Brent O. Hatch (5715)
HATCH, JAMES & DODGE, PC
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)
BOIES, SCHILLER & FLEXNER LLP
333 Main Street
Armonk, New York 10504
Telephone: (914) 749-8200
Facsimile: (914) 749-8300

Devan V. Padmanabhan (admitted pro hac vice)
DORSEY & WHITNEY LLP
50 South Sixth Street, Suite 1500
Minneapolis, Minnesota 55402
Telephone: (612) 340-2600
Facsimile: (612) 340-2868

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

Stuart Singer (admitted pro hac vice)
BOIES, SCHILLER & FLEXNER LLP
401 East Las Olas Blvd.
Suite 1200
Fort Lauderdale, FL 33301
Telephone: (954) 356-0011
Facsimile: (954) 356-0022

Attorneys for Plaintiff, 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 OF EVAN IVIE IN SUPPORT OF SCO'S MOTION FOR RECONSIDERATION BY THE MAGISTRATE COURT OF THE ORDER DENYING SCO'S MOTION FOR RELIEF FOR IBM'S SPOLIATION OF EVIDENCE** Case No. 2:03CV-0294DAK Honorable Dale A. Kimball Magistrate Judge Brooke C. Wells

1. I was retained by counsel to SCO to analyze the technical evidence in this case and to serve as a consultant and expert witness. I have been asked to comment on the use of sandboxes or similar programming environments in the development of Linux code at IBM, the importance to the case of information in such sandboxes, and whether other sources of information such as CMVC or RCS might suffice in evaluating what went on in that development process in sandboxes. My qualifications are set forth in my May 19, 2006 report submitted in this case.

Programming Environments (e.g. Sandboxes) Were Used by IBM Linux and Dynix/ptx Programmers

2. I understand that counsel for IBM represented that sandboxes were not used for Linux development. Any creative process requires an environment and facility where that creative process can take place. An artist creates paintings in a studio. A woodcarver creates carvings in a woodshop. Artisans, craftsmen, and skilled workers develop facilities where they can perform their work: workbenches, body shops, bakeries, etc. Programmers are no different.

3. At Bell Labs I decided to use Ken Thompson's newly developed Unix operating system as the basis for a Programmer's Workbench (PWB), a facility where programmers could create software. Unix was an ideal environment for such work when enhanced with the set of tools that we developed. Within Bell Labs, and at a number of other companies, the PWB became the standard environment for software development at that time. Whether you call this workspace a programmer's workbench, a sandbox, or some other name, it is essential to the software development process. IBM has used the term "sandbox" in a number of depositions and other documents.

4. Thus, contrary to IBM counsel's representation, I believe that IBM programmers for Dynix/ptx and Linux, as well as AIX, used sandboxes, or other similar workspaces or programming environments, to draft, revise and implement computer software for those systems.

5. If IBM had not adopted and/or developed some type of suitable environment for their programmers, it would have taken them back to the late 1950's and early 1960's and would have made programming an incredibly inefficient and slow process. This would be like taking a tractor away from a farmer and giving him a shovel. Even if IBM had tried it, programmers would have resisted the move.

Information in Programming Environments (e.g. Sandboxes)

6. An adequate programming environment provides a place where all of the basic functions involved in software development can be performed. This includes the storage space to keep code, data, documents, and other information. It also includes suitable tools to perform the functions needed to create a software system. Below is a list of some of the activities performed in a programming environment or sandbox:

- the creation of design documentation specifying the functionality of each module, routine, subsystem, etc.;
- the definition of interfaces between software entities including arguments, parameters, flags, sequencing, etc.;
- the specification of structures, file formats, databases, constants, variables, etc.;
- the approach to be used in the testing each function, subroutine and module;
- the plan for subsystem and system level testing;
- the specific tests to be undertaken, the test scenarios, the test data, etc.;
- the capability to perform such tests in the context of the total system;

- the creation of code modules, the compiling of that code, and the linking of it together;
- the reasoning, behind algorithms and the motivation behind the various designs;
- all of this documentation for each software entity at each level of abstraction; and
- suitable backup to protect the system and so that when programmers hit a roadblock the system can be rolled back to an earlier state and development can begin again there.

7. In addition to the sandbox or programming environment functionalities enumerated above, any sizable software development effort requires the use of a change/version management and control system. The SCCS system developed by Marc Rochkind is the granddaddy of such systems. Other such systems involved in this case include CMVC, RCS, CVS, and bitkeeper.[12] A source code control system is one good source of information when trying to track a software development effort, and I did a number of searches of revision control information in these systems. One example of my use of this information is found paragraph 98 of my expert report where I note that a CMVC entry admits that JFS is based on System V Unix.

CMVC and Other Such Systems Are Not a Substitute for Programming Environment Information

8. There are several fundamental flaws in the use a change control system, such as CMVC or RCS, to track a software development effort. A typical change control system allows a programmer to "check out" a module, to modify and test it for some unspecified

[1] "What is your preferred revision control system, http://www.perlmonks.org/?displaytype=print;node_id=394350;replies=1

[2] " List of Revision Control Software, http://en.wikipedia.org/wiki/List_of_revision_control_software

amount of time, and then when satisfied to "check it back in" to the system. Perhaps this might be compared to trying to see what is going on in a darkened room with a strobe light. However, the strobe only illuminates a small part of the room (the code checked in and out) and the strobe is controlled by a programmer who may or may not want you to see all that is going on (visibility only at check-in time).

9. For example, let us assume that the programmer is developing a module for Linux, but is basing it on a module that comes from a contractually-protected operating system owned by another company. If the only visibility that we have is the module after it has been appropriately disguised, then tracing the source becomes much more difficult. Perhaps one could compare this to a body shop for processing stolen cars. It is much easier to prove auto theft if one can find the body shop being used. After a paint job, changes to upholstery, options, accessories, and careful modification of the engine and body numbers, it is much more difficult to identify the theft.

10. It should be noted that a developer's computer can hold more than one sandbox. They could be different versions of the same project, or they could be different projects. Take for example a computer with both an AIX and Linux sandbox. This creates the capability for a programmer to copy code from one sandbox (that had been checked out from the change control system) and use it in developing code for another sandbox.

11. Another specific example of valuable information that would have been in a programming environment or sandbox, but not in a change control system such as CMVC or RCS, is the timing of access to code files. Most computers, including Unix and Windows,

maintain the last access time of each file they store.[3] In other words, the computer keeps a record of the last time a file was viewed. If a file containing AIX or Dynix/ptx code was viewed in proximity to access to a Linux file, or even after the Linux files were created, that would cause concern that the AIX files were used to develop the Linux files.

12. The claim that SCO did not need access to the programming environment/sandbox information because the code was available in CMVC (or RCS) ignores the following basic problems:

- CMVC and RCS would not show whether code from AIX and Dynix/ptx was copied, retained, and used by IBM Linux programmers in the development of IBM contributions to Linux, or what particular code was copied, retained, and used.

- A programming environment or sandbox is the only place where the progression of code drafts can be viewed, from the initial version to subsequent versions. For AIX code, CMVC shows the initial code that was checked out, and the final code that was checked back in, but not all the steps in between. RCS, the system on which Dynix/ptx code is saved, shows even less. These intermediate drafts, – saved only on programmers' sandboxes or similar workspaces –would have been important to develop further proof of IBM's copying.

Importance of Programming Environment Information to SCO/ IBM Case

13. Despite the loss of programming environment and sandbox information through the destruction initiated by IBM management, we were able to find numerous cases where code from Dynix/ptx and AIX found its way into IBM's disclosures to Linux. This effort would have been significantly easier if the evidence in programming environments and sandboxes had not been destroyed.

[3] "Unix records three file times in the inode, these are referred to as ctime, mtime, and atime. The ctime field refers to the time the inode was last changed, mtime refers to the last modification time of the file, and atime refers to the time the file was last accessed."
http://userpages.umbc.edu/~jack/ifsm498/filesystem.html

14. If I had known which Dynix/ptx and AIX code IBM's Linux programmers had retained on their programming environments or sandboxes, I would have compared the programmers' Linux disclosures to that code – which would have been easier than trying to compare the final Linux disclosures to the entire body of AIX and Dynix/ptx code available. This would have enabled more specific identification of the AIX or Dynix/ptx code on which the programmers' Linux disclosures was based

15. If I had access to drafts of programmers' Linux code from programming environments and sandboxes that also contained AIX or Dynix/ptx code, I could have identified even more specifically the copying that occurred.

I hereby certify under penalty of perjury and the laws of the State of Utah and the United States of America that the foregoing is true and correct.

Signed this 16th day of March 2007 at Nauvoo, Illinois.

Evan L Ivie

Dr. Evan Ivie

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/Counterclaim-Plaintiff, International Business Machines Corporation, on this 20th day of March 2007, via CM/ECF to the following:

David Marriott, Esq. (dmarrriott@cravath.com)
Cravath, Swaine & Moore LLP
Worldwide Plaza
825 Eighth Avenue
New York, New York 10019

Todd Shaughnessy, Esq. (tshaugnessy@swlaw.com)
Snell & Wilmer LLP
1200 Gateway Tower West
15 West South Temple
Salt Lake City, Utah 84101-1004

/s/ Edward Normand