

 ORIGINAL

FILED
U.S. DISTRICT COURT

2005 JUL -5 P 4:45

Brent O. Hatch (5715)
Mark F. James (5295)
HATCH, JAMES & DODGE
10 West Broadway, Suite 400
Salt Lake City, Utah 84101
Telephone: (801) 363-6363
Facsimile: (801) 363-6666

Robert Silver (admitted pro hac vice)
Edward Normand (admitted pro hac vice)
Sean Eskovitz (admitted pro hac vice)
BOIES, SCHILLER & FLEXNER LLP
333 Main Street
Armonk, New York 10504
Telephone: (914) 749-8200
Facsimile: (914) 749-8300

Stuart H. Singer (admitted pro hac vice)
BOIES, SCHILLER & FLEXNER LLP
401 East Las Olas Boulevard – Suite 1200
Ft. Lauderdale, Florida 33301
Telephone: (954) 356-0011
Facsimile: (954) 356-0022

Stephen N. Zack (admitted pro hac vice)
BOIES, SCHILLER & FLEXNER LLP
Bank of America Tower – Suite 2800
100 Southeast Second Street
Miami, Florida 33131
Telephone: (305) 539-8400
Facsimile: (305) 539-1307

Attorneys for The SCO Group, Inc.

**IN THE UNITED STATES DISTRICT COURT
FOR THE DISTRICT OF UTAH**

THE SCO GROUP, INC.

Plaintiff/Counterclaim-Defendant,

v.

INTERNATIONAL BUSINESS
MACHINES CORPORATION,

Defendant/Counterclaim-Plaintiff.

**DECLARATION IN SUPPORT OF
SCO'S MOTION FOR
CONTINUANCE PURSUANT TO
RULE 56(f)**

[Docket No. 198]

(REFILED IN REDACTED FORM)

Case No. 2:03CV0294DAK
Honorable Dale A. Kimball
Magistrate Judge Brooke C. Wells

DECLARATION OF CHRIS SONTAG

1. My name is Chris Sontag and I am Senior Vice President and General Manager of The SCO Group, Inc. My office is located in Lindon, Utah. Unless otherwise noted or evident from their context, this declaration is based on my personal knowledge and information available to me from reliable sources. To the best of my knowledge, information and belief, the facts set forth herein are true and correct.

2. I submit this Declaration in support of the Plaintiff's Motion for a Continuance Pursuant to Rule 56(f) (the "Motion").

3. The Court should grant a continuance to allow SCO to conduct certain discovery that SCO requires to rebut IBM's Cross-Motion for Partial Summary Judgment on its Tenth Counterclaim for Declaratory Judgment of Non-Infringement ("IBM's Cross-Motion"). In addition, SCO will require sufficient time to review the materials produced by IBM in the event the Court grants the Motion.

I. Introduction

4. For SCO to obtain all necessary and reasonable evidence to support its claims and to oppose IBM's Tenth Counterclaim, SCO must undertake a line-by-line comparison of Linux code and UNIX code. Based on our review to date, SCO believes that such comparison will reveal substantial similarity between the Linux and UNIX code.

5. There are inherent obstacles in identifying substantial similarities between UNIX and Linux. Both the UNIX and Linux operating systems are large and complex computer programs with many lines of code to compare. Furthermore, Linux code that is modified or derived from UNIX code may not necessarily bear line-for-line character similarity.

6. A kernel is the core portion of the operating system. The kernel performs the most essential operating system tasks, such as handling disk input and output operations and managing the internal memory.

7. The operating system kernel is a lengthy, complex computer program comprising numerous modules and files, and millions of lines of code. The Linux kernel (ver. 2.4) comprises 4 million lines of code and the UNIX SVR 4.2 MP kernel comprises 3.4 million lines of code.

8. To show that Linux code is substantially similar to UNIX code requires a comparison of that code which, as described below, is an undertaking of great magnitude and complexity. In other words, the 4 million lines of Linux code must be compared with the 3.5 million lines of UNIX code, line-by-line, or in groups of lines according to the structure, sequence or function of the group of lines. In the paragraphs that follow, I will describe a time-consuming and resource intensive approach to this process, and ways in which this process can be streamlined.

9. There are two basic ways to execute the code comparison: 1) using an automated process or computer program, and 2) manual review by a knowledgeable individual.

10. Attempting to use an automated process to perform a complete comparison of all of the source code in UNIX and Linux computer operating systems is not feasible. Automated tools to find copied lines of code are available "off-the-shelf." The tools are designed to find lines of code that are identical in every detail; they perform that function well. SCO and its experts have sought to modify and improve the tools to locate lines of code that are not identical but that are nearly identical; the tools have not always performed that function well. The

automated tools occasionally assist a programmer locate blocks of code that might have similarities. The programmer must then visually review the code in a difficult and labor-intensive process. Often this review is only possible if each version of the code can be reviewed *to follow the changes from one version to the next.*

11. Minor changes to a line of code such as punctuation, renaming a variable, changing comments, spelling changes, or alterations to the text will prevent the automated system from locating the matches in the lines of code. Similarly, inserting, deleting, or reordering lines of code will prevent the automated system from identifying a block of similar code. *The reordering of lines of code may render the automated system useless.*

12. Despite these shortcomings, and as described further below, SCO and its experts have used automated tools to locate lines of identical code, and they have visually analyzed the larger blocks of code in which those lines appear. For example, in a block of code having 100 lines, if two or three lines were found to match, a visual review would then be undertaken of the entire 100 line block of code looking for other similarities.

13. The automated tools may provide "false positives" that need to be manually reviewed. Some automated tools can provide a numerical value or percentage that represents a degree of similarity. In practice, however, files with very low similarity numbers are sometimes found to be substantially similar, while others with high values of numerical similarity have been found not to be substantially similar. Therefore, files need to be checked manually and the *numerical similarity number is of little assistance.*

14. Because of shortcomings with automated code comparison processes, SCO and its experts must rely largely on manual comparisons. Such manual comparisons are very labor and

time intensive. SCO and its experts must know or learn both the UNIX and Linux operating systems in detail. This process can take many months. To execute the comparison, without some roadmaps or list of "hot spots" in Linux, SCO and its experts must compare page after page of code. The 4 million lines of Linux kernel code takes up 66,000 pages; the 3.4 million lines of UNIX code takes up 58,000 pages. A simplistic manual comparison would involve placing the pages of code side by side in some ordered manner and then looking for the same or similar structure, sequence and organization of the code. Assuming each page comparison takes one (1) minute, and that there are 66,000 x 58,000 comparisons, this "initial" review could take on the order of 25,000 man-years. Following the initial review, SCO and its experts must conduct a detailed comparison of likely copying candidates. This "second-level" review would also be very lengthy.

15. One shortcut to comparing the UNIX and Linux code might be comparing similar directory structures of the UNIX and Linux operating systems. For example, version 2.4 of the Linux kernel contains 530 subdirectories and about 8750 source and assembly files. See Understanding The Linux Kernel, D. Broet, O'Reilly, 2003. Assuming each of the 8750 files requires one (1) day to investigate, about 35 man-years would be required to review all the Linux kernel files. However, this calculation ignores the possibility that the two operating systems use different file names, and that similar code sequences may reside in entirely different files.

16. Another shortcut may be to compare files from the UNIX and Linux operating systems that share the same or similar names (because the names of certain files correspond to the file's function). However, any significant overlap in the names of files between the UNIX and Linux operating systems is statistically unlikely.

17. Two ways of determining the number of files in any two computer operating systems that share the same or similar names are: (i) look for files whose names share the exact same characters in the exact same order, or (ii) look for files whose names share almost all of the same characters in almost the exact same order. To produce the most relevant results, each such search should take account of certain pre-defined language tokens (whose similarity between operating systems is not particularly probative of copying).

18. To "look" for such files, SCO and its experts have used computer programs to compare the thousands of files in UNIX and Linux operating systems. That is, based on the limited discovery to date and the operating systems that SCO already possessed or were publically available, SCO performed initial searches to find files that share the same or similar names. These comparisons represent only a small fraction of the total number of comparisons that could be made among the numerous versions of the UNIX, AIX/Dynix, and Linux operating systems.

19. The foregoing searches have permitted SCO and its experts to identify numerous files that, as between the UNIX and Linux operating systems, share the same or very similar names. SCO and its experts have used the results of the file searches to then turn to comparing the source code in those files.

20. Once they identified particular files with the same or similar names in UNIX and Linux, SCO and its experts used a combination of a computer program and manual review of the results of the program to find instances of substantial similarity in the operating systems. *There are significant limitations to any such approach, and there is no way to eliminate human review and assessment of the program's results -- both of which are extremely time consuming.*

21. SCO and its experts have used computer programs to identify the extent of similarity between lines of source code in any two given files. The results of one computer program shows where there are any differences between the lines of code. When a second computer program is run on those results, it shows where the lines of code (although different in some way) nevertheless contain the same code in the same sequence.

22. Once both computer programs have been run, SCO and its experts manually reviewed the results to assess the similarity between the lines of code at issue. The manual reviewer searches for instances where parts of the lines of code being compared are syntactically synonymous. That is, the reviewer determines whether the line of code in one file uses different words or characters to describe the same structure, function, declaration or subroutine as a line of code in the other file.

23. To date, this combination of automatic and manual review has been completed for *only a very small portion of the Linux and UNIX operating systems, despite a significant man-hour expenditure, on the order of two-man years.*

24. Another way for SCO to obtain all of the reasonably available and necessary evidence to support its claims and to oppose IBM's Tenth Counterclaim is to access the numerous IBM and Sequent engineers and programmers who have, over the years, developed AIX and Dynix code, contributed AIX and Dynix code to Linux, or assisted others in contributing to Linux. These engineers have access to and have studied UNIX based operating systems that have been enterprise hardened and made multiprocessor capable.

25. Once identified, the programmers and engineers can be deposed and can provide identities of Linux contributors for further discovery, discuss their own Linux contributions,

discuss assistance given to Linux contributors, and discuss specific code segments that were contributed to Linux. This will also assist SCO in identifying former IBM employees who are contributing to Linux.

26. As will be discussed below, a revision control system (RCS – implemented by IBM as the Configuration Management/Version Control (CMVC) is an excellent source for finding the programmers and engineers familiar with relevant UNIX based code that has been contributed by IBM and third parties to make Linux enterprise hardened and multiprocessor capable. Deposing these programmers and engineers will allow SCO to prioritize its efforts to find Linux code that is substantially similar to UNIX code.

27. As discussed in the Declaration of Sandeep Gupta, the Linux kernel uses a ULS routine to block and unblock access to shared data. The Linux ULS routine is substantially similar to a ULS routine in UNIX. A Mr. Russel of IBM helped a Mr. Jamie Lokier contribute the UNIX ULS code into Linux. If SCO had access to IBM's CMVC, then SCO might have discovered that Mr. Russel worked on ULS for IBM, and could have deposed Mr. Russel to determine what specific help he provided in the contribution of ULS to Linux and to whom he provided that help.

28. Using the CMVC, and by deposing individuals such as Mr. Russel of IBM, SCO can significantly reduce the burden of reviewing Linux and UNIX code. Mr. Russel and other programmers can identify areas of Linux code that are copies of or are derived from AIX and Dynix code. Mr. Russel and other programmers can also identify contributors to the Linux code and can show the necessary access to AIX and Dynix that these contributors had.

29. SCO also can streamline obtaining all of the reasonably available evidence to support its claims and to oppose IBM's Tenth Counterclaim and can prioritize its search of Linux code that is substantially similar to UNIX code by examining the lineages of AIX and Dynix. By examining the source code in successive versions of AIX and Dynix, SCO can trace its UNIX code through to current versions of AIX and Dynix to determine where in Linux, SCO's UNIX code is copied. This tracing will allow SCO to prioritize its search of Linux code for evidence of copying of UNIX code.

30. Software (i.e., source code) undergoes many changes during its initial development and later over its operational life. Changes may occur as frequently as daily and can continue for years. Software changes typically are driven by the need to correct "bugs," to improve features, or to add new features. Because of changes made to source code over time, a current code version may "look" different than the initial code version, making identification of the initial code version difficult and substantial similarity and derivation more difficult to establish.

31. Software developers rely on version control systems (VCSs), or version management systems (VMSs), to control changes and revisions to source code. Version control systems are automated tools that provide specific access and tracking features to allow multiple parties to operate on and revise source code. For example, a "Checkout" feature allows a user to retrieve, from a source code repository, a section of source code for which some changes are intended. A "Checkin" feature deposits the changed source code in a source code repository. Version control systems also provide an approval process, and many other features. In short, version controls systems are software tools that provide detailed software change histories.

32. Related to the VCS is a "bug" tracking system or log. The bug tracking system allows users to log problems encountered with source code. Some bug tracking systems are implemented as web service applications, and allow software users to register problems using a Web page-provided form. Other bug tracking systems are internal to the company developing or supporting the source code. Because software changes are often driven by problem reports, it is natural to integrate these bug tracking systems with version control systems: this allows for a framework where changes resulting from a bug report can be easily located, and where some measure of certainty is provided that changes have been integrated into a product release. For large-scale software development projects, such integration is mandatory.

33. Both VCSs and bug tracking systems typically allow for some type of commentary to explain why a source code change was needed and to explain what was changed. VCSs and bug tracking systems are typically maintained in an electronic format, although hardcopy printouts may be available.

34. The advantage of the VCS is that it is an ongoing snapshot of how the software development took place. The VCS allows a user to view, through time, all changes to software by time and date, author, and possibly a reference to the bug tracking system. The VCS is an essential tool for software developers. For example, a bug may be reported to a software company, and to develop a correction, a software developer may refer back several years, or even decades, to prior versions of code so as to understand how the error (bug) developed, and how to revise the code to eliminate the bug. Without the VCS, this process could not be completed.

35. The VCS is also an essential tool that SCO can use to support its claims and to oppose IBM's Tenth Counterclaim. By viewing each version of AIX and Dynix, along with the associated CMVC or similar system, SCO will be able to track a Dynix/AIX code segment through its many derivative versions to its ultimate location in Linux. This will significantly streamline SCO's efforts to find code in Linux that is substantially similar to UNIX code. Moreover, the CMVC will identify programmers who can be deposed and who can explain where in Linux the code was contributed. By viewing each version of the Dynix/AIX code, SCO will be better able to determine if the structure, sequence, and organization of the corresponding Linux code matches that of UNIX.

II. Discovery Required From IBM

36. Similar to a software developer chasing a bug through time, SCO should be able to trace the development of UNIX-based source code from its initial AIX and Dynix versions through to current versions of the AIX and Dynix code and then into Linux. AIX, Dynix, ptx, and Dynix/ptx consist of millions of lines of source code, much of which likely will have undergone numerous (possibly hundreds) of changes. Tracing the current AIX, Dynix, ptx, and Dynix/ptx code versions to earlier code versions, and then ultimately to the corresponding UNIX code, will not be possible within any workable timeframe without a detailed "road map." The VCSs and bug tracking systems provide this road map. Additionally, the VCSs identify the software developer who authored the change and may now be assisting with development of Linux. These developers can be deposed to provide information that will help SCO prioritize its efforts to locate Linux code that is substantially similar to UNIX code.

37. Shown and described below is an example of UNIX SVR4 source code illustrating accumulated modifications over time. This is SCO source code for which SCO has the versions available. The specific file name is `perror.c`.¹ Table 1 illustrates `perror.c` version 1.1, written in 1981 as compared to version 1.17, written in 1992. The shading indicates differences between the two versions. As can be seen from a casual review of the table, over 50 percent of the source code lines changed from the 1981 version to that from 1992. In fact, `perror.c` version 1.17 is so changed from version 1.1, that even an experienced UNIX programmer would have trouble determining that one was derived from the other.

38. The difference plot shown in the above Table 1 is replicated in color as Exhibit A. In the exhibit, the shaded areas are shown in two colors, pink and yellow. The yellow-shaded areas are lines of code where differences exist. Within each yellow-shaded area are pink-shaded areas that highlight specific differences in the code. Unshaded areas in the exhibit are lines of code where the two versions are identical.

39. Table 1 and Exhibit A show differences between UNIX code versions 1.1 and 1.17, and the differences are significant. However, as the code version numbers imply, there are many versions of the `perror.c` file from 1981 to 1992. Each of these code versions involves generally small variations from prior versions. It is the accumulation of these changes over time that makes the final version (i.e., version 1.17) look so different from the initial version (i.e., version 1.1). Exhibits B through L are differences plots of selected neighboring `perror.c` versions

¹ The code file "`perror.c`" is an unusual example of a UNIX source code file in that the file consists of only one short function (to generate a one-line description of the most recent error code, and provide a diagnostic). `perror.c` is written in 28 lines (in version 1.1). More typically, UNIX source code files consist of multiple functions and thousands of lines of code. File `perror.c` was chosen as an example for this Declaration because it illustrates the relevant concepts related to version control in a compact, easy to understand format.

Table 1

REDACTED

from the initial version 1.1, through version 1.17. Viewing any of Exhibits B through L, a skilled UNIX programmer can readily see the evolutionary, derivative nature of the `perror.c` code development from 1981 to 1992. This derivative nature is not, however, readily apparent to the same skilled UNIX programmer based on the *difference plot of Exhibit A alone*.

40. As noted above, programmers use a VCS to track changes to code. The VCS serves as a road map that tracks all the code changes over time. Exhibit M is a printout of that portion of a VCS related to the `perror.c` file. As can be seen from Exhibit M, each change in the `perror.c` file is accompanied by an entry in the VCS that includes the date, identity of the author, and a *comments section that lists the nature of and the reason for the change*. Each entry in the VCS also references a corresponding entry in a bug tracking log. For example, entry D1.10 in the VCS, which relates to the change in `perror.c` from version 1.9 to version 1.10, refers to corresponding entry (referred to as *UNIX Modification Request # bl86-28117*) in the bug tracking log. Exhibit E shows the difference plot for this change. Exhibit N is the corresponding entry bl86-28117 from the bug tracking log. As can be seen from Exhibit N, the bug tracking log entry describes the specific problem that exists with the current version (in Exhibit E, the noted problem is that the existing error code does not check for an *error (errno) less than zero*) and lists what should be done to the `perror.c` code sequence to correct this problem. The bug tracking log entry also lists the originator of the log entry (in Exhibit E, D.E. Good), and the individual assigned to *correct the problem (mao)*. The log entry further identifies the individual who approved the proposed problem correction (prb), the reason for change (error in design implementation) and other information that *relates to the perror.c code*.

41. Viewing Exhibit E, which is the difference plot between `perror.c` versions 1.9 and 1.10, line 20 of versions 1.9 and 1.10 are highlighted. The actual code change between the versions is shown in Table 2:

Table 2

REDACTED

42. As the example in Table 2 makes clear, even a skilled programmer would likely not be able to determine the derivation of the current `perror.c` code sequence without a road map that lays out specific changes in detail.

43. In view of the information provided in Paragraphs 4 - 42, SCO requires the following materials to analyze source code so that it can rebut IBM's Tenth Counterclaim:

- all version control system and bug tracking information (including documents, data, logs, files, and so forth) for AIX, Dynix/ptx, ptx, and Dynix from 1984 to the present,
- source code and log information for all interim and released versions of AIX, Dynix, ptx and Dynix/ptx from 1984 to the present, and
- depositions as appropriate of programmers identified from the foregoing.

44. The VCS information is especially important to SCO's opposition of IBM's Tenth Counterclaim. Without VCS information for AIX, Dynix/ptx, ptx, and Dynix, and any related

information such as documents, data, logs, files, SCO will not be able to prioritize its efforts to identify all lines of code in Linux that are derived or copied from UNIX System V. SCO will instead have to rely on luck and happenstance to find derived and copied code, then trace such code back to System V. The VCS information, however, will help SCO streamline its search efforts to find evidence that Linux code was copied or derived from UNIX System V code.

45. The materials in Paragraphs 31 and 32 – both the VCS information and the source code and log information – directly respond to IBM's factual allegations that the Linux code was developed or created by programmers, rather than taken from System V. *See, e.g., IBM's Cross-Motion* ¶¶ 1 (“collaborative development”); 2 (Linus Torvalds created a “new” operating system); 3 (“programmers joined to create code”); 4 (developers “contributed to the further development of Linux”). The VCS information will establish that various versions of AIX and Dynix are in fact derivative works of UNIX System V, and consequently, IBM's contributions to Linux from AIX and Dynix constitute copyright infringement.

46. Furthermore, the VCS information and the source code and log information will allow SCO to rebut IBM's allegations that SCO cannot prove copying (*IBM's Cross-Motion* ¶¶ 46 (SCO cannot show that IBM's activities infringe SCO's copyrights); 48 (SCO cannot establish that material in Linux is covered by SCO's copyrights)), as such information shows the history of development of AIX and Dynix code, the authors of the various versions of those systems, and the sources of the code. In other words, if a portion of Dynix code was obtained from UNIX System V, the VCS information for Dynix would show where that portion of code originally came from, who obtained it and when, and how that code was used in Linux.

47. SCO believes that much of its copyrighted code was copied from AIX and Dynix into Linux. While SCO has some evidence of literal copying between System V on the one hand, and Linux on the other hand, the VCS and source log information will show changes between various versions of AIX and Dynix, and the detailed history of those changes. Thus, SCO will be able to show that Linux code is substantially similar to UNIX code. SCO must have this material to establish what material in Linux is covered by SCO's copyrights, which IBM alleges SCO cannot do. *IBM's Cross-Motion* ¶ 48.

48. The evidence SCO currently has – three versions of AIX that IBM selected, Linux code, and System V code – is insufficient to show infringement because IBM could have copied System V code into any number of the multiple versions of AIX and Dynix. To trace SCO-owned code from System V into the code's current form in Linux, SCO must be able to trace every step and change the code underwent through AIX and Dynix. To do so, SCO requires the VCS information and the source code and log information. Without this material, SCO will not be able to prioritize and streamline its search efforts and will have to expend considerable time and resources to find evidence that Linux code is substantially similar to UNIX code.

49. *IBM will not bear any significant burden to produce VCS information, IBM stores this information on its Configuration Management Version Control (CMVC) system. See IBM's CMVC Introduction (1710058191-92) (Exhibit O hereto).*

50. SCO also requires the following materials to oppose IBM's Tenth Counterclaim: All design documents, white papers and programming notes, created from 1984 to the present.

These materials provide a wealth of information related to code development beyond that which can be found in the source code testing, VCS and bug tracking log.

51. White papers are usually generated early in the software code development process, and often discuss reasons for implementing code changes, problems with existing code, and alternative solutions. Thus, white papers serve as an early indication of possible code changes. By setting forth solutions, white papers can be used to look for specific code segments in Linux and thus help SCO prioritize its search.

52. Design documents are often prepared by the group that ultimately authors the changes to the code sequences. Design documents are generally more detailed than white papers. For example, SCO proprietary design document "Virtual Memory Design for UNIX System V Release 4.2 Multiprocessor," contains almost 150 pages of detailed description and code requirements to implement virtual memory in a UNIX-based processor. The design document is directed to such implementation on a specific processor family, namely the Sequent Symmetry Model S16. This and other design documents explain the initial code concepts, and how such code will be developed and written. As such, design documents provide an invaluable bridge between existing code sequences, such as in UNIX, and derivative works, such as in AIX and Dynix. Because these design documents describe the basis for code development, they may be useful for pointing to a portion of Linux that contains code substantially similar to UNIX code.

53. Finally, programming notes contain the thought processes of individual programmers as they write and revise code sequences. For example, programming notes might list changes made to code, and might list additional changes to consider. As such, programming notes provide detailed rationale for code changes and an indication of how the code may change

in the future. Programming notes may reflect the purpose for code changes and where in the kernel those changes occur. Thus programming notes are another source SCO can use to streamline its efforts to locate Linux code that is substantially similar to UNIX code.

54. SCO requires these white papers, design documents, and programming notes for all AIX, Dynix, ptx, and Dynix/ptx.

55. To find copying of other operating systems (e.g., AIX, Dynix, ptx, and Dynix/ptx) and features of UNIX System V, SCO must have the discovery related to items listed in Paragraphs 4 - 54. SCO believes that many of these features of its System V were copied, directly and/or indirectly, into AIX and Dynix by IBM. Therefore, SCO requires discovery of materials related to these aspects of AIX and Dynix to establish IBM's copyright infringement and rebut its Cross-Motion. Also, information is needed as to these code sequences to determine if some UNIX code has been copied, and if the Linux version is a substantially similar to UNIX.

56. SCO previously requested the above-listed materials in *SCO's Memorandum Regarding Discovery* submitted to Magistrate Judge Brooke C. Wells on May 28, 2004, pursuant to Magistrate Judge Wells' Order dated March 3, 2004. To date, Magistrate Judge Wells has not ruled on SCO's May 28 memorandum.

III. Discovery Required From Third Parties

57. Unlike AIX and Dynix, Linux code was developed in a somewhat unstructured format. At a minimum, no single version control system, or group of version control systems was implemented to track the derivation and evolution of Linux and to screen out contributions that may constitute copyright infringement. Furthermore, IBM has stated that thousands of programmers contributed code to Linux, and hundreds of Linux versions exist. Simply put, no

road map exists that will allow SCO to trace the migration of UNIX code into Linux completely. Thus, the discovery sought by SCO will permit SCO to identify major contributors to Linux so as to focus and narrow discovery. Clearly, it is impossible to seek discovery from thousands of contributors worldwide, and SCO does not intend to do so. Accordingly, SCO requires discovery relating to third party contributions to Linux – thus demonstrating that IBM's use of Linux constitutes infringement – as follows:

- Determine what third parties IBM has partnered with to develop Linux and what work those groups have done. Many of these arrangements are not in the public domain, particularly as to the details of the partnering, such as which party makes what contribution, the motivation for the contribution, and the starting and ending code versions that resulted from the partnership. This discovery will also help SCO identify specific code authors, who can then be deposed.
- Take discovery on Linus Torvalds, the purported creator of Linux, about the contributors and contributions to Linux since its inception, and the maintenance of any records about the development history of Linux. Mr. Torvalds is expected to have detailed records of these contributors and their contributions, material that is not publicly available. Further, Mr. Torvalds can answer specific questions as to what each contributor intended, and where and how the contributor acquired or developed the derived code.
- Take discovery on maintainers of the kernels. Kernel maintainers take responsibility for approving and including patches for Linux, and should have a wealth of information on who has contributed what code to the various Linux kernels over the years.
- There are many contributors to the kernels, some who have significant contributions to Linux code over the years. Some of these individuals, whose names are publicly available, should be deposed to find out their sources for their contributed code.
- Many corporations have made contributions to Linux, and SCO needs to take discovery on certain of these companies to determine the sources of their contributions. Also, SCO needs to depose the programmers who work for these companies and made the contributions to determine the sources of those programmers' code contributions. This discovery will show why the contributions were made and what features the contributions relate to, and will allow SCO to trace back from the Linux code to UNIX.

- SCO has identified some, but not all, independent authors of various portions of Linux code. (See partial list at Exhibit P hereto.) Those authors should know the sources of their code and should be able to provide information as to whether the code they contributed to Linux was obtained from SCO copyrighted code.
- Several private groups also made major contributions to Linux, so SCO should also be permitted adequate time to identify and take discovery from these entities.
- Many organizations exist whose purpose is to track and report on changes to Linux, and in many cases to collect documentation on Linux and distribute that information. However, such reporting is generally very summary, and SCO needs access to the more detailed information these organizations maintain. Such organizations are also potential sources of infringement information.
- Licensees and former licensees of UNIX source code to see if these entities, their employees, or former employees are contributing UNIX code to Linux.

58. IBM asserts that SCO has yet to set forth evidence that any Linux code infringes any SCO copyright and that SCO cannot do so. To counter this assertion more fully, SCO needs the discovery requested herein and the time to analyze it to find all instances of substantial similarity.

59. This discovery will provide leads as to which portions of the Linux code have copied portions from UNIX. Linux is so extensive that an evaluation of each of its 8750 files would be an enormous task. SCO needs some initial discovery in the area of third-party contributions to Linux to focus SCO research and further discovery.

60. SCO has not yet undertaken to conduct this far-reaching third party discovery because it is relevant only to IBM's tenth counterclaim which was only recently filed and not to SCO's copyright infringement claim. As a user of Linux (which IBM contends is one of its Linux activities), IBM copies Linux. Therefore, IBM's use and copying of Linux necessarily involves all of the source code contributed by others, as well as its own. Thus, SCO requires a continuance to take this discovery.

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

July __, 2004

A handwritten signature in black ink, appearing to read "CS", written over a horizontal line.

Chris Sontag

CERTIFICATE OF SERVICE

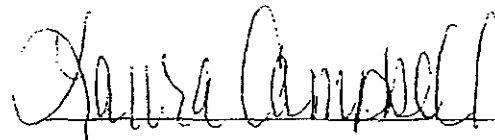
Plaintiff, The SCO Group, hereby certifies that a true and correct copy of
**DECLARATION IN SUPPORT OF SCO'S MOTION FOR CONTINUANCE
PURSUANT TO RULE 56(f)** was served on Defendant International Business Machines
Corporation on the 9th day of July, 2004, as follows:

BY HAND DELIVERY:

Alan L. Sullivan, Esq.
Todd M. Shaughnessy, Esq.
Snell & Wilmer L.L.P.
15 West South Temple, Ste. 1200
Salt Lake City, Utah 84101-1004

Evan R. Chesler, Esq.
Cravath, Swaine & Moore LLP
825 Eighth Avenue
New York, NY 10019

Donald J. Rosenberg, Esq.
1133 Westchester Avenue
White Plains, New York 10604

A handwritten signature in black ink, appearing to read "Aurora Campbell". The signature is written in a cursive, flowing style.

CERTIFICATE OF SERVICE

Plaintiff/Counterclaim Defendant, The SCO Group, Inc., hereby certifies that a true and correct copy of the foregoing was served on Defendant IBM on the 5th day of July, 2005 by U.S. Mail to:

David Marriott, Esq.
CRAVATH SWAINE & MOORE LLP
Worldwide Plaza
825 Eighth Avenue
New York, NY 10019

Donald Rosenberg, Esq.
1133 Westchester Avenue
White Plains, NY 10604

Todd Shaughnessy, Esq.
SNELL & WILMER LLP
1200 Gateway Tower West
15 West South Temple
Salt Lake City, UT 84101-1004

Aura K. Chavez