235 -C

SNELL & WILMER LLP
Alan L. Sullivan (3152)
Todd M. Shaughnessy (6651)
15 West South Temple
Gateway Tower West
Salt Lake City, Utah 84101-1004
Telephone: (801) 257-1900
Facsimile: (801) 257-1800

CRAVATH, SWAINE & MOORE LLP
Evan R. Chesler (admitted pro hac vice)
David R. Marriott (7572)
Worldwide Plaza
825 Eighth Avenue
New York, New York 10019
Telephone: (212) 474-1000
Facsimile: (212) 474-3700

Attorneys for Defendant/Counterclaim-Plaintiff
International Business Machines Corporation

IN THE UNITED STATES DISTRICT COURT

FOR THE DISTRICT OF UTAH

THE SCO GROUP, INC.,

Plaintiff/Counterclaim-Defendant,

-against-

INTERNATIONAL BUSINESS
MACHINES CORPORATION,

Defendant/Counterclaim-Plaintiff.

DECLARATION OF RANDALL DAVIS

Civil No. 2:03CV-0294 DAK

Honorable Dale A. Kimball

Magistrate Judge Brooke C. Wells

I. INTRODUCTION

1. My name is Randall Davis. I am a Professor of Computer Science at the Massachusetts Institute of Technology. Exhibit I contains a resume providing details of my technical background and experience. I received my undergraduate degree from Dartmouth, graduating summa cum laude, Phi Beta Kappa in 1970, and a Ph.D. from Stanford University in artificial intelligence in 1976. I came to MIT in 1978, served for five years as Associate Director of the MIT Artificial Intelligence Laboratory, and currently serve as a Research Director in the newly formed MIT Computer Science and Artificial Intelligence Laboratory.

2. I have published some 50 articles on issues related to artificial intelligence and have served on several editorial boards, including *Artificial Intelligence*, *AI in Engineering,* and the MIT Press series in AI. I am a co-author of *Knowledge-Based Systems in AI.*

3. In recognition of my research in artificial intelligence, I was selected in 1984 as one of America's top 100 scientists under the age of 40 by *Science Digest.* In 1986 I received the *AI Award* from the Boston Computer Society for contributions to the field. In 1990 I was named a Founding Fellow of the American Association for AI and in 1995 was elected to a two-year term as President of the Association. From 1995–1998 I served on the Scientific Advisory Board of the U. S. Air Force.

4. In addition to my work with artificial intelligence, I have also been active in the area of intellectual property and software. Among other things, I have served as a member of the Advisory Board to the US Congressional Office of Technology Assessment study on software and intellectual property, published in 1992 as *Finding a*

Balance: Computer Software, Intellectual Property, and the Challenge of Technological Change. I have published a number of articles on the topic, including co-authoring an article in the *Columbia Law Review* in 1994 entitled "A Manifesto Concerning Legal Protection of Computer Programs" and an article in the *Software Law Journal* in 1992 entitled "The Nature of Software and its Consequences for Establishing and Evaluating Similarity."

5. In 1990 I served as expert to the Court (Eastern District of NY) in *Computer Associates v. Altai*, a software copyright infringement case whose decision was upheld by the Appeals Court for the 2nd Circuit in June 1992, resulting in the articulation of the abstraction, filtration, comparison test for software. I have also been retained by the Department of Justice in its investigation of the INSLAW matter. In 1992 (and later in 1995) my task in that engagement was to investigate alleged copyright theft and subsequent cover-up by the Federal Bureau of Investigation, the National Security Agency, the Drug Enforcement Agency, the United States Customs Service, and the Defense Intelligence Agency.

6. From 1998-2000 I served as the chairman of the National Academy of Sciences study on intellectual property rights and the emerging information infrastructure entitled *The Digital Dilemma: Intellectual Property in the Information Age,* published by the National Academy Press in February, 2000.

7. I have been retained as an expert in over thirty cases dealing with misappropriation of intellectual property, such as the allegations raised in this case. I have been retained by plaintiffs who have asked me to investigate violations of intellectual property, by defendants who have asked me to investigate allegations made

against them, and by both sides to serve as the sole arbiter of a binding arbitration. A list of cases in which I have been involved is attached as Exhibit II.

8. I have been retained by counsel for IBM in this lawsuit and am being compensated at a rate of $550 per hour.

II. THE TASK

9. I have been asked to examine the question of whether the lines of source code in the 98 files in Table I (the "IBM Code") are modifications of, or derivative works based on, any source code in any of the 21 versions of Unix System V listed in Table II (the "Unix System V Code").

10. I have been instructed by counsel that one work is a "derivative work" of another under federal copyright law if it incorporates in some form a portion of the preexisting work and is substantially similar to the preexisting work. In my understanding, and as I use the term in my analysis, a "modification" based on a preexisting work must also incorporate in some form a portion of the preexisting work, else there would be no basis for calling it a modification.

11. In performing my analysis, I have therefore undertaken to determine whether the IBM Code incorporates any portion of source code contained in the Unix System V Code or is any other manner similar to such Unix System V Code.

Table I: Files and Lines of Code Identified By SCO

AIX 9922A_43NIA Files	
File Name	**Lines Identified By SCO**
kernel/sys/IA64/bootrecord.h	64-170
kernel/sys/hd_psn.h	32
usr/include/jfs/inode.h	16-37, 39-40, 62-66, 72-76, 83-158, 161-66, 172-80, 199-205
usr/include/liblvm.h	234-250, 252-72, 289-307, 316-63
usr/include/lvm.h	26-35
usr/include/lvmrec.h	24-92
kernel/sys/vnode.h	109-33
kernel/sys/vgsa.h	37, 56-73
Dynix 4.6.1 Files	
File Name	**Lines Identified By SCO**
kernel/os/kern_clock.c	2028-59
kernel/os/kma_defer.c	191-353, 370-427, 550-582, 603-703
kernel/sys/kma_defer.h	46-52, 95-119, 129-32, 140
kernel/i386/locore.s	1487-97
kernel/i386/plocal.h	1517-37
kernel/os/rclock.c	303-17, 383-613, 616-1825
kernel/sys/rclock.h	175-228, 238-41, 243-423
kernel/i386/startup.c	2054
kernel/i386/trap.c	1554-63
kernel/os/vfs_dio.c	No lines identified
JFS Files	
File Name	**Lines Identified By SCO**
include/linux/jfs/ref/jfs_aixisms.h	26-27, 32, 62, 193, 227, 248
include/linux/jfs/ref/jfs_dirent.h	55
include/linux/jfs/ref/jfs_inode.h	76-77, 81, 95, 97, 192-233, 343-425
include/linux/jfs/ref/jfs_os2.h	33-34
include/linux/jfs/ref/jfs_dasdlim.h	No lines identified
include/linux/jfs/ref/jfs_dinode.h	35-49, 53-200
include/linux/jfs/ref/jfs_lock.h	72-119, 338-391, 395-406
include/linux/jfs/ref/jfs_superblock.h	19-105
include/linux/jfs/ref/jfs_btree.h	19-113, 115-143
include/linux/jfs/ref/jfs_bufmgr.h	30-33, 37-49, 123-141, 274-279
include/linux/jfs/ref/jfs_cachemgr.h	71-108, 371-388
include/linux/jfs/ref/jfs_chkdsk.h	No lines identified
include/linux/jfs/ref/jfs_clrbblks.h	24-48, 52-60
include/linux/jfs/ref/jfs_debug.h	28-30, 81-93, 96-106, 117-134, 137-142, 146-168
include/linux/jfs/ref/jfs_defragfs.h	20-56
include/linux/jfs/ref/jfs_dmap.h	22-272, 276-324
include/linux/jfs/ref/jfs_dtree.h	25-79, 88-210, 233-287, 312-323
include/linux/jfs/ref/jfs_extendfs.h	19-29, 32-39
include/linux/jfs/ref/jfs_filsys.h	76-103, 167-172, 230-256, 266-277, 279-321
include/linux/jfs/ref/jfs_imap.h	19-168
include/linux/jfs/ref/jfs_io.h	No lines identified
include/linux/jfs/ref/jfs_logmgr.h	34-506, 540-577
include/linux/jfs/ref/jfs_proto.h	58-62, 117-128
include/linux/jfs/ref/jfs_txnmgr.h	25-251, 255-345

include/linux/jfs/ref/jfs_types.h	100-223, 299-582
include/linux/jfs/ref/jfs_util.h	38-62
include/linux/jfs/ref/jfs_xtree.h	24-131, 139-212
fs/jfs/ref/jfs_dio.c	333
fs/jfs/ref/jfs_logmgr.c	27-67, 113-132, 165-781, 1052-1607, 1623-3211
fs/jfs/ref/jfs_bufmgr.c	289-311, 364-441, 557-649, 682-917, 1270-1468, 1691-2016, 2102-2194
fs/jfs/ref/jfs_cachemgr.c	No lines identified
fs/jfs/ref/jfs_dnlc.c	55-89, 140-200, 212-224, 251-322, 325-338, 402-451, 485-573, 685-713
fs/jfs/ref/jfs_dtree.c	No lines identified
fs/jfs/ref/jfs_ifs.c	No lines identified
fs/jfs/ref/jfs_init1.c	No lines identified
fs/jfs/ref/jfs_inode.c	312-350, 390-463, 483-510
fs/jfs/ref/jfs_link.c	33-152
fs/jfs/ref/jfs_mknod.c	No lines identified
fs/jfs/ref/jfs_readdir.c	38-113
fs/jfs/ref/jfs_readlink.c	26-110
fs/jfs/ref/jfs_statfs.c	23-139
fs/jfs/ref/jfs_symlink.c	23-204
fs/jfs/ref/jfs_txnmgr.c	26-89, 122-132, 155-351, 380-414, 463-482, 531-661, 677-682, 710-767, 806-1153, 1162-1182, 1194-1246, 1293-1298, 1318-1539, 1577-1761, 1796-1856, 1883-1910, 1922-2097, 2115-2151, 2219-2321, 2350-2674, 2822-2845, 2983-3003
fs/jfs/ref/selector.c	No lines identified
fs/jfs/ref/jfs_create.c	41-121, 127-135, 153-169, 193-223, 233-239, 241-264
fs/jfs/ref/jfs_defragfs.c	33-75, 84-89, 108-111, 119-264
fs/jfs/ref/jfs_dmap.c	43-4475
fs/jfs/ref/jfs_extendfs.c	43-153, 185-249, 293-579
fs/jfs/ref/jfs_fsync.c	32-84
fs/jfs/ref/jfs_ftruncate.c	37-129, 143, 156-170, 230-341
fs/jfs/ref/jfs_getattr.c	33-124
fs/jfs/ref/jfs_hold.c	33-63
fs/jfs/ref/jfs_imap.c	27-665, 680-2855, 2876-2893, 2904-2990
fs/jfs/ref/jfs_lookup.c	37-179
fs/jfs/ref/jfs_mkdir.c	37-111, 130-213, 222-264, 322-345
fs/jfs/ref/jfs_mount.c	31-188, 198-215, 229-785
fs/jfs/ref/jfs_open.c	37-98, 117-126, 218-277, 292-312
fs/jfs/ref/jfs_rele.c	31-64
fs/jfs/ref/jfs_remove.c	36-145, 157-464
fs/jfs/ref/jfs_rename.c	36-222, 246-313, 390-526, 577-651, 760-791
fs/jfs/ref/jfs_rmdir.c	36-125, 137-156, 188-193
fs/jfs/ref/jfs_umount.c	45-182, 198-307, 318-322
fs/jfs/ref/jfs_util.c	49-120, 133-163, 175-230, 300-425
Linux 2.6.5 Files	
File Name	**Lines Identified By SCO**
arch/i386/kernel/srat.c	1-450
arch/i386/kernel/numaq.c	1-112
arch/i386/mach-es7000/topology.c	35-49

arch/i386/mach-default/topology.c	35-49
arch/i386/mm/discontig.c	1-434
arch/i386/pci/numa.c	1-129
arch/ppc64/kernel/smp.c	733-754, 783
arch/ppc64/mm/numa.c	1-374
include/asm-i386/topology.h	1-85
include/asm-i386/mmzone.h	1-154
include/asm-i386/numaq.h	1-164
include/asm-ppc64/mmzone.h	1-95
include/asm-ppc64/topology.h	1-49
include/linux/mmzone.h	350-62
include/linux/numa.h	1-16
kernel/sched.c	44, 212-13, 239-72, 1002-1126, 1390-1401, 1407, 1421-22, 1432-33
mm/page_alloc.c	[724]-726, 737-738, 827-35, 889-92, 983-92, 1137-1238

The 8 AIX files are listed in SCO's Revised Supplemental Response to IBM's First and Second Set of Interrogatories, dated 12 January 2004; SCO identified a total of 468 lines.

The 10 Dynix files are listed in SCO's Revised Supplemental Response and Exhibit D of the letter from B. Hatch to T. Shaughnessy of 19 April 2004; SCO identified a total of 2,162 lines.

The 17 Linux 2.6.5 files are listed in Exhibit C of the letter of 19 April 2004; SCO identified 2,437 lines. SCO's letter identifies lines 794 to 726 of mm/page_alloc.c, which appears to be a typographical error.

The 63 JFS files are listed (with some repetition) in Tables H and I of SCO's Revised Supplemental Response, and Exhibit B of the letter of 19 April 2004; SCO identified 21,692 lines.

Grand total: 26,759 lines identified by SCO in 98 files.

Table II: Versions of Unix System V used in this comparison

VERSION OF UNIX SYSTEM V	NUMBER OF FILES	TOTAL LINES OF SOURCE MATERIAL
System V version 1.0	1,400	347,099
System V version 1.1	1,253	208,086
System V version 2.0	4,372	896,148
System V version 2.0_3B20	3,256	577,484
System V version 2.2.0_3B15	4,530	985,196
System V version 2.1.0V1_VAX	2,401	477,251
System V version 2.1_3	1,280	360,281
System V 3.0	4,781	818,403
System V 3.1	3,849	631,382
System V 3.2	4,369	702,328
System V 3.2 for 386	4,810	991,212
System V 4.0 for 386	9,472	1,853,434
System V 4.0v2 for 386	11,771	2,367,995
System V 4.0v3 for 386	9,466	1,957,328
System V 4.0 MP	12,649	2,876,245
System V 4.1	21,798	3,567,560
System V 4.1 ES	11,902	2,595,549
System V 4.2 ES-MP	21,577	5,148,564
UnixWare 1.1	28,869	6,493,708
UnixWare 2.1	44,340	10,182,665
UnixWare 7.1.3	70,397	23,759,651
TOTALS	278,542	67,797,569

12. The conclusions set out here are not intended as, and do not represent, legal conclusions. My conclusions are instead based upon my understanding of the law with respect to the appropriate process and procedures for making a judgment of substantial similarity.

13. I understand the accepted process for determining substantial similarity to call for abstraction, filtration, and comparison, although when modest amounts of code are involved, the abstraction step may not be required. I understand filtration to involve the removal of at least the following elements: ideas, purposes, functions, procedures, processes, systems, methods of operation, facts, unoriginal elements (e.g., those in the public domain), expression that is inseparable from or merged with ideas or processes,

and expressions that are standard, stock, or common to a particular topic, or that necessarily follow from a common theme or setting.

14. I understand further that with respect to computer programs in particular, the scènes à faire doctrine:

> excludes from protection those elements of a program that have been dictated by external factors. In the area of computer programs these external factors may include: hardware standards and mechanical specifications, software standards and compatibility requirements, computer manufacturer design standards, target industry practices and demands, and computer industry programming practices.
>
> *Gates Rubber v Bando*, all citations omitted

15. The opinions I report here are based on the documents I have reviewed (a list is given in Exhibit III), and on my knowledge, background, and experience in the field of computer science. I am continuing work on this and reserve the right to augment my findings as additional information becomes available to me.

III. SUMMARY OF FINDINGS

16. Despite an extensive review, I could find no source code in any of the IBM Code that incorporates any portion of the source code contained in the Unix System V Code or is in any other manner similar to such source code. Accordingly, the IBM Code cannot be said, in my opinion, to be a modification or a derivative work based on the Unix System V Code.

17. As explained in detail below, I used two programs, called COMPARATOR and SIM, to help automate the process. COMPARATOR looks for lines of text that are literally or nearly literally identical, while SIM looks for code that is syntactically the same.

18. I used both programs to compare all 26,759 lines of the IBM Code identified by SCO against all 67,797,569 lines in the Unix System V Code.

19. I believe that the comparisons I performed using these tools are conservative and hence resulted in more potential matches than might otherwise be found using a less conservative approach.

20. These comparisons required on the order of 10 hours of computation time on a dual 3 GHz Xeon processor system with 2GB of RAM. This is a high-end workstation, routinely and easily available off the shelf from commercial vendors such as Dell.

21. COMPARATOR reported 15 potential hits. I reviewed each of these potential hits in detail and determined them not to be true matches of copied code, but rather coincidental matches of common terms in the C programming language. (Paragraphs 27-30 below discuss coincidental matches in COMPARATOR.)

22. SIM did not report any potential hits.

IV. METHODOLOGY

23. I was asked to analyze the specific AIX and Dynix files and lines of code cited by SCO in their filings (and listed in Table I). In instances where SCO failed to identify any specific AIX and Dynix code upon which code in Linux is allegedly based, I was asked to analyze the Linux files and lines of code cited by SCO (and listed in Table I). Finally, I was asked to analyze the JFS files and lines of code cited by SCO (and listed in Table I), even though SCO did not identify any corresponding AIX, Dynix, or Linux code for such files. All of this IBM Code in Table I was compared to all of the Unix System Code in Table II to determine if the IBM Code contains any portion of the Unix

System V Code or is in any other manner similar to any portion of the Unix System V Code.[1]

24. For purposes of my review, I did not first apply the "abstraction" and "filtration" analyses to the Unix System V Code. Instead, to be conservative, I assumed that all of the Unix System V code was in fact protectable (although I do not believe all of such code in fact to be protectable) and proceeded to compare all of the Unix System V Code with all of the IBM Code to see if there were any true matches of copied code in the first place. To the extent necessary, I then applied the "filtration" analysis to the reportedly matching code to determine if such code was in fact protectable.

25. In doing my analysis I used two programs, employing two different algorithms, to detect material in the IBM Code that might contain, or be similar to, material in the Unix System V Code. The first, called COMPARATOR [1], is designed to find sequences of lines in two different files that are literally, or nearly-literally the same. The second program, SIM [2], is designed to detect non-literal similarities at the level of syntactic structure.

26. Both programs take two lists of files and compare every line in the first set of files against every line in the second set, and report every match they find. Each match

[1] In addition to the analysis reviewed herein, I also manually reviewed the following Linux code cited in the 7 July 2004 Declaration of Sandeep Gupta: ipc/util.c (lines 119-52) and kernel/futex.c (159, 178, 187, 188-91, 456, 489, 495, 298-300, 302-08). This review could be carried out manually because Mr. Gupta had specified specific lines that were alleged to be similar. There was thus no need to run the comparison tools, which are designed to find matches. I compared the lines of Linux code identified by Mr. Gupta with the specific lines of System V 4.2 ES-MP code that Mr. Gupta claims matches the Linux code. As is obvious upon review (and may be obvious even to a non-technical reviewer), the Linux code cited by Mr. Gupta does not contain any of, and is not in any way similar to, the Unix code that he cites. The code is entirely different. In my opinion, therefore, the code cited by Mr. Gupta for ipc/util.c and kernel/futex.c cannot be considered modifications or derivative works of Unix System V.

consists of a file name and line numbers indicating places in each file that the program believes to be similar.

27. The first step of my methodology was to compare all the IBM Code against all the Unix System V Code. At my direction, one of my assistants ran the IBM Code and the Unix System V Code through the COMPARATOR and SIM programs to generate a set of initial matches.

28. Next, I manually reviewed all of the matches reported by the comparison tools. All of the matches that I reviewed were not true matches of copied code. As a result, I did not have to perform any "filtration" analysis on the code.

29. The matches reported by COMPARATOR between the IBM Code and the Unix System V Code consisted of coincidental matches of terminology in the C programming language, and thus not true matches. These coincidental matches arise in much the same way that, if we compared the entire text of two novels (e.g., *War and Peace* and *A Tale of Two Cities*), we would surely find that they both contain the phrase "and then they" somewhere within them. Such coincidences of common language are no more indicative of copying in English than the corresponding matches of programming text are in the large bodies of code examined here.

30. The box below shows one of the reported matches from the lines of code cited by SCO. COMPARATOR reported a match between lines 588-591 in rclock.c and lines 1665-1667 from System V UW1.1 /src/i386at/uts/io/target/sdi.c:

Lines 588-591 from rclock.c	Lines 1665-1667 from sdi.c

`#endif /* RCLOCK_PROF */`
 	
`return;`
`}`
	`#endif`
	`return;`
	`}`

The two "words"– `endif` and `return` – that appear in the two files are so common in code written in the C language that finding them together like this is purely an accident, of no significance in detecting copying. In particular, the code from each file above simply signifies the ending of a routine; it is as if we had found two bodies of unrelated English text that each happened to conclude with the words "the end".

31. Note that there are 4 lines cited from the IBM file but only three from the Unix file. This is because COMPARATOR ignores blank lines (the second line in the IBM code excerpt is blank), which keeps it from being misled by this sort of immaterial variation. COMPARATOR also ignores single line comments (i.e., a line of text that start with "/*"), hence its finding that the first line in each of these excerpts is similar.[2] This is another way in which it is not misled by immaterial variation. These are two of the factors why COMPARATOR is described above by saying that it "looks for lines of text that are literally or *nearly literally* identical".

32. All of the potential hits reported by COMPARATOR were of the type discussed in paragraphs 29 and 30; i.e., they consisted entirely of coincidental matches of common terms in the C programming language. Even two programs known to have no code copied from one to the other will show these sorts of coincidental matches. Given the volume of code in question here (e.g., 68,000,000 lines of Unix code), the presence of these type of matches is both to be expected, and evidence that the tool was in fact performing successfully in finding potential matches.

[2] While COMPARATOR ignores a single line comment, i.e., a line of text that starts with "/*", it does compare the English text that appears in multi-line comments, allowing it to find identical or nearly identical multi-line comments in code. This is useful because overlaps in English comments can be an effective indicator that we ought to search for both literal and non-literal similarity in the source code that follows the comment.

33. In this instance, then, I did not need to perform a "filtration" analysis with respect to these matches, because they were not true matches of code at all. In any case, these matches would not be protectable under the filtration analysis. At best, they could be thought of as clichés or stock phrases, the sorts of things that are routinely "said" in source code by any author, and that cannot therefore be considered significant when looking for copying.

34. The SIM program did not report any matches between the IBM Code and the Unix System V code. As I result, I did not have to manually review any such code for false positives.

35. The remainder of this section describes the algorithms used by the comparison programs and the local modifications that were made to enhance the programs.

IV.1. COMPARATOR

36. The COMPARATOR program considers each file 3 lines at a time, and identifies all files that share the same 3 (or more) lines of code.

37. COMPARATOR "normalizes" its input, so that differences resulting from comments, case, and white space are ignored. This prevents immaterial changes that may arise from code copying from fooling the program. Then, all input is "shredded" into overlapping 3 line segments and identical segments from different files are gathered together.[3] Adjacent identical sections (e.g., lines 3–5 and lines 4–6) are then combined into a single section (e.g., lines 3–6).

[3] We require 3-line segments as a basis for a match in order to avoid the large number of accidental matches that would show up if only 1 or 2 lines were required to match. As a rough analogy, if we took two unrelated textual documents and looked for all matching 2-word sequences, we would find many of them (e.g., "and the," "used by," "with a," "were made," etc.), despite the fact that the documents were unrelated.

IV.2. SIM

38. The SIM program works by breaking source files into tokens (i.e., such as language keyword, punctuation, variables, constants, and the like) and comparing sequences of tokens for commonality. This conversion of source into tokens allows the program to focus solely on the structure of the code.

39. For example, a statement like

```
if (a > b) return a; else return b;
```

is structurally the same as

```
if (c > b) return c; else return b;
```

40. Both statements have the same syntactic structure, namely:

```
If (Var > Var) Return Var; Else Return Var;
```

which SIM would identify as a match.[4]

IV.3. Modifications to the Programs

41. Slight modifications were made to both of these programs to make them faster and more efficient, so that they could handle the large amount of source code under consideration in this case.

42. As publicly distributed, COMPARATOR and its associated scripts have several major performance bottlenecks, which were identified and removed by my assistant. These fixes improved the speed at which the program operated; they did not alter the methodology used by the program to find matches.

If we look for 3-word sequences in common (e.g., "used by the"), we would find far fewer of them, and could use those more reliably to build up evidence for matches.

[4] This is analogous to finding that the following two English sentences have exactly the same syntactic structure, yet are clearly not copied from one another: (a) "The tall boy threw the ball to the dog," and (b) "The coded message divulged the secret to the spy."

43. SIM was modified by my assistant to reduce the number of false matches it produced. It was determined that many matches reported by SIM arise because the program treats all numbers, strings and variable identifiers identically. For example, to SIM, a list of integers such as `1, 2, 3, 4` looks just the same as a list of very different numbers, such as `73234, 1592, 7182, 31415,` because syntactically they are both simply a list of four numbers. This occurs in the current context because operating systems code commonly includes long arrays of numbers that encode instructions for hardware. This also arises in structure initializations where there may be long sequences of identifiers. Arrays of character strings are also common as means of associating strings with certain numeric values (e.g., error codes and messages).

44. These false matches in SIM were avoided by first making tokenizing stricter – strings and numbers are considered to be the same only if they have the same value.[5] Next, a step within SIM itself removes matches that consist of a sequence where over 70% of the tokens are commas, identifiers, numbers, strings and tokens that are part of C's "switch" statements.

IV.4. Alternative Tools

45. Most other tools available to assist in organizing code for expert inspection operate in a similar manner. Tools like Jplag [3] and MOSS [4] operate similarly to SIM, tokenizing the input stream in order to compare code structure, but differing in the way they optimize the algorithms for performance. MOSS in particular uses a statistical

[5] More precisely, strings and numbers are considered the same only if they have the same hash value when hashed into a 256-value key. This is, in effect, a slightly "noisy" equality test: a few strings and numbers that are not in fact equal will be reported as equal. Note that this, too, makes our search *more* conservative, i.e., it will report a few *more* false positives.

sampling technique which results in a very small probability that a duplication may be missed.

46. The combination of line matching and syntactic analysis used in this comparison is similar to the technique used by CodeMatch [5], a commercial program for detecting code copying. CodeMatch uses the same algorithms as COMPARATOR and SIM and adds three smaller tests: comparing the number of identical words in two files, comparing the number of words in one file that appear as sub-words in another file, and checking comment lines.

47. SIM and COMPARATOR were chosen both because they provided the capabilities needed, and because they offered full access to their source code, making it possible to understand exactly how they worked and to customize them to the needs of this case. The comparisons I performed using SIM and COMPARATOR were intended to be as conservative as possible and to produce the most potential matches for me to review individually.

V. SUMMARY

48. After a detailed review that exhaustively compared almost 27,000 lines of IBM Code against almost 68,000,000 lines of Unix System V Code, I could find no evidence that any of the IBM Code incorporates a portion of, or is similar to, any of the Unix System V Code.

49. I therefore conclude that the IBM Code is not a modification or a derivative work based on the Unix System V Code.

50. I declare under penalty of perjury that the foregoing is true and correct.

Randall Davis

Randall Davis

13 August 2004

VI. REFERENCES

1. Raymond, Erik, COMPARATOR, http://www.catb.org/~esr/comparator

2. Grune, Dick, The software and text similarity tester SIM, http://www.cs.vu.nl/~dick/sim.html, Version 2.12.

3. Lutz Prechelt, Guido Malpohl, Michael Philippsen, Finding Plagiarisms among a Set of Programs with Jplag, *Journal of Universal Computer Science*, 2002, 8: 11, pp. 1016–1038.

4. Saul Schleimer, Daniel Wilkerson, Alex Aiken, Winnowing: Local Algorithms for Document Fingerprinting, *Proceedings of the 2003 ACM SIGMOD International Conference on Management of Data*, 2003, pp. 76–85.

5. Zeidman, Bob, *CodeMatch detects plagiarism,* http://www.zeidmanconsulting.com/codematch.htm