May 6, 2004

Test Environment and Procedures for Testing dd provided with FreeBSD 4.4

Version 1.0

National Institute of Standards and Technology
Technology Administration, U.S. Department of Commerce

Abstract[†]

This document describes the testing of dd in the FreeBSD environment. The test cases that were applied are described in *Disk Imaging Tool Specification, Version 3.1.6*.

The tests were run on test systems in the Computer Forensics Tool Testing Lab at the National Institute of Standards and Technology. A variety of hard drives 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 MS-DOS operating system, computer operation, computer hardware components such as hard drives, hard drive interfaces (e.g., IDE or SCSI) and computer forensics.

Partition Magic® is a registered trademark of Power Quest Corporation, Inc.
Red Hat® is a registered trademark of Red Hat, Inc.
Linux™ is a trade mark of Linus Torvalds.
Turbo Assembler® is a registered trademark of Borland International, Inc.
Easy CD Creator® **5** is a registered trademark of Roxio, Inc.
Borland® is a registered trademark of Borland International, Inc.
MS-DOS® is a registered trademark of Microsoft Corporation, Inc.
Microsoft Windows® is a registered trademark of Microsoft Corporation, Inc.
Pentium® is a registered trademark of Intel, Inc.
All other products mentioned herein may be trademarks of their respective companies.

[†] **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

List of Figures

1 Introduction

The objective of the Computer Forensics Tool Testing (CFTT) 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 tool 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 (NIST), 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 in the FreeBSD Version 4.4 environment. 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. An independent replication of the testing would require sufficient hardware and software resources to execute the test cases, this document, the **dd** test report, *Disk Imaging Tool Specification, Version 3.1.6*, the FS-TST 1.0 software plus revised error simulation software. Since it is unlikely that the exact hardware used by NIST is present, adjustments and substitutions must be made to run the test cases.

The intended audience for this document should be familiar with FreeBSD, the MS-DOS 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

Several items were assembled and prepared before the testing began including: 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*, revised error simulation software, scripts and the documentation for the software is available from the web site: http://www.cftt.nist.gov.

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

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

For each test case, the commands that need to be executed were contained in a set of script files. 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 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, Windows media drive are described in Section 3. The script files for each step are described in Section 4. Section 5 describes the execution procedures.

2 Test Environment Support Software

The tests were run at the CFTT lab at NIST.

Support software, FS-TST Release 1.0, was developed to support the testing of disk imaging tools. FS-TST Release 1.0 can be obtained from the http://www.cftt.nist.gov web site. The support software serves three main functions: initialization of a disk to a known value [DISKWIPE], comparison of a source with a destination [DISKCMP, PARTCMP, and SECCMP], and detection of changes to a disk [DISKHASH and SECHASH]. All programs were written in ANSI C and compiled with the Borland C++ compiler version 4.5. BADDISK and BADX13 were written in assembler language and compiled with Borland Turbo Assembler version 5.0. The **csh** scripts were executed using the version of csh provided with FreeBSD 4.4.

3 Media Setup

The test cases required several media components to be created before the test cases could be executed. The following items were created.

1. Source hard disk drives for the test cases.

2. A media hard drive with FreeBSD, scripts to execute dd and space to contain image files.
3. A Windows 98 DOS boot floppy that contains control scripts, log files and creates the run-time environment for the test case setup and measurement.
4. A CD that contains support software, and utility software.

3.1 Source Disks

There were too many possible disk layouts for all to be used in the tests. Five configurations were selected that covered the most common partition types. The first configuration was a dual boot Red Hat Linux 7.1 and Windows Me. This configuration also included FAT16, Linux EXT2, hidden partitions and deleted partitions (Table 3-1). The second configuration was a Windows 2000 system with both FAT32 and NTFS file systems (Table 3-2). The third configuration did not contain a valid partition table. The fourth configuration was a single FAT12 partition and the fifth configuration was a single FAT32 partition. All partitions were created with **Partition Magic Pro 6.0**.

Table 3-1 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 3-2 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 was as follows:

1. Selected type of setup: Windows Me/Linux, Windows 2000 or none.
2. Selected a hard drive
3. Selected computer, installed drive, booted into PC DOS 6.3 from a boot floppy.
4. Ran **LOGSETUP** to make a record of the setup.
5. Ran **DISKWIPE** to initialize the drive contents.
6. If the setup type uses an operating system the following two steps were completed
7. Ran **partition magic** to partition the drive. For Windows Me/Linux source disk the script in Table 3-3 was used and for Windows 2000 source disk the script in Table 3-4 was used.
8. The installation instructions for each operating system were followed. For a Windows Me/Linux configuration, Windows Me was installed first then Linux. For a Windows 2000 configuration, Windows 2000 was installed.
9. Deleted files were created using a script (DOS batch file) that created a directory (X:\UDT, where X is a drive letter) with deleted files and a deleted subdirectory (Table 3-5).
10. Ran **DISKHASH** to create a reference SHA-1 hash for the source disk.

Table 3-3 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 3-4 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 3-5 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 Media Drive

Several media drives were setup to contain the runtime environment for dd.

1. A media drive was selected.
2. The FreeBSD operating system was loaded to the drive.
3. The **/x** directory was created to contain scripts, image files and log files.
4. Test scripts were installed.

3.3 Windows 98 Boot Floppy

The Windows 98 boot floppy disk provided an execution environment for the support software and **dd** execution. The commands used to setup the floppy are presented in Table 3-8. The **io.sys** file on the boot floppy was scrubbed of any references to the C: drive and the following programs: **DBLSPACE.BIN, DRVSPACE.BIN**, and **STACKER**.

In addition to the setup documented in Table 3-8 the files EN.EXE, BADX13.COM Version 3.2 and BADDISK.COM Version 3.2 are copied to the boot floppy.

Table 3-6 Windows 98 Boot Floppy AUTOEXEC.BAT

```
@ECHO OFF
A:\drivers\mscdex.exe /D:mscd001 /L:Z
a:\guest\guest letter=x
```

The **CONFIG.SYS** file loads drivers for ASPI SCSI devices, CDROM drive and sets the last drive letter.

Table 3-7 Windows 98 Boot Floppy CONFIG.SYS

```
ACCDATE=C- D- E- F- G- H- I- J- K- L- M- N- O- P- Q-
device=A:\drivers\himem.sys /testmem:off
device=A:\drivers\oakcdrom.sys /D:mscd001
device=A:\drivers\aspi8u2.sys /D /PD800 /Q9
device=A:\drivers\aspi8dos.sys
device=a:\drivers\aspicd.sys /D:mscd001
files=30
buffers=10
dos=high,umb
stacks=9,256
lastdrive=z
```

Table 3-8 Windows 98 Boot Floppy Setup

From a Windows 98 System, insert a blank floppy disk and open a DOS command window

```
FORMAT A: /S
```

```
MKDIR A:\GUEST
MKDIR A:\DRIVERS
COPY HIMEM.SYS A:\DRIVERS
COPY MSCDEX.EXE A:\DRIVERS
COPY MOUSE.COM A:\DRIVERS
COPY MOUSE.INI A:\DRIVERS
COPY SMARTDRV.EXE A:\DRIVERS
COPY GUEST.EXE A:\GUEST
COPY GUEST.INI A:\GUEST
COPY GUESTHLP.TXT A:\GUEST
COPY ASPI2DOS.SYS A:\DRIVERS
COPY ASPI4DOS.SYS A:\DRIVERS
COPY ASPI8DOS.SYS A:\DRIVERS
COPY ASPI8U2.SYS A:\DRIVERS
COPY ASPICD.SYS A:\DRIVERS
COPY BTCDROM.SYS A:\DRIVERS
COPY BTDOSM.SYS A:\DRIVERS
COPY FLASHPT.SYS A:\DRIVERS
COPY OAKCDROM.SYS A:\DRIVERS
setup AUTOEXEC.BAT and CONFIG.SYS
```

The **Scrub Log** is a list of changes made to the **IO.SYS** file to ensure no references to the C: drive during boot.

Figure 3-1 Windows 98 Boot Floppy Scrub Log

```
A:\SCRUB.EXE compiled at 23:47:10 on Nov  7 2001
C:\ found at 17268, replaced by A:\
C:\ found at 17664, replaced by A:\
C:\ found at 20484, replaced by A:\
C:\ found at 20499, replaced by A:\
C:\ found at 42632, replaced by A:\
C:\ found at 42667, replaced by A:\
DBLSPACE.BIN found at 42720, replaced by NOOSPACE.BIN
DRVSPACE.BIN found at 42734, replaced by NOOSPACE.BIN
C:\ found at 60980, replaced by A:\
C:\ found at 60996, replaced by A:\
C:\ found at 66795, replaced by A:\
C:\ found at 66805, replaced by A:\
C:\ found at 69755, replaced by A:\
DBLSPACE.BIN found at 60999, replaced by NOOSPACE.BIN
DRVSPACE.BIN found at 60967, replaced by NOOSPACE.BIN
STACKER found at 61013, replaced by SLACKER
Starting Windows 98 found at 68272, replaced by Starting NIST Fboot
C:\ found at 220112, replaced by A:\
222390 bytes read from io.sys
```

3.4 CD-ROM

The CD-ROM was created using **Easy CD Creator 5** and was divided into the following directories used in testing:

- PM: Contained a copy of **Partition Magic**.
- SS: Contained a copy of FS-TST support programs.

In addition, the CD contained the following additional directories not used in testing:

- NU: Contained some commercial utility programs.
- Tools: Contained other DOS utility programs.

4 Test Execution Scripts

Each test case was executed in three parts: setup, tool execution and results measurement. The setup and measurement steps were executed as a series of DOS batch files (i.e., scripts) that guided the test operator through each phase of the test. The tool execution part was executed by **csh** scripts in the Linux environment.

4.1 DOS Batch Files

This section describes the DOS batch files used to setup a test case and measure the results.

The START-IT.BAT script was used to start a test case. The script called the LOGCASE program to record information about the test case. The script also created simple scripts with parameters filled in so that the operator did not need to reenter information to execute the other required scripts. The simple scripts created by START-IT.BAT were DST.BAT (to execute XDST.BAT), CMP.BAT (to execute XCMP.BAT), HASH.BAT (to execute XHASH.BAT) and NOHASH.BAT (to execute XNOHASH.BAT). The simple scripts took a single parameter, *hostname*, to indicate the test computer used to execute the script.

Figure 4-1 Script to Start a Test Case (START-IT.BAT)

```
@ECHO OFF
REM Case Host Operator Src Dst Boot/Media reverse/partition
REM reverse/partition = r ==> cmp with src on 81 (rather than 80)
REM else reverse/partition indicates partition script file
REM Script Version 1.0
cd A:\
del a:\*.txt
REM
set ThisCase=%1
set Host=%2
set Op=%3
set Src=%4
set Dst=%5
set Media=%6
set Sz=%7
Z:\ss\logcase %ThisCase% %Host% %Op% 80:%Src% 81:%Dst% 80:%Media%
ver >> A:\CASE.TXT
echo Script Version 1.0 >> A:\CASE.TXT
echo A:\s\xdst %ThisCase% %%1 %Op% %Dst% %Sz% > A:\DST.BAT
echo A:\s\xdst-pt %ThisCase% %%1 %Op% %Dst% > A:\PT.BAT
echo A:\s\xcmp %ThisCase% %%1 %Op% %Src% %Dst% %Sz% > A:\CMP.BAT
echo A:\s\xpcmp %ThisCase% %%1 %Op% %Src% %Dst% %%2 %%3> A:\ACMP.BAT
echo A:\s\xhash %ThisCase% %%1 %Op% %Src% > A:\HASH.BAT
echo A:\s\xnohash %ThisCase% %%1 %Src% > A:\NOHASH.BAT
echo
echo Run case %ThisCase% on %Host% by %Op%
echo Source %Src% Destination %Dst% Media %Media%
echo On partition %Sz%
```

The script XDST.BAT was executed by DST.BAT to setup a destination drive. If a partition was required, **partition magic** was executed to create the partition.

Figure 4-2 Script to Setup a Destination Drive (XDST.BAT)

```
REM Script Version 1.0
REM Test-case Host Operator Destination Partition-Script
@ECHO OFF
Z:\ss\diskwipe %1 %2 81 %4 /noask /dst /new_log /comment "%3"
if "%5"=="" goto FinishUp
if "%5"=="r" goto FinishUp
REM %ramd%:
REM pqmagic /CMD=A:\PM\%5
A:\S\XMAGIC %5
:FinishUp
A:
```

The XMAGIC.BAT script was executed from the XDST.BAT script to create a partition with **partition magic** and logged the partition creation.

Figure 4-3 Script to Create a Partition (XMAGIC.BAT)

```
REM run and log partition magic
echo %ramd%:\pqmagic /CMD=A:\PM\%1 > A:\MAGICLOG.TXT
%ramd%:
pqmagic /CMD=A:\PM\%1
```

The XDST-PT.BAT script was executed by PT.BAT.

Figure 4-4 Script to Print a Partition Table (XDST-PT.BAT)

```
REM Script Version 1.0
REM Test-case Host Operator Destination
@ECHO OFF
Z:\ss\partab %1 %2 81 /all /new_log /comment %3(%4)
ren A:\PT81LOG.TXT PT_DSTB.TXT
```

The XCMP.BAT script was executed by CMP.BAT. The script compares two drives.

Figure 4-5 Script to Compare Source to Destination (XCMP.BAT)

```
REM Script Version 1.0
REM Test-case Host Operator Source Destination Reverse/Partition
@ECHO OFF
if "%6"=="r" goto Src81
Z:\ss\partab %1 %2 81 /all /new_log /comment %3(%4)
ren A:\PT81LOG.TXT PT_DSTA.TXT
Z:\ss\partab %1 %2 80 /all /new_log /comment %3(%4)
ren A:\PT80LOG.TXT PT_SRC.TXT
if "%6"=="" goto Src80
Z:\ss\partcmp %1 %2 80 %4 81 %5 /new_log /comment "%3" /select 1 1
goto FinishUp
:Src80
Z:\ss\diskcmp %1 %2 80 %4 81 %5 /new_log /comment "%3"
goto FinishUp
:Src81
Z:\ss\partab %1 %2 80 /all /new_log /comment %3(%4)
ren A:\PT80LOG.TXT PT_DSTA.TXT
Z:\ss\partab %1 %2 81 /all /new_log /comment %3(%4)
ren A:\PT81LOG.TXT PT_SRC.TXT
Z:\ss\diskcmp %1 %2 81 %4 80 %5 /new_log /comment "%3"
:FinishUp
```

```
echo ready to hash for case %1
```

The XPCMP.BAT script was executed by ACMP.BAT. The script compares two partitions.

Figure 4-6 Script to Compare two Partitions (XPCMP.BAT)

```
REM Script Version 1.0
REM Test-case Host Operator Source Destination Src_pt Dst_pt
@ECHO OFF
Z:\ss\partab %1 %2 81 /all /new_log /comment %3(%4)
ren A:\PT81LOG.TXT PT_DSTA.TXT
Z:\ss\partab %1 %2 80 /all /new_log /comment %3(%4)
ren A:\PT80LOG.TXT PT_SRC.TXT
Z:\ss\partcmp %1 %2 80 %4 81 %5 /new_log /comment "%3" /select %6 %7
echo ready to hash for case %1
```

The XHASH.BAT script was executed by HASH.BAT. The script hashed the source drive and created a directory for the test case log files and the log files were copied to the test case directory.

Figure 4-7 Script to Rehash Source and Save Log Files (XHASH.BAT)

```
REM Script Version 1.0
@ECHO OFF
Z:\ss\diskhash %1 %2 80 /comment %4(%3) /new_log /after
mkdir a:\%1
copy A:\*.TXT a:\%1
echo Case: %1 finished
```

4.2 csh Scripts

This section describes the **csh** scripts that were used to execute the test cases. After FreeBSD was booted, the floppy drive was mounted to save the log files. This was accomplished by executing **/x/mount-fd.** After the floppy was mounted, then either **dd_copy** or **dd_image** was executed as required. The script **dd_copy** was used for cases that made a direct drive to drive copy; **dd_image** was used for cases that created an image file.

Figure 4-8 Script to Mount Floppy (mount-fd)

```
#
mount -t msdos /dev/fd0 /x/fd
dmesg | egrep '(MB)|(Copyright.*BSD)' | grep -v cd0 | tail -3
```

The **dd_copy** script copied a source drive to a destination. The script used a block size of one sector (512 bytes). The script also loged any messages from **dd** and copied the logs to the floppy drive.

Figure 4-9 Script to Run to Copy a Drive (dd_copy)

```
#!/bin/csh -f
# Script to copy disk to disk
# dd_copy source_device destination_device Test_case
# log results to /x/logs/test_case/copy_log.txt
#
set src = $1
set dst = $2
set log_dir = /x/logs/$3
mkdir $log_dir
set log_file = $log_dir/C$3.TXT
echo -n "Start ${3}: " > $log_dir/C${3}.TXT
date >> $log_dir/C${3}.TXT
uname -a >> $log_file

set cmd = "dd if=/dev/$src of=/dev/$dst bs=512"
echo "Command: $cmd" >> $log_dir/C${3}.TXT
$cmd >>& $log_dir/C${3}.TXT

echo -n "Finish ${3}: " >> $log_dir/C${3}.TXT
date >> $log_dir/C${3}.TXT
cp $log_dir/C${3}.TXT /x/fd
```

The script **dd_image** created an image of a source on the media drive and then restored the image to a destination drive. The image was created by the script **dk-backup** and the restore was done by the script **dk-restore**. After the restore was finished the script copied any log files to the floppy drive.

Figure 4-10 Script to Image a Drive (dd_image)

```
#!/bin/csh -f
# dd_image src dst case
##  dd_image $src $dst case
#
set src = $1
set dst = $2
set log_dir = /x/logs/$3
mkdir $log_dir
set log_file = $log_dir/A$3.TXT
echo -n "Start ${3}: " > $log_file
date >> $log_file
uname -a >> $log_file
echo "dd_image $argv" >>$log_file

echo "Running backup, Log to $log_file"
./dk-backup /dev/$src $3
echo "Running restore, Log to $log_file"
./dk-restore /dev/$dst $3
echo "Backup/restore finished"

echo -n "Finish ${3}: " >> $log_file
date >> $log_file
cp $log_file $log_dir/[ABR]$3.TXT /x/fd
```

Figure 4-11 Script to Acquire (dk-backup)

```
#!/bin/csh -f
if ($#argv != 2) then
echo "Usage: $0 src_dev test_case
exit (1)
endif
```

```
#set echo
set src = $1
set dst_dir = /x/img/$2
set log_file = /x/logs/$2/B$2.TXT
echo "dk-backup $argv" >$log_file
date >> $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.$pass
set cmd = "(dd if=$src skip=$skip count=$n bs=512 | gzip > $dst.gz) >& $dst_dir/this_pass"
echo "$cmd" >> $log_file
echo "Cmd: $cmd"
(dd if=$src skip=$skip count=$n bs=512 | gzip > $dst.gz) >& $dst_dir/this_pass
echo "Pass: $pass Status: $status" >> $log_file
cat $dst_dir/this_pass >>$log_file
@ skip = $skip + $n
grep '^0+0' $dst_dir/this_pass >/dev/null
set more = $status
end
echo -n "Finish at " >> $log_file
date >> $log_file
```

Figure 4-12 Script to Restore (dk-restore)

```
#!/bin/csh -f
if ($#argv != 2) then
echo "Usage: $0 dst_dev test_case
exit (1)
endif
#set echo
echo "dk-restore $argv"
set dst = $1
set dst_dir = /x/img/$2
set log_file = /x/logs/$2/R$2.TXT
echo "dk-restore $argv" > $log_file
date >> $log_file
@ total = 1
@ max = 1000000
@ n = $max
@ skip = 0
@ pass = 101
set more = 1
while ($more )
echo "Restore pass $pass"
set src = $dst_dir/${2}-image.$pass.gz
set cmd = "(gunzip <$src | dd of=$dst seek=$skip count=$n bs=512 ) >>& $dst_dir/this_pass"
echo "$cmd" >> $log_file
echo "Cmd: $cmd"
(gunzip <$src | dd of=$dst seek=$skip count=$n bs=512 ) >& $dst_dir/this_pass
echo "Pass: $pass Status: $status" >> $log_file
cat $dst_dir/this_pass >> $log_file
@ skip = $skip + $n
@ pass = $pass + 1
if (-e $dst_dir/${2}-image.$pass.gz ) then
```

```
set more = 1
else
set more = 0
endif
end
echo -n "Finish at " >> $log_file
date >> $log_file
```

5 Test Case Execution Procedures

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

5.1 Computer Checkout Procedure

Before any test cases were run, each test computer was checked for suitability as a test platform and the FS-TST Version 1.0 CD-ROM was installed. The procedure was as follows:

1. A host computer was selected for check out.
2. If the computer was on, the computer was shutdown and turned off
3. Any hard drives were removed.
4. The DOS boot floppy was inserted into the computer.
5. The computer was turned on and booted to DOS.
6. It was verified that the computer BIOS boot order specified boot to floppy drive before hard drive.
7. If the boot or Power on Self Test (POST) failed, an attempt was made to diagnose the problem. If the problem could not be diagnosed and repaired, then the computer was not suitable for testing and was not used for testing. A computer that was not suitable for testing had a sticky label with the words *DO NOT USE* placed over the power switch to indicate that the computer should not be used.
8. If the computer was available for testing, the FS-TST Version 1.0 CD-ROM was inserted.
9. The computer was shutdown.

5.2 Execution Procedure

This section describes the generic test procedures used to test dd; slight deviations were required for some test cases.

1. A test case was selected to run. Test case selection is documented in Section 3 of *Test Results for Disk Imaging Tools: dd in the FreeBSD environment,* available at *http://www.cftt.nist.gov*.
2. The required removable media was collected.
3. The test computer and hard drives were selected.
4. The destination drive was installed.
5. The test computer was booted into DOS.
6. The script START-IT.BAT was executed.
7. The script DST.BAT was executed.
8. The script PT.BAT was executed.
9. If required, the excess sectors of the destination were hashed with SECHASH.EXE.
10. The test computer was shut down.
11. The destination drive was removed.
12. The source, destination and media drives were installed in the test computer.
13. The test computer was booted to FreeBSD on the media drive.
14. The operator logged in as **root**.
15. The command **/x/mount-fd** was executed.
16. If the test case was a *copy case* the script **/x/dd_copy** was executed otherwise the script **dd_image** was executed to image the source to the media drive.
17. The test computer was shut down.
18. The media drive was removed.
19. The test computer was booted to DOS.
20. The CMP.BAT script was run for *physical drive cases*; the ACMP.BAT script was run for *logical drive cases*.
21. If required, the excess sectors of the destination were hashed with SECHASH.EXE.
22. The script HASH.BAT was run.
23. The test computer was shutdown.
24. The log files created on the floppy disk were copied to a permanent read only location subject to regular backup.

5.3 Results Evaluation Procedure

After a test case was run, the results were examined to determine if the results should be accepted or if some further actions were required to complete the test case. The evaluation of results determines if the apparent results are an accurate reflection of the tool under test. Either a successful or unsuccessful test outcome was reviewed to ensure that an error in the testing process had not occurred.

The first issue was if a test appeared 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 was mitigated by always ensuring that the destination disk was wiped at the beginning of each test. The **diskwipe** log file was reviewed to verify that the correct number of sectors were wiped for the given destination disk.

The second issue was if a test appeared to be successful, should the result that the tool has produced the expected result for the particular test case be accepted? Each anomalous test run was reviewed to characterize the anomaly and then a course of action was selected.

1. If a hardware or procedural problem was found, e.g., disk drive has failed, or improper configuration for the test, then the test was rerun with appropriate adjustments.
2. If no hardware or procedural problem was found and the anomaly matched a known anomaly then the anomaly was accepted as genuine.
3. If the anomaly was unique then a decision was deferred until more test cases were run. These test cases are referred to as *defer until more*.
4. If the anomaly matched an anomaly in the *defer until more* category then both results were examined for common factors. If sufficient common factors were found, a new *known anomaly* was established, or the test cases remained *defer until more*.

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