August 1, 2002

Setup and Test Procedures

dd (GNU fileutils) 4.0.36 Forensic Tests

Version 1.0

NIST

National Institute of Standards and Technology

Technology Administration, U.S. Department of Commerce

Abstract[†]

This document describes the testing of **dd** (GNU fileutils) 4.0.36 as a disk imaging tool on a Linux platform. The Linux version used was Linux version 2.4.2-2 (Red Hat Linux 7.1 2.96-79). The test cases that were applied are described in *Disk Imaging Tool Specification, Version 3.1.6*.

The tests were run on five 933 Mhz computers. A variety of hard drives (7 different models, 5 major brands) were used for the tests. The source disks (the ones that are copied from) were setup with FAT16, FAT32, NTFS or Linux EXT2 type partitions to represent the most common partition types.

The main objective of this document is to provide enough information about the testing process for either an independent evaluation of the process or independent replication of the results. The intended audience for this document should be familiar with the DOS operating system, Linux (or some UNIX like) operating system, computer operation, computer hardware components such as hard drives, hard drive interfaces (e.g., IDE or SCSI) and computer forensics.

[†] **Certain trade names and company products are mentioned in the text or identified. In no case does such identification imply recommendation or endorsement by the National Institute of Standards and Technology, nor does it imply that the products are necessarily the best available for the purpose.**

Table of Contents

List of Tables

1 Introduction

The objective of the Computer Forensics Tool Testing project is to provide a measure of assurance that the tools used in computer forensics investigations produce accurate results. This is accomplished by developing specifications and test methods for computer forensics tools and then testing specific tools. The test results provide the information necessary for toolmakers to improve tools, for users to make informed choices about acquiring and using computer forensics tools, and for the legal community and others to understand the tools' capabilities. Our approach for testing computer forensic tools is based on well-recognized methodologies for conformance testing and quality testing.

The CFTT is a joint project of the National Institute of Justice, the National Institute of Standards and Technology, and other agencies, such as the Technical Support Working Group. The entire computer forensics community helps develop the specifications and test methods by commenting on drafts as they are published on the NIST website http://www.cftt.nist.gov/.

This document describes the procedures used for testing **dd** (GNU fileutils) 4.0.36 as a disk imaging tool on a Linux platform. The Linux version used was Linux version 2.4.2-2 (Red Hat Linux 7.1 2.96-79). The test cases that were applied are described in *Disk Imaging Tool Specification, Version 3.1.6*. The main objective of this document is to provide enough information about the testing process for either an independent evaluation of the process or independent replication of the results. To attempt an independent replication of the reported test results, an agency or lab other than NIST would require sufficient hardware and software resources to execute the test cases plus this document, the **dd** test report and *Disk Imaging Tool Specification, Version 3.1.6*. Since it is unlikely that the exact hardware used by NIST is present, adjustments and substitutions must be made to run the test cases. Section 7 gives suggestions for adapting to a test environment different from the environment at NIST.

The intended audience for this document should be familiar with the DOS operating system, Linux (or some UNIX like) operating system, computer operation, computer hardware components such as hard drives, hard drive interfaces (e.g., IDE or SCSI) and computer forensics.

1.1 Testing Overview

To accomplish the testing several items must be assembled and prepared. These include computers to execute the tests, hard disk drives, removable media, support software (FS-TST Version 1.0) and scripts to control the testing process. The support software, *FS-TST: Forensic Software Testing Support Tools*, is available from the web site: http://www.cftt.nist.gov.

A subset of the hard drives is selected for initial setup as source drives for the test cases. The source drives are setup once and then used for multiple test cases. After all the components are prepared the test cases are run. All the test cases follow a similar execution plan of three steps.

1. Prepare the destination drive. Specified values are written to each sector of the destination drive. If a partition is required, it is created and formatted. This step is executed in a DOS environment.
2. Execute **dd**. This step is executed in a Linux environment.
3. Measure the results. The accuracy and completeness of the copy is checked by a sector-by-sector comparison. The source drive is checked for any change by comparing a SHA-1 taken before the execution of **dd** with a SHA-1 taken after **dd** is executed. This step is executed in a DOS environment.

For each test case, the commands that need to be executed are contained in one script file for each step. Except for partition creation and formatting, the programs required to setup each test case and to measure the results are contained in the FS-TST package.

1.2 Test Case Selection

Not all of the test cases defined in *Disk Imaging Tool Specification, Version 3.1.6* apply to every tool. Table 1.2-1 presents the parameters for each test case defined in *Disk Imaging Tool Specification, Version 3.1.6*. The test cases are written for a general Linux based imaging tool. Not all cases apply to **dd**. In the case of **dd**, 32 test cases were run, 20 of the test cases did not apply. A mapping of test cases to requirements tested is presented in Table 1.2-2.

The interpretation of each parameter is as follows:

- **Case**. This is the test case number.
- **Operation**. This parameter specifies one of three values. **Copy** indicates that the tool under test is to copy from the source device directly to the destination device. **Image** specifies that the tool under test is to copy from the source device to an *image file* and then restore the *image file* to the destination device. **Image-RM** indicates that the tool under test should write an image file to removable media (e.g., tape). Test cases with **image-RM** are only executed if the tool under test has an intrinsic removable media functionality. Since **dd** does not have an intrinsic removable media feature, these cases were not executed.
- **Src** and **Dst**. The type of disk access interface for the source and destination drive is specified.
- **Rel Size.** The size relationship between the source and destination is specified.
- **Partition.** Specification of a partition type indicates that the test case is a partition operation. The parameter value *no spec* indicates a full disk operation.
- **Errors.** A value of *Error* indicates that a corrupted image file should be used. Since **dd** has no feature to detect a corrupted image file, these cases are not executed.

Table 1.2-1 Test Case Parameters

Case	Operation	Src	Dst	Rel Size	Partition	Errors
DI(LINUX)-01	copy	IDE	IDE	src < dst	no spec	none
DI(LINUX)-02	copy	IDE	IDE	src = dst	no spec	none
DI(LINUX)-03	copy	IDE	IDE	src > dst	no spec	none

Case	Operation	Src	Dst	Rel Size	Partition	Errors
DI(LINUX)-04	copy	IDE	IDE	src < dst	FAT16	none
DI(LINUX)-05	copy	IDE	IDE	src = dst	FAT32	none
DI(LINUX)-06	copy	IDE	IDE	src > dst	LINUX	none
DI(LINUX)-07	copy	SCSI	SCSI	src < dst	no spec	none
DI(LINUX)-08	copy	SCSI	SCSI	src = dst	no spec	none
DI(LINUX)-09	copy	SCSI	SCSI	src > dst	no spec	none
DI(LINUX)-10	copy	SCSI	SCSI	src < dst	NTFS	none
DI(LINUX)-11	copy	SCSI	SCSI	src = dst	FAT32	none
DI(LINUX)-12	copy	SCSI	SCSI	src > dst	FAT16	none
DI(LINUX)-13	copy	IDE	SCSI	src < dst	no spec	none
DI(LINUX)-14	copy	IDE	SCSI	src > dst	no spec	none
DI(LINUX)-15	copy	SCSI	IDE	src < dst	no spec	none
DI(LINUX)-16	copy	SCSI	IDE	src > dst	no spec	none
DI(LINUX)-17	image	IDE	IDE	src < dst	no spec	Error
DI(LINUX)-18	image	IDE	IDE	src < dst	no spec	none
DI(LINUX)-19	image	IDE	IDE	src = dst	no spec	Error
DI(LINUX)-20	image	IDE	IDE	src = dst	no spec	none
DI(LINUX)-21	image	IDE	IDE	src > dst	no spec	none
DI(LINUX)-22	image	IDE	IDE	src < dst	FAT32	Error
DI(LINUX)-23	image	IDE	IDE	src < dst	LINUX	none
DI(LINUX)-24	image-RM	IDE	IDE	src < dst	FAT16	Error
DI(LINUX)-25	image-RM	IDE	IDE	src < dst	NTFS	none
DI(LINUX)-26	image	IDE	IDE	src = dst	LINUX	Error
DI(LINUX)-27	image	IDE	IDE	src = dst	FAT32	none
DI(LINUX)-28	image-RM	IDE	IDE	src = dst	NTFS	Error
DI(LINUX)-29	image-RM	IDE	IDE	src = dst	FAT16	none
DI(LINUX)-30	image	IDE	IDE	src > dst	FAT32	none
DI(LINUX)-31	image-RM	IDE	IDE	src > dst	LINUX	none
DI(LINUX)-32	image	SCSI	SCSI	src < dst	no spec	Error
DI(LINUX)-33	image	SCSI	SCSI	src < dst	no spec	none
DI(LINUX)-34	image	SCSI	SCSI	src = dst	no spec	Error
DI(LINUX)-35	image	SCSI	SCSI	src = dst	no spec	none
DI(LINUX)-36	image	SCSI	SCSI	src > dst	no spec	none
DI(LINUX)-37	image	SCSI	SCSI	src < dst	FAT16	Error
DI(LINUX)-38	image	SCSI	SCSI	src < dst	NTFS	none
DI(LINUX)-39	image-RM	SCSI	SCSI	src < dst	LINUX	Error
DI(LINUX)-40	image-RM	SCSI	SCSI	src < dst	FAT32	none
DI(LINUX)-41	image	SCSI	SCSI	src = dst	NTFS	Error
DI(LINUX)-42	image	SCSI	SCSI	src = dst	LINUX	none
DI(LINUX)-43	image-RM	SCSI	SCSI	src = dst	FAT32	Error
DI(LINUX)-44	image-RM	SCSI	SCSI	src = dst	LINUX	none
DI(LINUX)-45	image	SCSI	SCSI	src > dst	FAT16	none
DI(LINUX)-46	image-RM	SCSI	SCSI	src > dst	FAT32	none
DI(LINUX)-47	image	IDE	SCSI	src < dst	no spec	Error
DI(LINUX)-48	image	IDE	SCSI	src < dst	no spec	none
DI(LINUX)-49	image	IDE	SCSI	src > dst	no spec	none
DI(LINUX)-50	image	SCSI	IDE	src < dst	no spec	Error
DI(LINUX)-51	image	SCSI	IDE	src < dst	no spec	none
DI(LINUX)-52	image	SCSI	IDE	src > dst	no spec	none

Table 1.2-2 documents the relationship between the test cases and test requirements defined in *Disk Imaging Tool Specification, Version 3.1.6*. The rows of the table

represent Linux test cases. The columns correspond to either a mandatory or an optional requirement. A bullet (●) indicates that the indicated requirement applies to the test case.

Table 1.2-2 Test Cases to Requirements Matrix

	Mandatory Requirements								**Optional Requirements**						
Case	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
01	●	●			●	●	●								
02	●	●			●	●									
03	●	●			●	●		●							
04	●	●			●	●	●				●				
05	●	●			●	●					●				
06	●	●			●	●		●			●				
07	●	●			●	●	●								
08	●	●			●	●									
09	●	●			●	●		●							
10	●	●			●	●	●				●				
11	●	●			●	●					●				
12	●	●			●	●		●			●				
13	●	●			●	●	●								
14	●	●			●	●		●							
15	●	●			●	●	●								
16	●	●			●	●		●							
17	●		●	●	●	●	●								
18	●	●			●	●	●								
19	●		●	●	●	●									
20	●	●			●	●									
21	●	●			●	●		●							
22	●		●	●	●	●	●				●				
23	●	●			●	●	●				●				
24	●		●	●	●	●	●				●				
25	●	●			●	●	●				●				
26	●		●	●	●	●					●				
27	●	●			●	●					●				
28	●		●	●	●	●					●				
29	●	●			●	●					●				
30	●	●			●	●		●			●				
31	●	●			●	●		●			●				
32	●		●	●	●	●	●								
33	●	●			●	●	●								
34	●		●	●	●	●									
35	●	●			●	●									
36	●	●			●	●		●							
37	●		●	●	●	●	●				●				
38	●	●			●	●	●				●				
39	●		●	●	●	●	●				●				

	Mandatory Requirements								Optional Requirements						
Case	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**1**	**2**	**3**	**4**	**5**	**6**	**7**
40	●	●			●	●	●				●				
41	●		●	●	●	●					●				
42	●	●			●	●					●				
43	●		●	●	●	●					●				
44	●	●			●	●					●				
45	●	●			●	●		●			●				
46	●	●			●	●		●			●				
47	●		●	●	●	●	●								
48	●	●			●	●	●								
49	●	●			●	●		●							
50	●		●	●	●	●	●								
51	●	●			●	●	●								
52	●	●			●	●		●							

1.3 Document Overview

Section 2 describes the test hardware, hard disk drives used and system configurations for running the tests. The procedures for creation or setup of source disks, DOS boot floppies, Linux boot drive and Jaz disk are described in section 3. The script files for each step are described in section 4. Section 5 describes the execution procedures. Technical difficulties encountered during testing are discussed in section 6. Guidelines for executing the tests in a different environment are presented in the last section.

2 Test Hardware

The tests were run on five host computers: **Cadfael, Morse, Rumpole, Wimsey** and **JudgeDee**. More than 20 hard drives (7 different models, 5 different brands) were used for the tests (Table 2.2-1). The tests were run with the hard drives arranged in one of five possible configurations (Table 2.3-1) as required by the test parameters (Table 1.2-1).

2.1 Host Computers

The hardware components listed in Table 2.1-1 are installed in all five computers.

Table 2.1-1 Host Computer Hardware Components

ASUS CUSL2 Motherboard Intel Pentium III (Coppermine) 933Mhz 512672k Memory Adaptec 29160N SCSI Adapter card Plextor CD-RW PX-W124TS Rev: 1.06 Iomega 2GB Jaz drive Rev: E.17 LS-120 SuperDisk Two slots for removable IDE hard disk drives Two slots for removable SCSI hard disk drive

The computers **Morse** and **Rumpole** each also have a 30GB OnStream SC30 tape drive (not used in the test procedures). The computer **JudgeDee** has a third slot for a removable IDE hard disk drive.

2.2 Hard Disk Drives

The hard disk drives used are listed in Table 2.2-1. These hard drives are mounted in CRU DataPort removable storage modules. Any combination of zero, one or two IDE hard drives and from zero, one or two SCSI hard drives can be installed in a host computer as required for a test.

The IDE disks have jumpers set for *cable select.* The SCSI ID for the SCSI disk is set to either 0 or 1 as required by the test case. Except as noted, a SCSI source disk is set to ID 0 and a SCSI destination disk is set to ID 1.

Table 2.2-1 Hard Disk Drives Used

Label	Disk Drive Model	Sectors	Interface	Capacity (GB)
A5	WDC WD200BB-00AUA1	39102336	IDE	20.02
A6	WDC WD200BB-00AUA1	39102336	IDE	20.02
A8	WDC WD200BB-00AUA1	39102336	IDE	20.02
A9	WDC WD200BB-00AUA1	39102336	IDE	20.02
AA	Maxtor53073H4	60030432	IDE	30.73
AB	Maxtor53073H4	60030432	IDE	30.73
AD	Maxtor53073H4	60030432	IDE	30.73
AE	Maxtor53073H4	60030432	IDE	30.73
CB	SEAGATE ST336705LC	71687370	SCSI	36.70
CC	SEAGATE ST336705LC	71687370	SCSI	36.70
CD	SEAGATE ST336705LC	71687370	SCSI	36.70
CE	SEAGATE ST336705LC	71687370	SCSI	36.70
E0	QUANTUM ATLAS10K2-TY092J	17938985	SCSI	9.18
E3	QUANTUM ATLAS10K2-TY092J	17938985	SCSI	9.18
E4	QUANTUM ATLAS10K2-TY092J	17938985	SCSI	9.18
E6	SEAGATE ST318404LC	35843670	SCSI	18.35
EA	SEAGATE ST39204LC	17921835	SCSI	9.17
EB	SEAGATE ST39204LC	17921835	SCSI	9.17
F5	IBM-DTLA-307020	40188960	IDE	20.57
F6	IBM-DTLA-307020	40188960	IDE	20.57
F7	IBM-DTLA-307020	40188960	IDE	20.57
F8	IBM-DTLA-307020	40188960	IDE	20.57
FA	IBM-DTLA-307045	90069840	IDE	46.11

2.3 Test Configurations

The hard drive setup is determined by the test case parameters described in Table 1.2-1 adapted from *Disk Imaging Tool Specification, Version 3.1.6*. Three disks are required for each test case, *source, destination* and *boot/media*. The source disk provides something to copy. The destination disk provides a place to put the copy. The boot/media disk has two functions. First, it provides the Linux environment for the execution of **dd**. Second, it provides a place to put the image file for test cases that require the creation of an image file.

The factors determining the source disk selection are the source disk interface and type of partition to copy. A disk is selected with the matching interface and partition type required for the test case.

The factors for the selection of the destination drive are the destination interface and the relative size parameters. A drive is selected with the specified interface and, for whole disk copies, size relative to the source. For partition copies, the actual size of the destination drive does not matter since it is the size of the partition on the destination that is relevant.

After the source and destination drives are selected, the boot/media disk is selected for one of the two remaining available drive slots.

The five system hard drive and boot configurations used for the tests are presented in Table 2.3-1. The *Source* column indicates where the source drive is mounted. Only the primary IDE channel was used. The drive was usually positioned as *drive 0*; however, a few test cases had the source drive positioned as *drive 1*. SCSI source drives were set to SCSI ID 0. The *Destination* column indicates the positioning of the destination drive. The *Boot/Media* column indicates the positioning of the destination drive. The *BIOS Boot Order* indicates the BIOS setting for Boot order required for the test so that a Linux environment is established for **dd** execution.

For test cases using system configuration 3, two host computers were used and the steps of the test procedure that are executed in a DOS environment are performed on a host system with a *BIOS boot order* of *Floppy, IDE*. The *execute* ***dd*** step is performed in a Linux environment obtained by booting from the SCSI disk (ID 0) on a different host with a *BIOS boot order* of *Floppy, SCSI.*

Table 2.3-1 System Configurations

ID	Source	Destination	Boot/Media	BIOS Boot Order
1	IDE primary 0	IDE primary 1	SCSI ID 0	Floppy, SCSI
2	SCSI ID 0	SCSI ID 1	IDE primary 0	Floppy, IDE
3	IDE primary 0	SCSI ID 1	SCSI ID 0	Floppy, SCSI (run **dd** step only)
4	SCSI ID 0	IDE primary 1	IDE primary 0	Floppy, IDE
5	IDE primary 1	SCSI ID 0	IDE primary 0	Floppy, IDE

3 Media Setup

The test cases require several media components to be created and setup for the test cases to be executed. The following items need to be setup one time.

1. Source hard disk drives for the test cases.
2. A DOS Boot floppy that creates the run-time environment for the test case setup and measurement.
3. A removable hard disk drive that contains a bootable Linux system.

4. A Jaz disk that contains support software, control scripts, log files and utility software.

In addition to the components that are setup once, a destination hard drive must be setup for each test case.

3.1 Source Disks

The source disks play the role of hard drives containing original digital evidence that must be preserved. There are too many possible disk layouts for all to be used in the tests. Three configurations were selected that cover the most common partition types. The first configuration is a dual boot Red Hat Linux 7.1 and Windows Me. This configuration also includes FAT16, Linux EXT2, hidden partitions and deleted partitions (Table 3.1-1). The second configuration is a Windows 2000 system with both FAT32 and NTFS file systems (Table 3.1-2). The third configuration does not contain a valid partition table. All partitions are created with **partition magic Pro 6.0**.

Table 3.1-1 Windows Me/Linux Source Drive Layout

Type	Size (MB)	Comment
FAT16	600	Windows Me C drive
none	500	Unallocated Space
Extended	3500	Extended partition containing the next four partitions
EXT2	100	Linux EXT2 partition
FAT16	70	D drive for Windows
FAT16	2000	A FAT16 partition that has been deleted
FAT16	90	A FAT16 partition marked as *hidden*
EXT2	3000	Linux EXT2 partition with Red Hat 7.1
none	variable	Unallocated space up to the next partition
SWAP	200	Linux Swap partition

Table 3.1-2 Windows 2000 Source Drive Layout

Type	Size (MB)	Comment
FAT32	3000	Windows 2000 C drive
none	1000	Unallocated space
Extended	variable	Remainder of disk space
NTFS	1000	Deleted NTFS partition
NTFS	600	D drive
FAT32	1000	Deleted FAT32 partition
NTFS	800	Hidden NTFS partition
none	variable	Unallocated space up to next partition
FAT32	600	Hidden FAT32 partition

The setup procedure for a source disk is as follows:

1. Select type of setup: Windows Me/Linux, Windows 2000 or none. Disks E3 and F5 were given the Windows Me/Linux layout; disks E4 and F6 were given the Windows 2000 Layout; disk CC was setup without a partition table.
2. Select a hard drive
3. Select computer, install drive, boot into PC DOS 6.3 from a boot floppy.
4. Run **LOGSETUP** to make a record of the setup.
5. Run **DISKWIPE** to initialize the drive contents.
6. If the setup type uses an operating system do steps 7-9
7. Run **partition magic** to partition the drive. For Windows Me/Linux source disk use the script in Table 3.1-3 and for Windows 2000 source disk use the script in Table 3.1-4.
8. Follow the installation instructions for each operating system that should be installed. For a Windows Me/Linux configuration, first install Windows Me then install Linux. For a Windows 2000 configuration, install Windows 2000.
9. Create deleted files. This is accomplished by a script (DOS batch file) that creates a directory (X:\UDT, where X is a drive letter) with deleted files and a deleted subdirectory (Table 3.1-5).
10. Run **DISKHASH** to create a reference SHA-1 hash for the source disk.
11. (optional) If practical, create a backup to another disk that can be used to restore the disk if it is modified. If a disk needs to be restored, the hash value can be recomputed to verify that the backup and restore were successful.

Table 3.1-3 Partition Magic script for Windows Me/Linux Source (`FAT-SRC.TXT`)

```
Select Drive 1
Select Unallocated First
Create /FS=FAT /Size=600 /Label="P1FAT"
Select Unallocated First
Create /FS=Extended /Size=4000
Select Partition Extended
Resize Left Boundary Smaller 500
Select Unallocated 2
Create /FS=LINUXEXT2 /Size=100 /Label="X1Unix"
Select Unallocated 2
Create /FS=FAT /Size=70 /label="X1Fat"
Select Unallocated 2
Create /FS=FAT /Size=2000 /label="GONE"
Select Unallocated 2
Create /FS=FAT /Size=90 /label="GHOST"
Select Unallocated 3
Create /FS=LINUXEXT2 /Size=3000 /Label="Unix"
Select Unallocated 3
Create /FS=LINUXswap /Size=200  /Position=END
Select Partition "GHOST"
Hide
Select Partition "GONE"
Delete "GONE"
Select Partition "P1FAT"
Set Active
```

Table 3.1-4 Partition Magic script for Windows 2000 Source (`NT-SRC.TXT`)

```
Select Drive 1
Select Unallocated First
Create /FS=FAT32 /Size=3000 /Label="FAT3GB"
Select Unallocated First
Create /FS=Extended
Select Partition Extended
Resize Left Boundary Smaller 1000
Select Unallocated 2
Create /FS=NTFS /Size=1000 /label="GONE1"
Select Unallocated 2
Create /FS=FAT32 /Size=600 /Label="GHOST32" /position=end
Select Unallocated 2
Create /FS=NTFS /Size=600 /label="X1NT"
Select Unallocated 2
Create /FS=FAT32 /Size=1000 /label="GONE2"
Select Unallocated 2
Create /FS=NTFS /Size=800 /label="GHOST4NT"
Select Partition "GHOST4NT"
Hide
Select Partition "GHOST32"
Hide
Select Partition "GONE2"
Delete "GONE2"
Select Partition "GONE1"
Delete "GONE1"
Select Partition "FAT3GB"
Set Active
```

Table 3.1-5 Script to Create Deleted Files (`UDT-SET.BAT`)

```
echo undelete test setup
Rem Setup a directory with some deleted files and a deleted
subdirectory
date
time
:L1
Rem are we done?
      if "%1"=="" goto L1X
      echo "Set up drive %1:"
rem   create a directory for the deleted files
      mkdir %1:\udt
rem   create two files
      copy a:readme.txt %1:\udt
      copy a:back.txt %1:\udt
rem   delete one file
      del %1:\udt\back.txt
rem   undelete %1:\udt
rem   create a subdirectory
      mkdir %1:\udt\sub
Rem   create some files in the subdirectory
      copy a:missing.txt %1:\udt\sub
      copy a:gone.txt %1:\udt
rem   delete one file
```

```
      del %1:\udt\sub\missing.txt
rem   delete the directory
      rmdir %1:\udt\sub
rem   delete another file
      del %1:\udt\gone.txt
rem   shift cmd line, look for another drive
      shift
      goto L1
:L1X
echo Setup finished
```

3.2 DOS Boot Floppy

The DOS floppy disk provides the execution environment for the support software. The commands to setup the DOS floppy are presented in Table 3.2-1. The DOS boot floppy is used for source drive setup, destination drive setup and for measuring the results of a test run.

Table 3.2-1 DOS Boot Floppy Setup Procedure

From a PC DOS 6.3 System, insert a blank floppy disk

```
FORMAT A: /S
MKDIR A:\ASPI
MKDIR A:\GUEST
MKDIR A:\MISC
COPY HIMEM.SYS A:\MISC
COPY MSCDEX.EXE A:\MISC
COPY MOUSE.COM A:\MISC
COPY MOUSE.INI A:\MISC
COPY SMARTDRV.EXE A:\MISC
COPY GUEST.EXE A:\GUEST
COPY GUEST.INI A:\GUEST
COPY GUESTHLP.TXT A:\GUEST
COPY ASPI8U2.SYS A:\ASPI
COPY ASPICD.SYS A:\ASPI
```

setup AUTOEXEC.BAT and CONFIG.SYS

The AUTOEXEC.BAT file presented in Table 3.2-2 is a simplified version of a typical forensic boot floppy based on the recommendations in the SafeBack 2.0 manual.

Table 3.2-2 DOS AUTOEXEC.BAT

```
@ECHO OFF
PROMPT $p$g
A:\misc\MOUSE
A:\misc\smartdrv
a:\misc\mscdex.exe /d:aspicd0 /L:Z /m:12
a:\guest\guest letter=x
```

The contents of the CONFIG.SYS is presented in Table 3.2-3.

Table 3.2-3 DOS CONFIG.SYS

```
device=A:\misc\himem.sys
dos=high,umb
lastdrive=z
FILES = 30
BUFFERS = 8
device=a:\aspi\aspi8u2.sys /D /PD800 /Q9
device=a:\aspi\aspicd.sys /d:aspicd0
```

3.3 Jaz Disk

The Jaz disk serves a number of functions. The test case scripts, support programs, and utility programs (e.g., **partition magic**) are located on the Jaz disk. Log files for each test case are kept on the Jaz disk. The disk is setup by the script in Table 3.3-1. The disk labeled *FA* is not used in any test cases but as a repository for files needed to execute the test cases. Drive *%2* referred to in Table 3.3-1 is *FA*. Drive *%1* is the Jaz disk.
One Jaz disk was setup for each host.

The Jaz disk is setup with several subdirectories. The content of each subdirectory is described in Table 3.3-2. In addition to tools required for running the tests, some additional utility programs were also placed on the Jaz disk.

Table 3.3-1 Jaz Disk Setup (SETUPJAZ.BAT)

```
Rem Script to setup a Jaz disk for dd testing
rem
rem
echo Copy files from drive %2 to JAZ disk on drive %1
rem
rem Create direstories ...
rem support software, partition magic, dos scripts: pre & post, linux
scripts
rem safeback, other utility software and log files
rem
mkdir %1:\ss
mkdir %1:\pm
mkdir %1:\pre
mkdir %1:\post
mkdir %1:\xdd
mkdir %1:\scripts
mkdir %1:\sb
mkdir %1:\util
mkdir %1:\logs
rem
rem copy files to directories
rem
copy %2:\projects\nij\docs\new-test-cases\scripts\pre-*.bat %1:\pre
copy %2:\projects\nij\docs\new-test-cases\scripts\post-*.bat %1:\post
copy %2:\projects\nij\docs\new-test-cases\scripts\xdd-* %1:\xdd
copy %2:\projects\nij\docs\new-test-cases\dd-scripts\*.* %1:\scripts
copy %2:\projects\nij\docs\new-test-cases\magic\*.* %1:\pm
copy %2:\projects\nij\docs\new-test-cases\util\*.* %1:\util
copy %2:\projects\nij\docs\new-test-cases\sb\*.* %1:\sb
copy %2:\projects\nij\dosbios\zbios\*.exe %1:\ss
```

Table 3.3-2 Contents of the Jaz Disk

Subdirectory	Contents
ss	The support software (FS-TST) programs.
pm	A copy of **partition magic** and the partition magic scripts.
pre	The pre-execution setup scripts for each test case.
post	The post-execution setup scripts for each test case.
xdd	The **xdd** scripts for each test case.
scripts	**csh** scripts to setup boot drive and to run **dd**
sb	A copy of **safeback**. (not required for the tests)
util	A text editor. (not required for the tests)
logs	for each test case a subdirectory to contain the log files.

3.4 Linux Boot Drive

The Linux hard drives are used to provide the execution environment for **dd** and to provide a place to store the image file for test cases that require an image file. The following procedure was used to setup the Linux boot drives.

1. Run LOGSETUP to make a record of the setup.

2. Run DISKWIPE to ensure that the disk has no residual information.
3. Load Linux OS (Red Hat 7.1)
4. Mount a Jaz disk with the setup scripts
5. Run **make-test-account** from the mounted Jaz disk.

The **make-test-account** (Table 3.4-1) script is a **csh** (C shell) script that creates an account for executing **dd** for a given test case. The Jaz disk created in section 3.3 must be mounted as **/x**. The account is created by line 8 (**useradd** program). The account is created with *root* (user id of 0) privilege. Lines 10 and 11 create a login script (**.login**) for the account. Line 12 copies two scripts that the login script uses to locate and mount the Jaz drive for the test runs.

Five hard drives were setup as Linux boot/media drives (AD, AE, CB, CD and CE).

Table 3.4-1 Script to create test account: make-test-account (`make-test-account`)

```
1. #!/bin/csh -f
2. # create an account to run dd test cases
3. # usage: make-test-account account_name
4. #
5. # Note: the Jaz disk must be mounted on /x
6. #
7. # create the account
8. /usr/sbin/useradd -c 'DD test run' -s /bin/csh -u 0 -o $1
9. # setup .login
10.   cp /x/scripts/run-test /home/$1/.login
11.   cat /x/scripts/run-test-done >> /home/$1/.login
12.   cp /x/scripts/{mount-jaz,find-jaz} /home/$1
13.   # set account password
14.   echo "Set password for $1"
15.   passwd $1
```

Table 3.4-2 Test Account Login Script (`.login`)

```
#!/bin/csh -f
# .login file for test account
./mount-jaz
cat /x/case.txt
set path = (. /x/xdd /x/scripts $path)
echo -n "Enter Case Number: "
set case = $<
echo -n "Enter Host name: "
set host = $<
echo -n "Enter operator name: "
set oper = $<
echo "Case $case"
@ case_no = $case
if ($case_no < 10) then
      set c = "0$case_no"
else
      set c = $case_no
```

```
endif
echo "The case is $c"
set log_dir = /x/logs/lx-$c
if (  ( -d $log_dir ) ) then
@ n = 0
while ( -d $log_dir-$n )
      @ n = $n + 1
end
mv $log_dir $log_dir-$n
endif
mkdir $log_dir
echo -n "Test case LX-$c run on $host by $oper at " >
$log_dir/config.txt
date >> $log_dir/config.txt
dmesg | egrep '([sh]d[abc])|(Vendor)' | egrep '(sector)|(Vendor)' >>
$log_dir/config.txt
df >> $log_dir/config.txt
source /x/xdd/xdd-$c
# part II of .login file
# shut down computer after test case
echo "Test case $case Done"
umount /x
echo "Unmount /x"
sleep 5
/sbin/shutdown -h now
```

4 Test Execution Scripts

The test cases are executed as a sequence of three steps. The first step (setup) sets up the destination drive from a DOS environment and logs the partition tables of the source and the destination disks. The second step (execute) is to execute **dd** from a Linux environment. The last step (measure), executed in a DOS environment, compares the source to the destination sector-by-sector and uses a SHA-1 hash to check the source for any changes. Except for partition creation and formatting, the programs required to setup each test case and to measure the results are contained in the FS-TST package. Shell scripts and DOS batch files are stored on the Jaz disk for each test case.

4.1 Pre-execution Setup

The destination drive setup scripts are stored on the Jaz drive in the directory `X:\PRE.` The script for text case *XX* is named `X:\PRE\PRE-XX.BAT`. Table 4.1-1 is an example of a pre-execution script that creates a FAT16 partition on the destination.

Table 4.1-1 Example Pre-execution Script (`PRE-04.BAT`)

```
1.  @ECHO OFF
2.  REM Host Operator Src Dst Boot/Media
3.  del a:*.txt
4.  X:\ss\logcase LX-04 %1 %2 80:%3:hda1 81:%4:hdb1 %5:sda
5.  copy A:\case.txt X:\case.txt
6.  X:\ss\diskwipe LX-04 %1 81 %4 /noask /dst /new_log /comment "%2"
7.  echo X:\ss\partab LX-04 %1 80 /all /new_log /comment %2(%3) >> A:\autoexec.bat
8.  echo X:\ss\partab LX-04 %1 81 /all /new_log /comment %2(%4) >> A:\autoexec.bat
9.  echo echo Shutdown and insert Linux boot drive >>A:\autoexec.bat
10. echo copy A:\clean.bat A:\autoexec.bat >>A:\autoexec.bat
11. X:\PM\PQMAGIC /cmd=X:\PM\F16-SS-2.txt
12. echo Reboot if you see this message
```

The script was executed by the following command line:

`x:\pre\pre-04 Cadfael JRL F5 A6 CD`

All of the pre-execution scripts take five command line parameters: host name, operator, source disk drive hex label, destination disk drive hex label, and Linux boot drive hex label. A line by line explanation of the script is as follows:

1. Turn off echo so that the commands are not echoed to the display screen.
2. A comment to document the command line parameters. *Host* (referred to as %1 in the script) is the name of the host computer. *Operator* (%2) identifies the staff member running the test. *Src, Dst* and *Boot/Media* (%3, %4 and %5) are the external, two hex digit label assigned to each hard disk drive.
3. Delete all the text files. All the log files are created as text files. After all programs required for the test case have been run, the last step is to copy all the text files to a directory created for the given test case. Deleting the text files removes the log files from the previous test case.
4. The **LOGCASE** program makes a record of drive assignments for the test case.
5. Putting a copy of the **CASE.TXT** file on the Jaz drive allows the next step (in the Linux environment) to automatically determine the test case being run.
6. **DISKWIPE** writes a known, unique content to each sector of the destination disk.
7. Modify the **AUTOEXEC.BAT** so that the next time the system is booted the partition table of the source disk is logged.
8. Modify the **AUTOEXEC.BAT** so that the next time the system is booted the partition table of the destination disk is logged.
9. Modify the **AUTOEXEC.BAT** so that the next time the system is booted and the partition tables of the source and destination are logged that a message is printed to instruct the operator to shutdown and go on to the next step.
10. Modify the **AUTOEXEC.BAT** so that the next time the system is booted the **AUTOEXEC.BAT** file is restored to a clean version.
11. Use **partition magic** to create the partition on the destination disk required for the test case. Test cases that do not require a partition on the destination omit this step.
12. Usually **partition magic** automatically reboots a computer after creating a partition, but not always. This message is to remind the operator to reboot so that the partition tables are logged if the computer is not automatically rebooted.

4.2 Execute dd

The **dd** execution step is invoked automatically by logging on to the Linux test account. The login script (Table 3.4-2) prints the test case number from the **CASE.TXT** file copied to the Jaz disk during the setup step. The operator is prompted to enter the test case number, operator identification and the host name, then **dd** is executed from the script corresponding to the test case. For test case ***xx***, the script name is **/x/xdd/xdd-*xx*.** The computer is automatically shut down after the execute step finishes.

The actual execution of **dd** is accomplished through several layers of **csh** scripts. At the top is the **xdd** script. It is a one line script that sets four parameters and invokes the next level via **run_dd**. For example, **`run_dd hda hdb copy LX-01`** is the **xdd** script for test case 01. The four parameters are as follows:

1. The source device is the first parameter. In the example, the full device name would be **/dev/hda**. Some possible values are **hda** for disk 0 on the primary IDE channel, **sda** for a disk on SCSI ID 0.
2. The second parameter is the name of the destination device, **/dev/hdb** in this case.
3. The third field should be either **copy** or **image** as required by the test case.
4. The last field is the test case ID.

The **run_dd** script (Table 4.2-1) selects the type of operation (copy or image) and invokes the next layer. If the third parameter is **copy** then the script **dd_copy** (Table 4.2-2) is invoked to directly copy the source to the destination. If the third parameter is **image**, then the script **dd_image** (Table 4.2-3) is invoked to first copy the source to an image file and then to copy the image file to the destination.

The **dd_image** script calls **dk-backup** (Table 4.2-4) to copy the source to the image file and **dk-restore** (Table 4.2-5) to copy the image file to the destination. The script **dk-backup** creates a set of compressed image files of the source. Each image file usually contains a maximum of 1,000,000 sectors.[*] Both **dk-backup** and **dk-restore** output status information to allow the operator to track test run progress.

Table 4.2-1 Script to Select Type of Operation (`run_dd`)

```
#!/bin/csh -f
# run_dd source_device dst_device function Case_number
#
set src = $1
set dst = $2
set func = $3
set log_dir = /x/logs/$4
if ($func == "image") then
    echo Image /dev/$src to file to /dev/$dst
    dd_image /dev/$src /dev/$dst $4 /tmp
else if ($func == "image-RM") then
    echo Image /dev/$src to file on removable media to /dev/$dst
```

[*] Some image files contain 4,000,000 sectors. The larger value was used for the initial test runs but was changed for some test cases to monitor the test process more closely. Changing the number of sectors in the image file is accomplished by changing line 15 to **@ max = 4000000,** with a corresponding change in **dk-restore**.

```
13.    dd_image /dev/$src /dev/$dst $4 /x
14. else if ($func == "copy") then
15.    echo Copy /dev/$src to /dev/$dst
16.    dd_copy /dev/$src /dev/$dst $4
17. else
18.    echo "[$func] is invalid as a function code"
19. endif
```

A line by line explanation of **run_dd** follows:

1. The script is executed in the **csh** environment, the **.cshrc** is skipped.
2. Comment describing the command line.
3. Comment
4. Set a variable, **src**, to the first command line parameter (source device name).
5. Set a variable, **dst**, to the second command line parameter (destination device name).
6. Set a variable, **func**, to the third command line parameter (operation: copy or image).
7. Set a variable, **log_dir**, to locate any log files on the Jaz drive in a directory named for the test case.
8. Test for type of operation
9. Give feedback to the operator.
10. Invoke **dd_image** script to make an image of the source and restore the image to the destination.
11. Test for removable media test. However, the removable media tests were not used.
12. Give feedback to the operator.
13. Invoke **dd_image** script to make an image on removable media.
14. Test for the copy operation.
15. Give feedback to the operator.
16. Invoke **dd_copy** to copy the source to the destination.
17. **else** indicates the function is incorrect.
18. Tell the operator something is wrong.
19. Done.

Table 4.2-2 Script to invoke dd for direct copy (dd_copy)

```
1.  #!/bin/csh -f
2.  # Script to copy disk to disk
3.  # dd_copy source_device destination_device Test_case
4.  # log results to /x/logs/test_case/copy_log.txt
5.  #
6.  set src = $1
7.  set dst = $2
8.  set log_dir = /x/logs/$3
9.  echo -n "Start ${3}: " > $log_dir/copy_log.txt
10. date >> $log_dir/copy_log.txt
11.
12. set cmd = "dd if=$src of=$dst bs=1b"
13. echo "Command: $cmd" >> $log_dir/copy_log.txt
14. $cmd >>& $log_dir/copy_log.txt
15.
16. echo -n "Finish: " >> $log_dir/copy_log.txt
17. date >> $log_dir/copy_log.txt
```

A line by line explanation of **dd_copy** follows:

1. The script is executed in the **csh** environment, the **.cshrc** is skipped.

2. Comment
3. Comment describing the command line.
4. Comment
5. Comment
6. Set a variable, **src**, to the first command line parameter (source device name).
7. Set a variable, **dst**, to the second command line parameter (destination device name).
8. Set a variable, **log_dir**, to locate any log files on the Jaz drive in a directory named for the test case.
9. Record the start time.
10. Record the date.
11. Comment
12. Set a variable, **cmd**, to the **dd** command to execute.
13. Log the **dd** command to a file.
14. Execute the **dd** command (created in step 12).
15. Comment
16. Record finish time.
17. Record the date.

Table 4.2-3 Script to use dd to copy via an image file (`dd_image`)

```
1.  #!/bin/csh -f
2.  # dd_image src dst case image_directory
3.  ##  dd_image /dev/$src /dev/$dst $4 /tmp
4.  #   echo "Usage: dk-backup src_dev test_case image_directory
5.  #
6.  set src = $1
7.  set dst = $2
8.  set log_dir = /x/logs/$3
9.  set log_file = $log_dir/$3_log.txt
10. echo -n "Start ${3}: " > $log_file
11. date >> $log_file
12. echo "dd_image $argv" >>$log_file
13.
14. echo "Running backup, Log $log_file"
15. dk-backup $src $3 $4 $log_file
16. set log_file = $log_file:r-r.txt
17. echo "Running: restore, Log: $log_file"
18. dk-restore $dst $3 $4 $log_file
19. echo "Backup/restore finished"
20.
21. echo -n "Finish: " >> $log_file
22. date >> $log_file
```

A line by line explanation of **dd_image** follows:
1. The script is executed in the **csh** environment, the **.cshrc** is skipped.
2. Comment
3. Comment describing the command line.
4. Comment
5. Comment
6. Set a variable, **src**, to the first command line parameter (source device name).
7. Set a variable, **dst**, to the second command line parameter (destination device name).

8. Set a variable, **log_dir**, to locate any log files on the Jaz drive in a directory named for the test case.
9. Set the name of the log file for the backup.
10. Record the start time.
11. Record the date.
12. Record command in log file.
13. Comment
14. Give feedback to the operator.
15. Create the image file
16. Set the log file for the restore.
17. Give feedback to the operator.
18. Restore from the image to the destination.
19. Give feedback to the operator.
20. Comment
21. Record finish time.
22. Record the date.

Table 4.2-4 Script to use dd to create an image file (`dk_backup`)

```
#!/bin/csh -f
if ($#argv != 4) then
    echo "Usage: $0 src_dev test_case image_directory log_file
    exit (1)
endif
#set echo
set src = $1
set dst_dir = $3/img
set log_file = $4
echo "dk-backup $argv" >>$log_file
echo "dk-backup $argv"
#rm -r -f $dst_dir
mkdir $dst_dir
@ total = 1
@ max = 1000000
@ n = $max
@ skip = 0
@ pass = 100
#@ nf = ($total / $max) + 1
#echo "$nf image files required"
set more = 1
while ($more )
    @ pass = $pass + 1
    echo "Pass: $pass"
    set dst = $dst_dir/${2}-image.$pass
    date > $dst_dir/${2}-log.$pass
    set cmd = "(dd if=$src skip=$skip count=$n bs=1b | gzip > $dst.gz) >& $dst_dir/this_pass"
    echo "$cmd" >> $dst_dir/${2}-log.$pass
    (dd if=$src skip=$skip count=$n bs=1b | gzip > $dst.gz) >>& $dst_dir/this_pass
    echo "Pass: $pass Status: $status" >> $log_file
    echo "Cmd: $cmd"
    cat $dst_dir/this_pass >> $dst_dir/${2}-log.$pass
    cat $dst_dir/this_pass >>$log_file
#   dd if=$src of=$dst skip=$skip count=$n bs=1b >>& $dst_dir/${2}-log.$pass
    date >> $dst_dir/${2}-log.$pass
    @ skip = $skip + $n
    grep ’^0+0’ $dst_dir/${2}-log.$pass >/dev/null
    set more = $status
end
echo -n "Delete empty log: "
```

```
41. ls $dst_dir/*${pass}*
42. cat $dst_dir/*-log.* > $log_file:r-b-all.txt
```

A line by line explanation of **dk_backup** follows:

1. The script is executed in the **csh** environment, the **.cshrc** is skipped.
2. Test for correct number of parameters
3. Operator feedback describing the command line.
4. Quit.
5. End of the **if**
6. Commented out statement.
7. Set a variable, **src**, to the first command line parameter (source device name).
8. Set a variable, **dst_dir**, to the third command line parameter (image file directory name).
9. Set a variable, **log_file**, to the log file name.
10. Log action.
11. Give feedback to the operator.
12. Commented out statement.
13. Create directory for image file.
14. Dead code. Could be deleted.
15. Set the maximum size of an image file (1,000,000 sectors)
16. Set **n** to **max**
17. Set the number of sectors to skip for the first pass.
18. Start numbering at 100 to force pass value to be three digits.
19. Dead code. Could be deleted.
20. Dead code. Could be deleted.
21. **more** indicates if there is more of the source to get (1) or not (0).
22. Loop as long as there is more source to get.
23. Set the current pass number
24. Give feedback to the operator.
25. Set the full path to the image file.
26. Log date and time
27. Create the command to execute **dd**. The image is compressed with **gzip** to save space.
28. Log the command.
29. Execute the **dd** command.
30. Log the pass.
31. Give feedback to the operator.
32. Add logs for this pass.
33. Add logs for this pass to debug file.
34. Dead code. Could be deleted.
35. Log the date.
36. Adjust the number of sectors to skip on next pass.
37. Test for no more sectors left on source.
38. Set **more** to zero if source completely imaged.
39. End of the while loop.
40. Give feedback to the operator …

41. listing the image files created.
42. Finalize the backup log file

Table 4.2-5 Script to use dd to restore an image file (`dk_restore`)

```
1.  #!/bin/csh -f
2.  if ($#argv != 4) then
3.      echo "Usage: $0 dst_dev test_case image_directory log_file
4.      exit (1)
5.  endif
6.  #set echo
7.  echo "dk-restore $argv"
8.  set dst = $1
9.  set dst_dir = /$3/img
10. set log_file = $4
11. echo "dk-restore $argv" >> $log_file
12. @ total = 1
13. @ max = 1000000
14. @ n = $max
15. @ skip = 0
16. @ pass = 101
17. set more = 1
18. while ($more )
19.     echo "Restore pass $pass"
20.     set src = $dst_dir/${2}-image.$pass.gz
21.     date > $dst_dir/${2}-rlog.$pass
22.     set cmd = "(gunzip <$src | dd of=$dst seek=$skip count=$n bs=1b ) >>&
    $dst_dir/${2}-rlog.$pass"
23.     echo "$cmd" >> $dst_dir/${2}-rlog.$pass
24.     (gunzip <$src | dd of=$dst seek=$skip count=$n bs=1b ) >>& $dst_dir/this_pass
25.     echo "Pass: $pass Status: $status" >> $log_file
26.     echo "Cmd: $cmd"
27.     cat $dst_dir/this_pass >> $dst_dir/${2}-rlog.$pass
28.     cat $dst_dir/this_pass >> $log_file
29.     date >> $dst_dir/${2}-rlog.$pass
30.     @ skip = $skip + $n
31.     @ pass = $pass + 1
32.     if (-e $dst_dir/${2}-image.$pass.gz ) then
33.             set more = 1
34.     else
35.             set more = 0
36.     endif
37. end
38. cat $dst_dir/*-rlog.* > $log_file:r-r-all.txt
```

A line by line explanation of **dk_restore** follows:

1. The script is executed in the **csh** environment, the **.cshrc** is skipped.
2. Test for correct number of parameters
3. Operator feedback describing the command line.
4. Quit.
5. End of the **if**
6. Commented out statement.
7. Give feedback to the operator.
8. Set a variable, **dst**, to the destination device name.
9. Set a variable, **dst_dir**, to the name of the directory containing the image file.
10. Set the name of the log file.
11. Log action.
12. Dead code. Could be deleted.

13. Set the maximum size of an image file (1,000,000 sectors)
14. Set **n** to **max**
15. Set the number of sectors to skip for the first pass.
16. Start numbering at 101 to force pass value to be three digits.
17. **more** indicates if there is more of the source to get (1) or not (0).
18. Loop as long as there are more image files.
19. Give feedback to the operator.
20. Set the full path to the image file.
21. Log date and time
22. Create the command to execute **dd**. The image is compressed with **gzip** to save space.
23. Log the command.
24. Execute the **dd** command.
25. Log the pass.
26. Give feedback to the operator.
27. Add logs for this pass.
28. Add logs for this pass to debug file.
29. Log the date.
30. Adjust the number of sectors to skip for the next pass.
31. Adjust the pass number.
32. Test for more image files
33. Set **more** to 1 if more image files exist.
34. **else**
35. Set **more** to 0 if no more image files.
36. End of the **if**.
37. Consolidate the log files.

4.3 Post-execution Measure

The measurement scripts are stored on the Jaz drive in the directory `X:\POST.` The script for text case *XX* is named `X:\POST\POST-XX.BAT`. Table 4.3-1 is an example of a post execution script. The post execution scripts have three functions, compare the source to the destination, compute a SHA-1 hash of the source and save all log files to a directory.

Table 4.3-1 Example Post Execution Measurement Script (`POST-04.BAT`)

```
REM Host Operator Src Dst
X:\ss\partcmp LX-04 %1 80 %3 81 %4 /new_log /comment "%2" /select 1 1
X:\ss\diskhash LX-04 %1 80 /comment "%2 (%3)" /new_log /after
a:
cd \
mkdir a:\LX-04
copy *.TXT LX-04
copy A:\*.TXT X:\logs\LX-04
echo Case: LX-04 finished
```

The script was executed by the following command line:
`x:\post\post-04 Cadfael JRL F5 A6`

All of the post-execution scripts take four command line parameters: host name, operator, source disk drive hex label, and destination disk drive hex label. A line by line explanation of the script is as follows:

1. Comment to explain the command line parameters. First the host name (%1), then the operator ID (%2) and last the external hex labels for the source (%3) and destination (%4) drives.
2. Compare the source to the destination. If the test case is a partition copy then **PARTCMP** is executed. If the test case is a disk copy then **DISKCMP** is executed.
3. Compute the SHA-1 hash of the source disk.
4. Make the floppy disk the current drive.
5. Make the root directory current.
6. Create a directory for the test case log files.
7. Copy the log files from the floppy to the test case directory.
8. Copy log files from the floppy to the Jaz drive
9. Notify the operator that the test case is finished.

After the measurement script finishes, the log files should be copied to a permanent location.

5 Test Case Execution

This section presents the procedures for running a test case. It is assumed that the reader is familiar with basic computer operation.

5.1 Execution Procedure

The procedure to execute a **dd** test case is as follows:

1. Select the test case to run.
2. Collect removable media: DOS Boot floppy, Jaz disk.
3. Select a source disk based on the test case parameters. The *source interface* parameter determines if the disk is IDE or SCSI. If a partition type is specified then a source disk with the specified setup is selected.
4. Select a destination disk for the test based on the test case parameters. The *destination interface* determines if the disk is IDE or SCSI. The *relative size* parameter determines acceptable selections for test cases that operate on an entire disk. For partition operations, any size disk can be used.
5. Select a system configuration (Table 2.3-1 System Configurations) based on the *source interface* and *destination interface* test case parameters.
6. Select a host computer to run the test. Ensure that the BIOS boot order is set as required by the selected system configuration.
7. Select a boot/media disk based on the selected system configuration.
8. Ensure that the host computer is off. Install DOS boot disk, Jaz disk, source disk and destination disk. Do not install the boot/media hard drive yet.
9. Turn on the host computer to boot from forensic DOS floppy.
10. Run the pre-execution script (section 4.1) to setup the destination hard drive and log the partition tables of the source and destination drives. The full path name of the script is **X:\PRE\PRE-xx**, where **xx** is the two digit test case number. The script is on the Jaz disk (DOS drive X:). The host computer may reboot if a partition is created on

the destination. If the host does not automatically reboot then the host should be manually rebooted (to log the partition tables). There was a problem with this step that required a procedural modification, see Section 6 for details.

11. Turn off the system and remove the DOS boot floppy.
12. Install a boot/media hard drive with the Linux boot/execution environment, set the BIOS (if needed) to boot from this disk. Turn on the host computer and boot to Linux.
13. Logon to the test run account. The login script prompts for the test case number and then the script for the test case is run automatically. The computer will shutdown when the case finished.
14. Remove the Linux boot/media hard drive.
15. Insert the DOS boot floppy.
16. Turn on the host computer to boot into DOS from the boot floppy.
17. Run the post-execution script (section 4.3) to measure the results. The script compares the source to the destination and computes a SHA-1 for the source disk. The full path name of the script is **X:\POST\POST-xx**, where **xx** is the two digit test case number. The script is on the Jaz disk (DOS drive X:).
18. After the measurement script finishes, the log files should be copied to a permanent location.

A note about BIOS boot order: The support software (**pre-xx** and **post-xx**), requires that IDE drives have lower drive numbers than any installed SCSI drives. This can be accomplished by ensuring that IDE drives are seen before SCSI drives in boot order. Therefore, the **pre-xx** step must be run with *IDE first boot order*, if both IDE and SCSI drives are present, but the **dd** (Linux environment) may need to be booted from a SCSI disk (i.e., *SCSI first boot order*).

5.2 Special Procedure for non-FAT Partitions

The procedure used to determine if a tool has modified any of the excess sectors of a partition copy where the destination is larger than the source depends on the type of partition copied. If a FAT partition is copied, the following procedure is sufficient to determine if a tool has changed any of the excess sectors:

1. Initialize the destination disk with the **diskwipe** program.
2. Create the destination partition.
3. Run the tool being tested.
4. Compare the source partition to the destination partition with the **partcmp** program.

Creating a FAT partition overwrites some sectors near the beginning of the partition with the *file allocation table*, but leaves most of the sectors in the partition alone, i.e., the excess sectors of the partition retain the original content produced by **diskwipe**. The **partcmp** program examines the excess sectors of the destination and assigns each sector to one of several possible categories. The **partcmp** program then writes the number of sectors in each category to a log file. Any sectors that are changed by the tool are counted in a category other than the *destination filled* category. If the tool has not changed any of the excess sectors, all of the excess sectors are assigned to the *destination filled* category.

This procedure may not be sufficient to determine if the excess sectors of non-FAT partition types are unchanged by a tool. If creating a partition modifies any of the excess sectors of the partition then additional steps are required to determine if the tool has changed any of the excess sectors of the destination partition. The following procedure can be used to determine if a tool has modified any of the excess sectors:

1. Initialize the destination disk with the **diskwipe** program.
2. Create the destination partition.
3. Compute a SHA-1 hash over the excess sectors with the **sechash** program.
4. Run the tool being tested.
5. Compute a second SHA-1 hash over the excess sectors.
6. Compare the source partition to the destination partition with the **partcmp** program.

If the tool has changed any of the excess sectors, then the two SHA-1 values will differ. If the two SHA-1 values are the same, the tool has not changed any of the excess sectors.

Test cases 10 and 38 involve NTFS partitions; test case 23 involves a Linux (EXT2) partition. The second procedure was applied to test cases 10, 23 and 38. In all three cases the hashes of the excess sectors were the same before and after running **dd**. The results are presented in Table 5.2-1. The column labeled *Case* is the test case. The *Step* column indicates if the hash value is computed before (**PRE**) or after (**POST**) running the tool. The *Total* column indicates the total number of sectors hashed. The *From LBA* and *To LBA* columns present the starting and ending LBA addresses of the excess sectors. The value of the SHA-1 is presented in the *SHA-1* column. The log files for **sechash** are named **PRELOG.TXT** and **POSTLOG.TXT**.

Table 5.2-1 SHA-1 Values of Excess Partition Sectors Before and After Running dd

Case	Step	Total	From LBA	To LBA	SHA-1
lx-10	PRE	96390	1237005	1333394	91901BA575B3D07CC7A3184A604F62C1B24DAEA4
lx-10	POST	96390	1237005	1333394	91901BA575B3D07CC7A3184A604F62C1B24DAEA4
lx-23	PRE	208782	6152958	6361739	8D15AD9FDB593704BDEDE0BD28E76DC57B37EEF8
lx-23	POST	208782	6152958	6361739	8D15AD9FDB593704BDEDE0BD28E76DC57B37EEF8
lx-38	PRE	96390	1237005	1333394	96B50159080B8C6A2505BC5FC528892A561D4709
lx-38	POST	96390	1237005	1333394	96B50159080B8C6A2505BC5FC528892A561D4709

5.3 Guide for examination of Log Files

After a test case is finished the results are contained in a set of log files that are located in a directory named **LX-xx** (**xx** is the test case number). Each of the support programs executed in the setup and measurement steps produces a log file that can be examined. For source drive setup log files, there is a directory named **setup** with a subdirectory for each source disk. Within each subdirectory are log files from the setup of the corresponding source disk. There are log files from the execution of **logsetup**, **diskwipe** and **diskhash**.

The remainder of this section discusses the relevant content of the log files produced by each support program used in **dd** testing.

5.3.1 LOGSETUP: Setup a Source Drive

Administrative details about the setup of a source disk drive are recorded in the log file, **SETUP.TXT**, from **logsetup**. The disk drive label, host computer used, operator, operating system loaded (if any) and date are recorded.

5.3.2 LOGCASE: Start a Test Case

Administrative details about the execution of a test case are recorded in the log file, **CASE.TXT**, from **logcase**. The labels of the disk drives used, the role assigned each disk, the BIOS drive number for each disk, host computer used, operator, and date are recorded.

5.3.3 PARTAB: Document partition tables

The **partab** program documents the partition tables of the source and destination disk drives. The log file for the source disk should show that the drive has one of the three initial setups from Section 3.1. The log file for the destination drive should show that for an operation on an entire disk drive there is no partition table, but that for an operation on a partition there is a partition of the type required by the test case on the destination drive.

5.3.4 DISKCMP and PARTCMP: Check Accuracy of Duplicate

The comparison programs, **diskcmp** and **partcmp**, have two functions: measure the accuracy of the duplication of the source to the destination and for destinations larger than the source, and determine if **dd** has changed any of the excess sectors.
To measure the accuracy of the duplication, two values from the log file are relevant. The value labeled **Sectors compared** indicates the number of sectors checked and the value labeled **Sectors differ** indicates the number of sectors that are not as expected. If there is a small number of sectors that do not match, the LBA addresses of the non-matching sectors is reported under **Diffs range**. The non-matching sectors can be examined in detail with the **seccmp** program. For **diskcmp** the LBA addresses are relative to the beginning of the disk, for **partcmp** the LBA addresses are relative to the beginning of the partition.

To determine if **dd** has changed the content of the excess sectors the comparison programs categorize the excess sectors of the destination. The evaluation of the categorization of the excess destination sectors is simple in the case of a FAT partition, but has a complication for NTFS and Linux EXT2 partitions. In the case of a FAT partition, all the excess sectors should be categorized as *destination fill*. The number of destination sectors is the value labeled **fewer sectors**. This value should match the value labeled **Dst Byte fill**. In the case of NTFS or Linux EXT2 partitions, some of the excess sectors have content other than *destination fill* and a different procedure (see Section 5.2) is required to evaluate if **dd** has changed the content of the excess destination sectors. The procedure is to identify the excess sectors before executing **dd**. Compute a SHA-1 hash of the excess sectors with the **sechash** program before and after executing **dd**. If the hash values match, then there has been no change to the excess sectors by **dd**.

5.3.5 SECCMP: Investigate Anomaly

The **seccmp** program is not part of the usual test procedures. However, it is used in some test cases (06 and 35) where either **diskcmp** or **partcmp** indicated that the last sector of a source or destination did not match the corresponding sector from the copy operation. The **seccmp** program showed that the last sector of the copy on the destination retained the original value written by **diskwipe** from the destination setup. This implies that **dd** did not copy the last source sector to the destination.

5.3.6 DISKHASH: Verify no Change to Source

The **diskhash** program is used to verify that a source disk has not been changed by **dd**. The **diskhash** log files contains a SHA-1 hash value. The verification is accomplished by comparing the hash value from the test case log file, **HASHALOG.TXT**, to the hash value from the source disk setup, **HASHBLOG.TXT**. If the value agree, then **dd** has not changed the source disk.

5.3.7 SECHASH: Verify no Change to Portion of Destination

The **sechash** program is used for three test cases (06, 23 and 38) to verify that **dd** did not make any changes to the excess sectors of a destination partition. The logfile **PRELOG.TXT** contains the SHA-1 hash of the excess sectors before executing **dd** and the log file **POSTLOG.TXT** contains the SHA-1 hash of the excess sectors after executing **dd**.

5.4 Results Evaluation Procedure

After a test case has been run, the results must be examined to determine if the results should be accepted or if some further actions are required to complete the test case. The evaluation of results must consider if the apparent results are an accurate reflection of the tool under test. Either a successful or unsuccessful test outcome must be reviewed to ensure that an error has not occurred.

The first issue is if a test appears to be successful should we accept the result that the tool has produced the expected result for the particular test case. There are several ways that the test could appear to produce expected results without actually doing so. This would usually involve entire steps not running and the measurement of disks that are left in the final state of an earlier successful test. This can be mitigated by always ensuring that the destination disk has been wiped at the beginning of each test. The **diskwipe** log file should show that the correct number of sectors were wiped for the given destination disk.

The second issue is if a test produces an anomaly and appears to fail, has **dd** failed or is something else wrong. Each anomalous test run is reviewed to characterize the anomaly and then a course of action is selected.

1. If a hardware or procedural problem can be found, e.g., disk drive has failed, or improper configuration for the test, then rerun the test with appropriate adjustments.
2. If no hardware or procedural problem can be found and the anomaly matches a known anomaly then accept the anomaly as genuine.
3. If the anomaly is unique then defer a decision until more test cases have been run. These test cases are referred to as *defer until more*.

4. If the anomaly matches an anomaly in the *defer until more* category then both results are examined for common factors. Based on the reviewer’s judgment a new *known anomaly* may be established, otherwise the test cases remain *defer until more*.

After all test cases have been run any test cases remaining in the *defer until more* category must be resolved by either accepting the anomaly as genuine or reclassified based on additional investigation as needed.

6 Technical Difficulties

Two problems were encountered during the tests. The first problem was that under some conditions the creation of a partition on one hard drive was accompanied by a change to another hard drive. The second problem was unreliable performance of Jaz disks. Neither problem had any effect on the test results.

The original procedure for creating a partition on a destination drive was to install both the source drive and the destination drive in a host computer, boot into DOS, **diskwipe** the destination, and create the partition. This procedure worked without any problems for creating FAT16 and Linux EXT2 partitions. However, when creating FAT32 or NTFS partitions a change would be made to the source drive for the test. The procedure was modified to not install the source drive until after the partition was created on the destination drive.

The Jaz disk would sometimes become unreadable. A Jaz disk would work fine for several test cases but would then become unreadable. An attempt to read from the disk would produce a message that the Jaz disk was *not formatted*. The Jaz disk was then reformatted and reloaded. After discussions with the vendor, the Jaz drives on **cadfael** and **rumpole** were replaced. This improved the problem significantly but did not eliminate it completely.

7 Adapting to a Different Test Environment

The tests were conducted in the CFTT lab at NIST. An attempt to reproduce the test results in another lab may require significant adjustments to the test scripts and procedures. Also the test cases can be used to investigate other issues. This section gives guidelines for adapting the support components of the test cases for other lab environments. Hardware, source hard drives, execution environments, and test scripts are discussed in turn.

7.1 Hardware

The available hardware determines the strategy for organizing the test process. At a minimum, six disk drives are required. Three should be IDE drives such that two of the drives are different sizes and the third drive is the same size as one of the other two drives. The other three disk drives should be SCSI drives with the same size relationship. In addition, one of the SCSI drives should be larger than at least one of the IDE drives and one of the IDE drives should be larger than at least one of the SCSI drives. The drives could be mounted (not removable) in one computer or as in the CFTT lab at NIST, each hard drive can be removed from one computer and placed in another.

Independent replication is accomplished by an agency or lab other than NIST repeating each test case in their own lab environment. Since it is unlikely that the exact hardware used by NIST is present, adjustments and substitutions must be made to run the test cases. For example, the NIST environment used Iomega Jaz drives to contain the support software accessed as drive X. Another lab that does not have Jaz drives available might put the support software on floppy disks, LS-110 drives (SuperDisk) or CD-ROM. Even more significant are the actual hard drives used in the tests. It is not always clear what would be an equivalent substitution for a hard drive used.

There is an important issue about replication here. The one anomaly (omission of the last sector from a source with an odd number of sectors) found did not depend on any of the actual test parameters. However, replication of the anomaly depends on selecting a drive with certain characteristics. There is no easy answer to this issue. The *ad hoc* answer is to require any substitution to conform to any conditions determined after the test is run that are required to replicate the result. Information learned during the test process should be applied to any attempted replication. In this case, we have learned that the oddness or evenness of the source is a hidden test parameter that has been discovered during testing. Any attempt to replicate the tests must use this hidden parameter. This implies a rule for disk or partition substitution. If the exact disk drive or partition is not available then the substitution must be of the same interface type (e.g., IDE or SCSI) and must have an odd number of sectors if and only if the object used in the test conducted by NIST had an odd number of sectors.

7.2 Source Hard Drives

The more hard drives available the easier it is to organize and setup source drives in advance. If there are only a few drives available then source drives may need to be repeatedly setup as the drive is moved into different roles. This can take a significant amount of time. This can be mitigated by a careful selection of test case order such that once a source drive is setup all the test cases that require that drive are run before assigning the drive to another role. This is facilitated by examination of test case parameters from Table 1.2-1 to determine the type of hard drive interface required for a given test.

7.3 Execution Environments

There are two execution environments used for the test cases. The support software that does test setup and results measurement runs in a DOS environment. The **dd** program is executed in a Linux environment.

The DOS environment is established from a DOS boot disk. The boot disk should be similar to the one described in section 3.2, except for changes to reflect the actual hardware present. For example, if no Jaz drive is present all files in the **guest** directory and references to **guest** in the **autoexec.bat** file can be deleted.

The Linux environment has more possibilities for change. A list of alternatives to consider follows:

- If the Linux kernel supports large (greater than 2GB) files then **dk_backup** and **dk-restore** can be revised to use a single image file rather than the current version that splits the image file into several small files. However, one advantage of breaking the image file into smaller chunks is the ability to monitor the test progress.
- The current Linux environment is booted from a hard drive. The environment could be obtained from a bootable CD or other media.
- These tests are written for the Linux environment but could be adapted for other UNIX-like operating systems such as FreeBSD with small changes. For example, in FreeBSD the disk drive device names would need to be changed.

7.4 Test Scripts

There are several revisions to the **pre** and **post** scripts that could improve the testing process.

- Make the location of programs and scripts a variable that can be either obtained from a configuration file or set when the computer is booted. This simplifies relocating files or specifying an alternative script.
- Copy any scripts used by a test case to the log directory. This simplifies identification of the scripts actually used in a test case if there have been revisions to the test cases.
- Any modification to the **pre** and **post** scripts needs to ensure that the drive assignments remain correct.
- An alternative to having a unique **pre** or **post** script for each test case would be to have a general script that takes additional parameters to specify drive number assignments, Linux device name and **partition magic** scripts.
- The **pre** script should be redesigned. Rather than modifying **autoexec.bat** to setup for print the partition tables, split the script into two parts. The first part should setup the destination drive without having the source drive installed. The system can then be shutdown, the source drive installed and the partition tables for both drives printed. Having both the source and destination drives installed while creating a partition on the destination caused a problem that is discussed in section 6.
- Setup source drives with a script.