bandai-America, Inc., 546 F. Supp. 125, 148 (D.N.J. 1982).

61. 714 F.2d 1240 (3d Cir. 1983).

merged.[62]

The idea of that operating system program could just as easily, and perhaps more accurately, have been stated as: a method translating source code programs written for the Apple II computer into machine language executable by that computer.

The "idea" of the operating system program is further distorted by judicial refusal to recognize compatibility as an element of a program's "idea." No reason is given for this refusal. Yet compatibility is an idea of profound consequence in the computer industry. Entire sectors of the industry have grown out of the need to make data, housed in basic file structures, created by different companies, and programs written by those companies, able to function with one another. The third circuit's statement in *Franklin* seems plainly inconsistent with its own later dictum in *Whelan Associates v. Jaslow Dental Laboratory, Inc.*: "[T]he 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."[63] This dictum was based upon the court's reading of *Baker v. Selden*[64] as holding that an idea is: "the end sought to be achieved."[65] If compatibility is part of a program's purpose—whether the reason for that purpose is "commercial and competitive" or not—how can it be anything but a part of the program's idea?

If the idea/expression dichotomy forms the dividing line between what is copyrightable and what is not, a reasonable and consistent analytical framework for defining the "idea" of a computer program must be devised in light of the primary purpose of copyright law: to promote the progress of science and the useful arts. To discourage compatibility is to retard progress.

At least with regard to computer languages, the method most likely to promote progress and least likely to disrupt the progress of the industry, is to regard the grammar/compiler distinction described in the previous section as equivalent to that between idea and expression. Because the creator of a language must also create a compiler in order to demonstrate the language's usefulness, and because the compiler is a copyrightable expression, the author's rights in his creation would be secured for a limited time. This would motivate compiler authors to create, as it would take some time for someone to come up with a superior compiler in that language. However, the creator of that next compiler would also be motivated, because he could also obtain copyright

62. *Id.* at 1253 (emphasis added).

63. Whelan Assocs. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1236 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1986) (emphasis omitted).

64. 101 U.S. 99 (1879).

65. *Id.*

protection, at least, so long as he did not steal any code from the language creator's compiler. This, too, is consistent with the purpose of the copyright statutes.

VIII. COPYRIGHT OF OPERATING SYSTEMS: BLURRING THE DISTINCTION BETWEEN COPYRIGHT AND PATENT COVERAGE

Title 17 U.S.C. § 102 provides that, "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system, method of operation, concept, principle, or discovery, regardless of the form in which it is described, explained, illustrated, or embodied in such work."[66]

The courts in *Apple Computer, Inc. v. Franklin Computer Corp.*[67] and *Apple Computer, Inc. v. Formula Int'l, Inc.*[68] both held operating systems to be copyrightable.[69]

This section is written, not because of any disagreement with what the third and ninth circuits have held with regard to the copyrightability of operating systems, but because of the potential damage that could occur if these holdings were improperly read. In particular, of course, the potential effect on the copyrightability of computer languages is at issue. A more general threat is the incursion of copyright into realms historically ruled by patent law—an event foreseen by Nimmer in his concurring opinion to the Final Report of the National Commission on New Technological Uses of Copyrighted Works (CONTU):

> What is most troubling about the Commission's recommendation of open-ended copyright protection for all computer software is its failure to articulate any rationale which would not equally justify copyright protection for the tangible expression of any and all original ideas (whether of computer technology, business, or otherwise). If literary works are to be so broadly construed, the Copyright Act becomes a general misappropriation law, applicable as well in what has traditionally been regarded as the patent arena . . . .[70]

The *Apple* decisions would seem, at first glance, clearly contrary to the language of 17 U.S.C. § 102(a), which prohibits copyright of "systems."[71] However, both courts noted two aspects of an operating system

66. 17 U.S.C. § 102(b) (1982).

67. 714 F.2d 1240 (3d Cir. 1983).

68. 725 F.2d 521 (9th Cir. 1984).

69. Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d at 1250-52; *Formula*, 725 F.2d at 523-25.

70. NATIONAL COMMISSION ON NEW TECHNOLOGICAL USES OF COPYRIGHTED WORKS, FINAL REPORT 1 (1978) (CONTU REPORT), *reprinted in* M. NIMMER, CASES AND MATERIALS ON COPYRIGHT 124 (3rd ed. 1985).

71. 17 U.S.C. § 102(a) (1988).

program—the processes carried out, and the instructions that define these processes and make possible their execution.[72] It is these instructions that the courts ruled copyrightable as "literary works" under section 101.[73] Focusing on the literary aspects of an operating system program, the third circuit simply held the systemic aspects of operating systems irrelevant:

> Since it is only the instructions which are protected, a "process" is no more involved because the instructions in an operating system program may be used to activate the operation of the computer than it would be if instructions were written in ordinary English in a manual which described the necessary steps to activate an intricate complicated machine.[74]

Webster's defines a system as, *inter alia*, "a group of devices or artificial objects or an organization forming a network especially for distributing something or serving a common purpose . . . [other examples] . . . a data processing system."[75]

Apple Computer, Inc. v. Franklin Computer Corp., in particular, can be interpreted to dismiss the systemic aspect of an operating system program as irrelevant to a decision as to whether a copyright should be granted.[76]

It must be remembered that the code of an operating system is itself merely an expression of the underlying system. The system itself—the entity defined by Webster's—is similar to a language grammar: the idea that underlies the expression. There are, as the losing defendant argued in *Franklin*, only a limited "number of ways to arrange operating systems to enable a computer to run the vast body of . . . compatible software."[77] Defendant did not err. Defendant lost the case because it copied Apple's code line by line. Defendant could have written its own code to duplicate the Apple operating system's processes which would

---

72. Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d at 1251; Apple Computer, Inc. v. Formula Int'l, Inc., 725 F.2d at 525.

73. 17 U.S.C. § 101 (1976) reads, in pertinent part: " 'Literary works' are works, other than audiovisual works, expressed in words, numbers, or other verbal or numerical symbols or indicia."

74. Apple Computer, Inc. v. Franklin Computer Corp., 714 F.2d at 1251. *Cf. id.* at 1244 n.4 (illustration of the difficulty courts dealing with computer issues have in defining and limiting the entities to which they grant or deny protection). The *Franklin* court improperly defined an operating system. Of the fourteen programs at issue in that case, five were not operating system programs at all, but were user-accessible application programs—including two language compilers—that were "bundled" (that is, sold with) the actual operating system. Since the thrust of the holdings of both courts was simply that code is code, and copyrightable whether it belongs to an application or an operating system, the error is minor.

75. WEBSTER'S NINTH NEW COLLEGIATE DICTIONARY 1199 (1988).

76. *See* 714 F.2d at 1251.

77. *Id.* at 1253.

not have infringed Apple's copyright. Such copying would be illegal only if Apple's system processes were held to be copyrightable expression, in which case the decision would have been inconsistent with the Copyright Act, just the sort of error Nimmer feared. In this, at least, the *Franklin* court ruled correctly:

> If other programs can be written or created which perform the same function as an Apple's operating system program, then that program is an expression of the idea and hence copyrightable. In essence, this inquiry is no different than that made to determine whether the expression and idea have merged . . . .[78]

The court's comment regarding compatibility[79] is logically inconsistent with this statement. If a process is not copyrightable, then independent creation of a program which duplicates that process is not an infringement. Therefore, compatibility must be an element of a program's idea, not its expression. If the only way to duplicate the process is to copy the code, then the expression and idea have merged and there is no copyright. In this case, the process should be patented.

Similarly, in the case of computer languages, there may be only a limited number of ways for compilers of a language to be arranged in order to handle the set of user-entered commands and functions consistently. Even if the code per translation module is different, there will still be certain substantial similarities in the sequences in which they are called.

It must be reaffirmed that systems—including linguistic grammar systems—are not copyrightable. First, 17 U.S.C. § 102(b) forbids it.[80] Second, any attempt to reverse-engineer a system's processes may result in code so substantially similar that the expression and the idea cannot be distinguished.

IX. "STRUCTURE, SEQUENCE AND ORGANIZATION": THE *WHELAN v. JASLOW* NON-BREAKTHROUGH

Whelan Associates v. Jaslow Dental Laboratory, Inc.[81] is perhaps the most important of the computer copyright cases because it held, for the first time, that in addition to the literal elements, "structure, sequence and organization" of the program may be copyrighted. In fact, this is not really an extension of copyright principles. But it has been interpreted as a breakthrough, and subsequent decisions are using it as a platform on which to base dangerous and confusing decisions. The

78. *Id.*

79. *See supra* quote accompanying note 62.

80. "In no case does copyright protection for an original work of authorship extend to any idea, procedure, process, system . . ." 17 U.S.C. § 102(b) (1988).

81. 797 F.2d 1222 (3d Cir. 1986), *cert. denied*, 479 U.S. 1031 (1986).

Apple cases falsely hint that a system program may be copyrighted without regard to its systemic aspects. The description in *Whelan* of what may be copyrighted—"structure, sequence and organization"—also, at first reading, seems to speak to a program's systemic aspects. Berkowitz has observed:

> The third circuit's extension of copyright protection of computer programs in *Whelan Associates v. Jaslow Dental Laboratory, Inc.*, has potentially placed the copyright protection of computer programs in conflict with the patent system's protection of functional processes. The court's proposal to protect the intangible "look and feel" of a program is a difficult standard for other courts to interpret properly. At present, for lack of a better standard which would protect the interests of all parties, the courts should adhere to the well grounded standard of comparing evidence of only limited copying as proof of substantial similarity in a computer copyright infringement action.[82]

In *Ashton-Tate v. Fox*, where it is claimed a language is copyrightable, this twisting of the purpose of copyright law may come to full fruition.

The problem can be avoided by viewing the *Whelan* decision as simply an application, for the first time, of the full scope of literary copyright principles to the literary expression that is a computer program. Unlike dramas, in which a sufficiently unique and developed plot can be copyrighted, computer programs could, until *Whelan*, be copyrighted only as to their literal elements. Only the actual program code instructions were protectible as literary works.

Consider *Synercom Technology, Inc. v. University Computing Co.*,[83] which held specifically that a program's structure and sequence are not copyrightable, but are elements of the program's idea. The court asked, "if sequencing and ordering is expression, what separable idea is expressed?"[84] This question makes sense if the reader identifies "sequence and order" with the ordering of processes that is the "idea" of the system. However, it becomes irrelevant if one analyzes the organization of a computer program as one would look at the plot of a drama or novel.

The third circuit, in *Whelan*, declined to follow *Synercom*. The court held that "To the extent that *Synercom* rested on the premise that there was a difference between the copyrightability of sequence and form in the computer context and in any other context, we think

82. Comment, *Computer Software Copyright Infringement: The Second Generation*, 4 TOURO L. REV. 97, 132 (1987).

83. 462 F. Supp. 1003 (N.D. Tex. 1978).

84. *Id.* at 1013.

that it is incorrect."[85] *Whelan* referred specifically to dramatic plot cases in holding that the "structure, sequence, and organization" of a program is just as protectible as that of a literary or dramatic work.[86] Thus, we see that *Whelan* is not a breakthrough at all, but merely a recognition that if a computer program is a literary work, then all the protections available to novels and the literary aspects of dramatic works apply equally to computer programs.

However, the *Whelan* court made two important errors, not of analysis or decision, but of definition and omission. First, by holding that the "idea" of the program was simply to aid "the efficient organization of a dental laboratory"[87]—just the sort of destructively broad definition discussed earlier in this paper—the court defined the "idea" far more broadly than the idea of a literary or dramatic work would have been defined. The court's definition of the program's idea—"the efficient organization of a dental office"—was adequate for purposes of the decision at bar, but a poor descriptor for purposes of precedent.

Since the *Whelan* court refers several times to the holdings and dicta of Learned Hand[88], it may be useful to look at the method of analysis posited by Judge Hand in *Nichols v. Universal Pictures Corp.*:[89]

> But when the plagiarist does not take out a block in situ, but an abstract of the whole, decision is more troublesome. Upon any work, and especially upon a play, a great number of patterns of increasing generality will fit equally well, as more and more of the incident is left out. The last may perhaps be no more than the most general statement of what the play is about, and at times might consist only of its title; but there is a point in this *series of abstractions* where they are no longer protected, since otherwise the playwright could prevent the use of his "ideas," to which, apart from their expression, his property is never extended . . . . We did not in Dymow v. Bolton, 11 F.(2d) 690 [sic], hold that a plagiarist was never liable for stealing a plot . . . . We found the plot of the second play was too different to infringe, *because the most detailed pattern, common to both, eliminated so much from each that its content went into the public domain* . . . .[90]

Viewed from this perspective, we can analyze the idea/expression dichotomy of a computer program as a "series of abstractions," seeking to find at which level of abstraction the most detailed pattern of organization is so unique as to be copyrightable. In theory, any program duplicating the original to at least this level would infringe. Any duplication

85. Whelan Assoc. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1240, *cert. denied*, 479 U.S. 1031 (1986).

86. *Id.* at 1234.

87. *Id.* at 1240.

88. *Id.* at 1234-35.

89. 45 F.2d 119 (2d Cir. 1930).

90. *Id.* at 121 (emphasis added).

of the pattern not reaching this level of detail would be in the public domain.

The dispute in *Nichols* concerned alleged copying of incident and character. Judge Hand analyzed it as follows:

> There are but four characters common to both plays, the lovers and the fathers. The lovers are so faintly indicated as to be no more than stage properties. They are loving and fertile; that is really all that can be said of them, and anyone else is quite within his rights if he puts loving and fertile lovers in a play of his own, wherever he gets the cue. The plaintiff's Jew is quite unlike the defendant's. His obsession is his religion, on which depends such racial animosity as he has. He is affectionate, warm and patriarchal. None of these fit the defendant's Jew, who shows affection for his daughter only once . . . . Both are grotesque, extravagant and quarrelsome; both are fond of display; but these common qualities make up only a small part of their simple pictures, no more than anyone might lift if he chose. . . . A comedy based upon conflicts between Irish and Jews, into which the marriage of their children enters, is no more susceptible of copyright than the outline of Romeo and Juliet.[91]

The levels of abstraction involved in this analysis may be charted from the highest level of detail to the lowest. For the sake of clarity, we will use "Romeo and Juliet," since the reader may be somewhat more familiar with it than with "Abie's Irish Rose," the play at issue in *Nichols*:

1. "Romeo and Juliet," by Shakespeare, with all its poetry, character-based speeches, etc.
2. Detailed outline, laid out scene by scene, encounter by encounter, with detailed summaries of each character and that character's fate.
3. Somewhat detailed outline: boy and girl from feuding Italian families fall in love after boy secretly crashes a party given by girl's family. Arrogant fop from girl's family kills boy's funny close friend and kinsman. Boy kills fop. Family battle-lines are drawn. Unable to be together, boy and girl commit suicide. The character types are chosen because of the actors who are available to the author, among others, two romantic leads, an athletic comic and a perfect fop.
4. Rough outline: Boy and girl from feuding families fall in love. They try to bring the families together, but only make matters worse. Unable to be together, boy and girl do something desperate and both die.

Clearly the outline in item 4 is not copyrightable. Nor is the somewhat detailed outline sufficiently unique. Professor Chafee has written: "No doubt the line does lie somewhere between the author's idea and

91. *Id.* at 122.

the precise form in which he wrote it down. I like to say that the protection covers the 'pattern' of the work . . .—the sequence of events, and the development of the interplay of characters."[92] By this analysis, the detailed outline of item 2 is sufficient for copyright, as of course, the play itself would be, were it not already in the public domain.

This mode of analysis may be applied to the program in *Whelan* and, more importantly, to the language issue of *Ashton-Tate*:

1. The program itself, as coded (and from which the actual copy, in *Whelan*, was made).[93]
2. The technical and functional specifications of the program: flow charts, data flow diagrams, file structures and other detailed documentation from which the program could be coded in any language. These are all developed from the basic requirements of the system as posited by the ultimate user.
3. The basic requirements or "idea" of the system. These requirements, in *Whelan*, certainly included the purpose of "the efficient organization of a dental office." In addition, if the dental laboratory in question had a preexisting computer system, it might have also included the requirement of compatibility with pre-existing file formats and structures, as well as structures and programming languages, because "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."[94]
4. The *Whelan* court description: "the efficient organization of a dental office."[95]

Clearly items 1 and 2 are copyrightable. Clearly item 4 is not. The question is with regard to item 3. If the languages and file structures used are in the public domain—as are the broad character types used in the "Romeo and Juliet" analysis, as well as Judge Hand's "Abie's Irish Rose" analysis, then it seems there is no problem with regarding item 3 as "idea" rather than "expression." If they are not in the public domain, then item 3 ought only to be "idea" insofar as it does not consist of the copyrighted "expressions." Based upon our earlier analysis, we find that neither language nor basic file structure[96] are "expressions."

---

92. Chafee, *Reflections on the Law of Copyright*, 45 COLUM. L. REV. 503, 513-14 (1945).

93. Whelan Assoc. v. Jaslow Dental Laboratory, Inc., 797 F.2d 1222, 1225-27, *cert. denied*, 479 U.S. 1031 (1986).

94. *Id.* at 1236 (citation omitted).

95. *Id.* at 1240.

96. We can also use the "series of abstractions" analysis to come to the same result, with regard to file structures. Applied to the file structures at issue in *Whelan*, we find that the analysis of the court is also in line with Hand's "series of abstractions" technique:

1. File structures completely filled in with requisite information.
2. File structures, empty of information, but organized and sequenced in a way "sufficiently complex and detailed . . . [to] require certain information and or-

X. APPLICATION OF "STRUCTURE, SEQUENCE AND ORGANIZATION" TO COMPUTER LANGUAGES

Since language has "structure, sequence and organization," which, in the case of the dBASE language, is part of the larger "structure, sequence and organization" that represents the dBASE III+ program, one might ask whether the language can be copyrighted as part of that larger program. One might also question whether such a copyright would protect the language if that language were used in a substantially different environment. The court in *Walt Disney Productions v. Air Pirates*[97] held that a copyrightable element of a work, comic book characters, could be held protected even though the plaintiff had copyrighted only the comic books in which the characters appeared and not the characters themselves.

However, the holding in *Walt Disney* may not apply to computer languages. First, it is arguable that a language does not have an underlying "structure, sequence or design" because it is merely a methodology by which a programmer can create such structures of his own. Second, the key to the *Disney* holding was that the element extracted from the protected work was itself copyrightable. A language would seem to be distinguishable in that, as an idea, it is not in itself copyrightable.

XI. "STRUCTURE, SEQUENCE AND ORGANIZATION" V. "LOOK AND FEEL": THE PRESENT STATE OF CONFUSION AND A MODEL FOR USE IN ACCORD WITH TRADITIONAL LITERARY/AUDIO-VISUAL COPYRIGHT DISTINCTIONS

The second failing of the *Whelan* decision is that the court failed to explicitly limit or define the meaning of "structure, sequence and organization" to the facts of the case, *i.e.*, to a situation in which an original program and the infringing copy both accept the same data and process it in almost precisely the same way. We have shown this to be a violation of literary copyright similar to the appropriation of the detailed plot of a play. But at least one court has extended the idea of "structure, sequence and organization" to audio-visual works.[98] There is no reason to so extend the doctrine, since audio-visual copyright has

---

der that information in a particular fashion . . . [so that the] "comprehensiveness and complexity of the file structures . . . are sufficiently informative to deserve copyright protection." *Id.* at 1243.

3. A very complex cataloguing structure like the structure of Lexis or Westlaw without any entries yet made.

4. A blank form.

97. 581 F.2d 751 (9th Cir. 1978).

98. Broderbund Software, Inc. v. Unison World, Inc., 648 F. Supp. 1127 (N.D. Cal. 1986).

been developing an analogous, new doctrine of its own, "look and feel." The confusion of these two new doctrines threatens to break down the traditional distinctions between types of copyright. Taken with the possible breakdown of the line between copyright and patent due to problems courts have had distinguishing idea from expression, this leads one to fear that, in the chaos, anything requiring originality may be copyrighted, including a language.

With conviction that the confusion need not exist, the author suggests a distinction between "structure, sequence and organization" and "look and feel" akin to the line dividing literary from audio-visual copyrights.

"Look and feel" is a new legal vernacular which, along with "structure, sequence and organization," seems to be on its way to becoming a term of art. The phrase is derived from "total concept and feel," first used in *Roth Greeting Cards v. United Card Co.*[99] Although, the court in *Sid & Marty Krofft Television Productions, Inc.*[100] held that similarity between characters could be established by the total concept and feel of two productions, no case has been decided solely on the basis of "look and feel."

Initially, "look and feel" applied to audio-visual works. However, it is sometimes used more broadly to cover the entire program. Sholkoff writes: "It is the 'look and feel' or totality of all aspects of the software, when viewed as a whole, that forms the most valuable part of a computer program."[101]

Both "structure, sequence and organization" and "look and feel" are outgrowths of a line of cases that have held "substantial similarity" to be a test of whether one work infringes upon another.[102] These two analytical criteria need to be more clearly defined than they presently are.

"Structure, sequence and organization" should be confined to the literary copyright area. Literary copyrights should protect all code and documentation, which in turn are the sources of the program's organization and flow. To determine whether "structure, sequence and organization" of a work accused of infringement is "substantially similar" to a plaintiff's work, the "series of abstractions" test should be applied. The test, and very concept of "structure, sequence and organization" is largely, if not completely, objective. It should be applied by a computer-

99. 429 F.2d 1106, 1110 (9th Cir. 1970).

100. 562 F.2d 1157, 1167 (9th Cir. 1977).

101. Note, *supra* note 6, at 446.

102. *See* E.F. Johnson v. Uniden Corp., 623 F. Supp. 1485 (D.C. Minn. 1985); *see also* Williams v. Arndt, 626 F. Supp. 571 (D. Mass. 1985).

literate master, as ordinary lay juries will rarely be qualified to make such an analysis.

On the other hand, "look and feel" should be viewed as a subjective criterion that concerns art more than science, ease and elegance more than raw efficiency. "The creative choices and artistic authorship of the screen display designers can be identified and should be evaluated wholly apart from the computer code used to implement the display."[103] "Look and feel" should be viewed as the work's overall visual concept, embracing all of a program's display screens and other visual outputs. One may conceptualize "look and feel" by imagining an art gallery in which many paintings are hung. It is not just the paintings themselves, but their arrangement, which gives rise to the viewing experience of the gallery visitor. As such, "look and feel" is readily identifiable by the lay observer.

The author submits that the confusion of these two terms in both case law[104] and literature is an outgrowth of the flaws of *Whelan*. One of the most egregious subsequent errors is found in *Broderbund Software, Inc. v. Unison World, Inc.*,[105] while a more sensible approach appears in *Lotus Development Co. v. Paperback Software Int'l*.[106]

XII. *BRODERBUND v. UNISON WORLD*: "STRUCTURE, SEQUENCE AND DESIGN" RUN AMOK

The *Broderbund* case interpreted *Whelan* to stand for "the proposition that copyright protection is not limited to the literal aspects of a computer program, but rather that it extends to the overall structure of a program, including its audiovisual displays."[107] This is not what *Whelan* held and at least two commentators recognize this.[108] The

103. United States Copyright Office Public Hearing on Registration and Deposit of Computer Screen Displays, Written Submission of Apple Computer, Inc., Comment Letter RM 87-4, filed Sept. 4, 1987.

104. *See* Lotus Dev. Corp. v. Paperback Software Int'l, 740 F. Supp. 37 (D. Mass. 1990); Apple Computer, Inc. v. Microsoft Corp., 717 F. Supp. 1428 (N.D. Cal. 1989); Apple Computer, Inc. v. Microsoft Corp., 709 F. Supp. 925 (N.D. Cal. 1989); Johnson Controls, Inc. v. Phoenix Control Sys., 886 F.2d 1173 (9th Cir. 1989).

105. 648 F. Supp. 1127 (N.D. Cal. 1986).

106. 740 F. Supp. 37 (D. Mass. 1990).

107. 648 F. Supp. at 1133.

108. Berkowitz, *Computer Software Copyright Infringement: The Second Generation*, 4 TOURO L. REV. 97, 130 (1987) ("While it is true that *Whelan* extended copyright protection of computer programs beyond the literal code, it did not extend the protection of the Program to include the structure of the audiovisual displays."); Case Comment, *Broderbund Software, Inc. v. Unison World, Inc.*, 7 COMPUTER/L.J. 535, 538 (1987). "Purporting to base its decision on *Whelan*, the court in *Broderbund*, broke new ground by extending protection for computer programs as *literary works*, to 'the overall structure of a program, including its audiovisual displays.'" *Id.* at 535 (quoting *Broderbund*, 648 F.

Broderbund court held the program's audiovisual displays to be copyrightable despite the failure of the plaintiff to register any claim to copyright those displays. The court's logic seems to have been that because the displays exist as elements of the program's structure, sequence and organization, any copyright of the program covers all those elements. Therefore, audio-visual displays are covered by a literary copyright. Tito observes that "insofar as the [*Broderbund*] court's holding extends the *Whelan* rule regarding the protected portions of a computer program to screen displays, without distinguishing the separate copyright for the audiovisual works, the case was wrongly decided."[109]

Yet, the reason that *Broderbund* occurred is that *Whelan* was seen as a breakthrough. If *Whelan* is viewed as it really is—a mere application of the full scope of copyright protection to an area already protected under traditional doctrine—we see that traditional copyright law holds true. In short, works that have historically been in the public domain ought to remain in the public domain, and thus, computer languages should not be subject to copyright.

XIII. *LOTUS DEVELOPMENT CORP. V. PAPERBACK SOFTWARE INT'L*: A PURELY STRUCTURAL APPROACH

Lotus Development Corp. v. Paperback Software Int'l[110] held that the user interface of Lotus 1-2-3, one of the dominant spread sheet programs in the industry, is copyrighted. The interface of Lotus 1-2-3 combines structural and visual elements. Applying the *Nichols* "series of abstractions" test to the underlying structure of the screen designs, Judge Keeton found that the characteristic look of Lotus 1-2-3 is not copyrightable because it is too common to most spreadsheet programs. What is unique to plaintiff's product is the "particular expression of a menu structure."[111]

The "command structure taken as a whole" was found copyrightable because "[i]f particular characteristics not distinctive individually have been brought together in a way that makes the 'whole' a distinctive expression of an idea . . . then the 'whole' may be copyrightable."[112] The court then noted that "[t]he statutory provisions regarding 'compilation,' 17 U.S.C. §§ 101, 103, are not essential to this conclusion, but do reinforce it."[113] In fact, the Lotus 1-2-3 command structure is very simi-

---

Supp. at 1132). "*Whelan* did not rule on the copyrightability of the overall structure, sequence and organization of the visual screens produced by the computer program." *Id.* at 538.

109. Case Comment, *supra* note 108, at 541-42.

110. 740 F. Supp. 37 (D. Mass. 1990).

111. *Id.* at 67.

112. *Id.*

113. *Id.*

lar to a compilation. Each command term as presented to the user occupies two particular places: first, in the ordered hierarchy of screen commands, and second, in the coordinate matrix of screen positions.

It is this ordering of commands—the way in which the commands and the programs they invoke are made accessible to the user—that the court found copyrightable: "the menu structure, taken as a whole—including the choice of command terms, the structure and order of those terms, their presentation on the screen, and the long prompts."[114]

The *Lotus* court's use of the *Whelan* "structure, sequence and organization" analysis, interpreted in light of *Nichols*' "series of abstractions" test, is consistent with the views propounded in this paper.

Equally consistent is the *Lotus* court's refusal to adopt "look and feel" as an analytical technique in this case. The court observed that courts have used look and feel "not . . . as an aid to the court in determining which nonliteral elements were copyrightable and why. Rather, these courts used the concept . . . apparently assuming copyrightability . . . in applying the substantial similarity test to determine whether forbidden copying had occurred."[115]

This observation caused Judge Keeton to conclude that " 'look and feel' is a conclusion, and the usefulness and applicability of a precedent depends on the *reasons* the conclusion was reached in a particular context, not on the *conclusion* itself."[116] Therefore, Judge Keeton wrote, "I have not found the 'look and feel' concept, standing alone, to be significantly helpful in distinguishing between nonliteral elements of a computer program that are copyrightable and those that are not."[117]

The author does not find Judge Keeton's conclusion completely convincing because it rests on the premise that what makes a visual work original can always be structurally analyzed and articulated. However, certain legal concepts cannot be articulated—for example, obscenity.[118] The author suggests that originality, particularly in visual works, is similarly undefinable. The originality of a painting is not really subject to structural or legal analysis. In comparing paintings for copyright infringement, ultimately the trier of fact can only look, apply the substantial similarity test, and draw a conclusion.

Be that as it may, the *Lotus* court was correct in eliminating "look

---

114. *Id.* at 68. "Long prompts" are sentence-long explanations of particular commands that appear on the screen when the user highlights that command.

115. *Id.* at 63.

116. *Id.* (emphasis in original).

117. *Id.* at 62.

118. Despite the standards for determining obscenity created by the United States Supreme Court in *Miller v. California*, 413 U.S. 15 (1973), probably the most famous legal quotation on the subject is Justice Potter Stewart's "I cannot define it, but I know it when I see it."

and feel" from consideration in the Lotus 1-2-3 case. Once Judge Keeton had determined by use of the *Nichols* that most spread sheets look fairly similar on the screen and that the copyrightable original material was the command structure, he limited the scope of his inquiry to something completely separate from artistic appearance of the screens. He could have just as easily made a determination of the command structure's copyrightability from an hierarchical chart listing the names or numbers of screens, the screen coordinates at which each command appears, and a description of the routine the command invokes. Like any sufficiently and original outline or compilation, this is copyrightable. "Look and feel," as this paper suggests the term be used, was not really an issue in this case.

Lotus should therefore not be read as eliminating "look and feel" from consideration in computer copyright cases. As new generations of software with even more sophisticated graphics enter the marketplace, "look and feel" will be increasingly important. As computers become more capable of graphic expression, programmers will—and already do—create new and distinctive looks for their programs that are strikingly original. These new looks may provide graphic metaphors that add to the understanding and usefulness of the programs and in doing so, may reflect the program's underlying structure. But they are not themselves elements of that structure. Rather, they are a gloss on it. Determinations of originality and infringement will have to be made as they are made for any visual work.

XIV. "LOOK AND FEEL" APPLIED TO COMPUTER LANGUAGES

It remains to be asked whether a computer language has a "look and feel" that rises to the level of copyrightable expression.

Generally, computer languages are communicated visually; people do not speak COBOL, they write it. Further, programs written in a language have a characteristic appearance. For example, COBOL programs begin with statements organized into Environment and Data divisions. Programs written in C usually begin with pre-processor instructions that are prefixed with the character "#"; C program statements nearly all end with the character ";" and C procedures are all designated "<PROCEDURE NAME> ()."

These instructions are no more expression than the characteristic look of English as opposed to German. It is not art that gives a language idea appearance; it is convention. The medium is *not* the expression, as evinced by the many cases—including *Whelan v. Jaslow*—that have held mere translation from one language to another to be infringe-

ment. Thus "look and feel" is not an issue that applies to computer languages.

XV. THE COPYRIGHT OFFICE POLICY OF 1988: CREATURE OF NEITHER STATUTE NOR CASE LAW

Two years after the 1988 *Broderbund* decision, the Copyright Office announced that it would allow only one copyright registration per computer program.[119] The Copyright Office's current policy is to assess which of the copyright classes predominates in a particular program—literary, audio-visual, etc.—and to register the whole under that one class. This extension of the *Broderbund* erosion of traditional copyright class boundaries can cause confusion, as it is contrary to a number of decisions that have held computer audio-visual displays to be separately copyrightable as audio-visual works.[120] Copyright Office policy is by no means the final word, although a court will rarely substitute its own judgment for that of an administrative agency unless the agency's policy is inconsistent with statutory language or congressional intent.[121] However, it is equally clear that the administrative bulletin is not binding on the court.[122]

Statutory language regarding the manner in which computer programs should be copyrighted is unclear. But, however one interprets the statute, current Copyright Office policy is poor. As 17 U.S.C. § 101 clearly evinces congressional intent that literary, audio-visual and other works be copyrighted separately, the Copyright Office policy should be overturned. Additionally, § 101 also defines computer programs as literary works, making no mention of other aspects of a program that may be copyrighted, such as audio-visual works. In either event, a policy of copyrighting all of a program's elements under the rubric of whichever aspect predominates is inconsistent with statutory language and Congressional intent, and should be struck down.

XVI. DO WE NEED NEW COPYRIGHT LEGISLATION FOR COMPUTERS?

The current trend in decisions regarding computer copyright is

119. *Copyright Office Notice on Computer Screen Registration*, [May-Oct.] Pat. Trademark & Copyright J. (BNA) Vol. 36, No. 884, at 152, 153-55 (June 9, 1988).

120. Williams Elecs., Inc. v. Arctic Int'l, Inc., 685 F.2d 870 (3d Cir. 1982); Stern Elecs., Inc. v. Kaufman, 669 F.2d 852 (2d Cir. 1982); Midway Mfg. Co. v. Strohon, 564 F. Supp. 741 (N.D. Ill. 1983); Atari, Inc. v. North Am. Philips Consumer Elecs. Corp., 672 F.2d 607 (7th Cir. 1982), *cert. denied*, 459 U.S. 880 (1982); M. Kramer Mfg. Co., Inc. v. Andrews, 783 F.2d 421 (4th Cir. 1986).

121. C.B.S. Imports Corp. v. United States, 450 F. Supp. 724, 727 (Cust. Ct. 1978); *see also* Suwanee S.S. Co. v. United States, 435 F. Supp. 389 (Cust. Ct. 1977).

122. C.B.S. Imports, 450 F. Supp. at 727.

clearly cause for concern. Sholkoff observes that "because the current standard is unclear, judicial determinations risk being arbitrary."[123]

Logically, two alternative approaches toward resolution of the problem exist.

The first is for the judicial system to garner the knowledge to interpret current law sensibly. This might be done in either of two ways: a pool of computer-expert masters might be established under the judicial system to stay current with developing technology, keep judges informed and act as expert witnesses. A more traditional and perhaps better solution would be to establish a small administrative agency charged with holding adjudicatory hearings under computer-expert hearing officers. This agency could work in conjunction with the Copyright and Patent offices to clarify issues and set policy.

The second, propounded by Sholkoff, among others, is that Congress pass laws to create a special statutory framework within which computer copyright issues can be analyzed. This approach is based upon the premise that current copyright law is obsolete.

> Current copyright protection for computer software remains unclear, inconsistent, and potentially under-protective and damaging to an industry which thrives on innovation. With standards that are questionable or confusing, and that fail to protect the financially important aspects of the work, *current* copyright law is not the most efficient method of protecting a software developer's intellectual creations.[124]

The approach of legislating currency into copyright law has history and *stare decisis* behind it. "Repeatedly, as new developments have occurred in this country, it has been the Congress that has fashioned the new rules that new technology made necessary. . . it [is] settled that the protection given to Copyright is wholly statutory."[125] This approach shares with judge-made computer law one fundamental flaw: neither judicial nor legislative knowledge can possibly evolve as quickly as the technology that must be governed. To remain current, Congress would have to create a standing committee or sub-committee for that purpose.

XVII. SUMMARY AND CONCLUSION

Recent decisions in the third and ninth circuits have raised questions regarding the potential application of copyright law to computer languages. *Apple Computer, Inc. v. Franklin Computer Corp.* and *Apple Computer, Inc. v. Formula Int'l, Inc.* raised the threat that copyright might invade the realm of patent, holding that operating systems could be copyrighted without explaining fully that only the literal code can be

123. Note, *supra* note 6, at 445.

124. *Id.* (emphasis in original).

125. Sony Corp. v. Universal City Studios, 464 U.S. 417, 430-31 (1984).

copyrighted as an "expression" of the systemic "idea." The *Apple* courts made the problem worse by defining the "idea" of a system too broadly and excluding compatibility—a concept vital to the computer industry—from the possible components of an "idea." *Whelan v. Jaslow* repeated Franklin's error by defining the idea of a program too broadly. Further, the *Whelan* court failed to define the limits of its holding that a program's "structure, sequence and organization" are subject to copyright. Lack of firm definitions for "structure, sequence and organization" and "look and feel" further exacerbates matters. *Broderbund v. Unison World* fanned the flames by holding that a literary copyright could cover an audiovisual display, thus breaking down the traditional categories of copyright law. Lack of clear statutory language, followed by a 1988 Copyright Office policy decision that appears misconceived by any statutory interpretation, have added to the chaos.

In this uncertain environment, some danger exists that things never before deemed copyrightable will be copyrighted. This is precisely the danger that now looms in the northern district of California, as Ashton-Tate attempts to copyright the dBASE computer language. The effects of this effort upon the programming community and businesses that depend upon data processing could be disastrous.

The "idea/expression" danger can be avoided by defining the "idea" of a program with more specificity, by regarding compatibility as one of the potential elements of an idea, and by interpreting the *Apple* and *Whelan* decisions properly as cases that concerned literary copyright. The *Apple* decisions focused on the copyrightability of operating system code, not the conceptual system itself. *Whelan* analogized the organization of a system to the plot of a drama or novel, a mode of analysis which can be useful if one applies the "series of abstractions" analysis drawn from *Nichols v. Universal Pictures.*

By continuing to allow separate registration of the disparate elements of computer programs—literary, audio-visual, and possibly phono-record—triers of fact and law can better organize the issues before them and keep analyses relatively simple. Both courts and people in the industry will have a clear idea of just what is protected by each copyright. Use of the recently developed tests of "structure, sequence and organization" and "look and feel" should apply to the literary and audio-visual areas of copyright respectively. The goal of each test is to determine whether there is "substantial similarity" and thus infringement. The former test should be applied on an objective basis by computer-trained masters utilizing Learned Hand's "sequence of abstractions" technique. "Look and feel," however, should be a subjective test exercisable by lay juries.

Nothing suggested here requires substantial revision of existing copyright law. The approaches articulated in this paper require only

clarification of the application of present law, and the disposal of misguided precedents, which have recently emerged as generalist judges grope to deal with an area requiring specialized knowledge. Such substantial new legislation as is advocated by Sholkoff is probably not required, although it may be useful as an interim clarification. More useful legislation might establish a small administrative agency under the Copyright Office for the purpose of keeping abreast with technology and adjudication through administrative hearings under computer-expert hearing officers of technically difficult issues. These opinions could then be incorporated into the complete cases before the court (which may also contain issues of fact that can reasonably be settled by lay juries).

Traditional copyright law can function well in the computer field, if applied with understanding of the field's technology and industry. The temptation to proliferate new statutes should be avoided.