Digital Investigation (2004) 1, 114—128

Digital Investigation

www.elsevier.com/locate/diin

PRODUCT REVIEW

Tool review—WinHex

Eoghan Casey[a,b,*]

[a]Knowledge Solutions LLC, 61535 S Hwy 97 #9-148, Bend, OR 97702, United States
[b]Stroz Friedberg LLC, 1150 Connecticut Ave., NW, Suite 200, Washington, DC 20036, United States

Received 5 April 2004; revised 14 April 2004; accepted 14 April 2004

KEYWORDS
Digital forensics tool testing;
Digital evidence preservation;
Forensic examination;
File systems;
Data recovery

Abstract This paper presents strengths and shortcomings of WinHex Specialist Edition (version 11.25 SR-7) in the context of the overall digital forensics process, focusing on its ability to preserve and examine data on storage media. No serious problems were found during non-exhaustive testing of the tool's ability to create a forensic image of a disk, and to verify the integrity of an image. Generally accepted data sets were used to test WinHex's ability to reliably and accurately interpret file date—time stamps, recover deleted files, and search for keywords. The results of these tests are summarized in this paper. Certain advanced examination capabilities were also evaluated, including the creation of custom templates to interpret EXT2/EXT3 file systems. Based on this review, several enhancements are proposed. In addition to these results, this paper demonstrates a systematic approach to evaluating similar forensic tools.
© 2004 Elsevier Ltd. All rights reserved.

Introduction

WinHex began as a disk editing program and has developed into a forensic tool that is useful to digital evidence examiners of all skill levels. Experienced examiners use WinHex Specialist Edition to validate the results of other tools and to perform specialized tasks such as file comparison and text extraction. This program is also used to teach novices about disk layout, file system structures, data recovery, and other fundamental concepts in digital forensics. When a new tool emerges in this field, it is necessary to assess its capabilities and identify any weaknesses.

When evaluating any forensic application, it is important to measure its performance against generally accepted test data sets whenever feasible. For this purpose, the Digital Investigation Tool Test Images created by Brian Carrier were used (publicly available at http://dftt.sourceforge.net/). These data sets are designed specifically to test a forensic tool's ability to reliably and accurately interpret file date—time stamps, recover deleted files, and search for keywords. When it is not feasible to test a particular feature using accepted data sets, it is usually sufficient to validate the results with other forensic tools to ensure they are consistent or that any differences are explicable. This validation approach was used during this tool review to confirm that WinHex was capable of making a forensic image of data on storage media, calculating the MD5 hash value of a disk or image,

* Corresponding author.
E-mail address: eco@corpus-delicti.com.

1742-2876/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.diin.2004.04.001

recovering deleted files, and correctly interpreting various date—time stamps embedded in different file types. Based on this tool review and testing, some future refinements are proposed to help developers prioritize enhancements in the forensic capabilities of WinHex.

In the process of reviewing WinHex, this paper also familiarizes digital investigators with different aspects of file systems as a source of evidence. The primary focus is on FAT and NTFS because these are the file systems that WinHex directly supports. Additionally, with the increasing use of UNIX systems on personal and corporate systems, digital investigators require a solid understanding of UNIX file systems. Many people avoid this area because of the misconception that it is beyond their technical abilities. A secondary aim of this paper is to make UNIX file systems more accessible by demonstrating how WinHex can be used to examine EXT2 and EXT3 file systems which are commonly found on Linux systems. This treatment helps novices who want to improve their understanding of UNIX file systems in a familiar Windows-based environment. This knowledge is also useful to experienced practitioners who examine UNIX systems with Windows-based tools such as EnCase and FTK, enabling them to confirm important findings at a low level using WinHex.

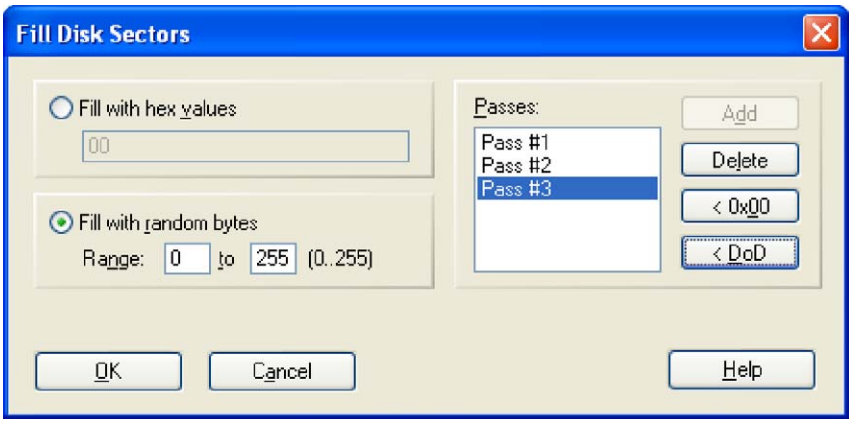
Preservation

Prior to making a forensic image of digital evidence it is a good practice to create a sanitized storage area. Using a storage area that has been wiped of all previous data and verified to be empty prevents the possibility of data from past investigations from commingling with the current case. A selected disk can be sanitized using WinHex's *Fill Disk Sectors* option on the Edit menu. For the purpose of forensic sterilization it is generally sufficient to overwrite data once with a known pattern. WinHex permits this type of wiping as well as the more rigorous approach recommended in USDOD (1995) to prevent extraordinary hardware-based recovery efforts (see Fig. 1). This author verified that WinHex successfully filled all sectors as instructed after warning the user several times that all data on the disk would be overwritten.

Additionally, prior to making a forensic image, it is a good practice to calculate the cryptographic checksum of the original disk. MD5 or SHA1 hash values of a disk are useful for verifying the integrity of digital evidence by confirming that a forensic image is identical to the original. The cryptographic checksum of a selected disk or file can be obtained using WinHex's *Calculate Hash* option on the Tools menu.

The WinHex *Clone Disk* option under the Tools—Disk Tools menu can be used to either clone the source disk onto another disk, or copy all sectors of a disk into a file as shown in Fig. 2. Before using this feature, it is advisable to configure WinHex to automatically calculate the MD5 value of data that it copies by setting the *Calculate auto-hash* to MD5 under the Options—Security menu item. It is also advisable to run WinHex on Windows 2000/XP when performing this operation to ensure that the disk size is obtained using I/O control codes (e.g., IOCTL_DISK_GET_LENGTH_INFO) instead of relying on information provided by the BIOS via the Int13h interface. Notably, because WinHex does not send ATA commands directly to the drive, it may not detect all sectors that are hidden by a Host Protected Area (HPA) or Device Configuration Overlay (DCO).

Testing performed by this author found that WinHex copied all data from storage media, and correctly calculated the MD5 values of storage media and image files. However, error handling during the acquisition process is lacking. For instance, when saving an image file to a disk that

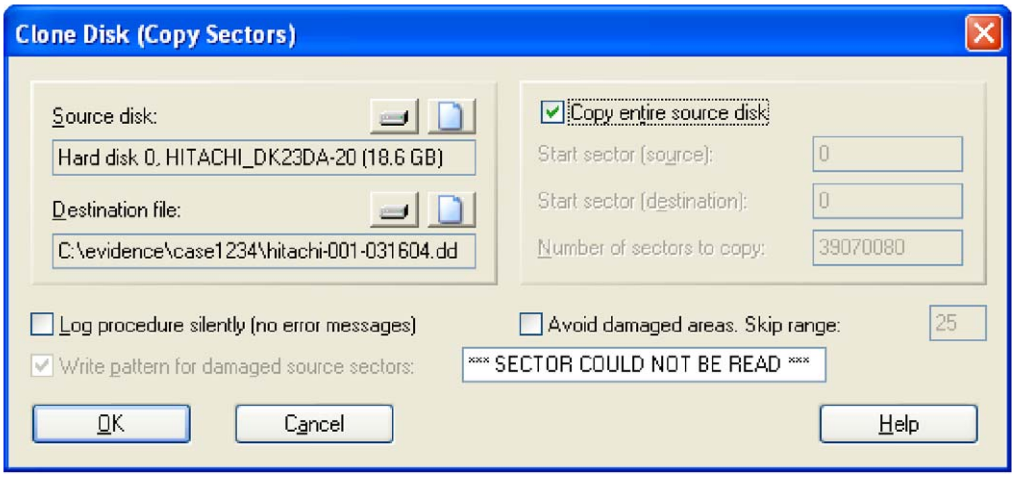

Fill Disk Sectors

( ) Fill with hex values: 00

(●) Fill with random bytes
Range: 0 to 255 (0..255)

Passes:
Pass #1
Pass #2
Pass #3

Figure 1 Sanitizing a disk using the "DoD" option of WinHex's *Fill Disk Sectors* facility.



Source disk: Hard disk 0, HITACHI_DK23DA-20 (18.6 GB)
Destination file: C:\evidence\case1234\hitachi-001-031604.dd
☑ Copy entire source disk
Start sector (source): 0
Start sector (destination): 0
Number of sectors to copy: 39070080
☐ Log procedure silently (no error messages)
☐ Avoid damaged areas. Skip range: 25
☑ Write pattern for damaged source sectors: *** SECTOR COULD NOT BE READ ***

Figure 2 Copying data in all sectors from a hard disk into a file.

was too small, WinHex aborted the acquisition process after it had filled the destination disk, giving an error that the disk was full. It would be more effective to inform the user from the outset that there is insufficient space on the target disk, or to prompt the user for additional storage when the destination disk is full. Similarly, when trying to clone a disk onto another disk that was too small, WinHex generated an unspecified input error and aborted the operation. It would be more helpful for WinHex to inform the user that the capacity of the destination drive was too small. Furthermore, WinHex permitted the unwary user to perform the dangerous operation of saving a forensic image onto the same disk that was being copied, thus overwriting unallocated space on the evidentiary drive. The potential for such mistakes emphasizes the importance of using a write-blocker and designing tools to only save data to a disk that the user has specifically designated as writable.

Examination

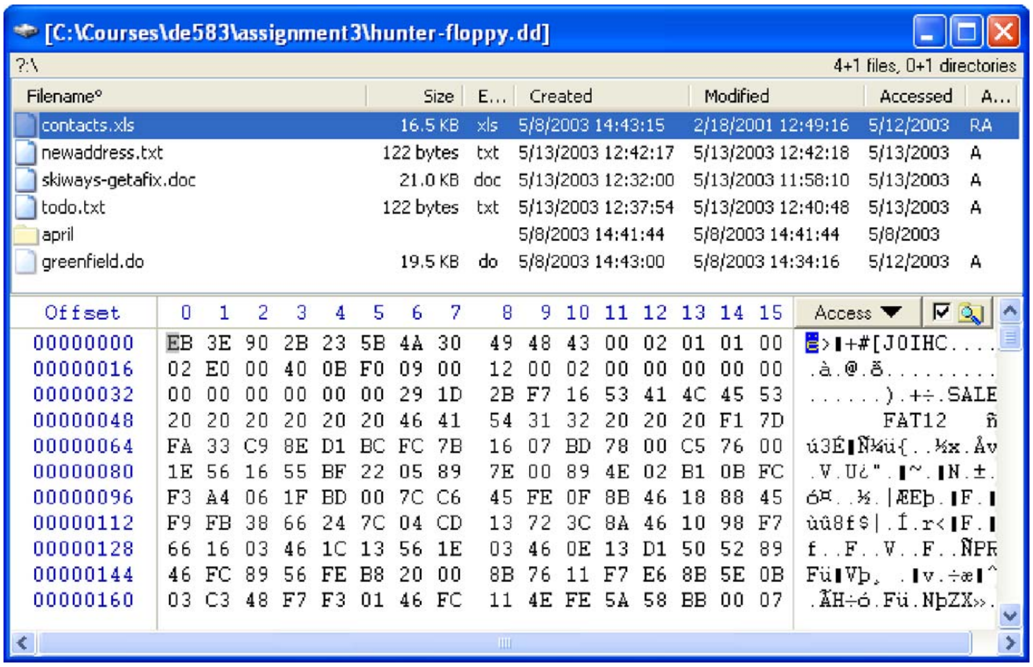
The examination process involves data recovery, harvesting, reduction, organization and searching (Casey and Palmer, 2004). A thorough examination results in all relevant data being organized and presented in a manner that facilitates detailed analysis. Each step in the examination process is dealt with separately here.

To begin, it is common practice to obtain information about the disk using the *Media Details Report* option on WinHex's Specialist menu as shown here:

```
WinHex 11.25
3/16/2004, 14:52:15

[C:\case1234\evidence\gamblor.1]

Total capacity: 170,139,648 bytes = 162 MB

Number of cylinders: 41
Number of heads: 128
Sectors per track: 63
Bytes per sector: 512
Total no. of sectors: 332,304
Surplus sectors at end: 1,680

Partition 1
162 MB, FAT16
Sectors 46 - 331,935
Partition table: Sector 0
Total no. of sectors: 331,890
Usable sectors: 331,528
First data sector: 357
Bytes per sector: 512
Bytes per cluster: 4,096
Free clusters: 12,396 = 30% free
Total clusters: 41,441

Unused inter-partition space:
Sectors 1 - 45 (22.5 KB)
Sectors 331,936 - 332,303 (184 KB)
= 207 KB
```

Using this feature, WinHex successfully detected all six partitions in the Extended DOS Partition Test (Carrier, 2003a).

Information in the *Media Details Report* can be compared with parameters on the disk label, CMOS settings, or with results from other tools to confirm that the actual values were detected by



```
[C:\Courses\de583\assignment3\hunter-floppy.dd]
?:\                                                         4+1 files, 0+1 directories
Filename°            Size       E...  Created             Modified            Accessed    A...
contacts.xls         16.5 KB    xls   5/8/2003 14:43:15   2/18/2001 12:49:16  5/12/2003   RA
newaddress.txt       122 bytes  txt   5/13/2003 12:42:17  5/13/2003 12:42:18  5/13/2003   A
skiways-getafix.doc  21.0 KB    doc   5/13/2003 12:32:00  5/13/2003 11:58:10  5/13/2003   A
todo.txt             122 bytes  txt   5/13/2003 12:37:54  5/13/2003 12:40:48  5/13/2003   A
april                                 5/8/2003 14:41:44   5/8/2003 14:41:44   5/8/2003
greenfield.do        19.5 KB    do    5/8/2003 14:43:00   5/8/2003 14:34:16   5/12/2003   A

  Offset    0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15   Access
00000000   EB 3E 90 2B 23 5B 4A 30  49 48 43 00 02 01 01 00   ë>▮+#[J0IHC.....
00000016   02 E0 00 40 0B F0 09 00  12 00 02 00 00 00 00 00   .à.@.ð..........
00000032   00 00 00 00 00 00 29 1D  2B F7 16 53 41 4C 45 53   ......).+÷.SALES
00000048   20 20 20 20 20 20 46 41  54 31 32 20 20 20 F1 7D         FAT12   ñ}
00000064   FA 33 C9 8E D1 BC FC 7B  16 07 BD 78 00 C5 76 00   ú3É▮Ñ¼ü{..½x.Åv.
00000080   1E 56 16 55 BF 22 05 89  7E 00 89 4E 02 B1 0B FC   .V.U¿".▮~.▮N.±.ü
00000096   F3 A4 06 1F BD 00 7C C6  45 FE 0F 8B 46 18 88 45   ó¤..½.|ÆEþ.▮F.▮E
00000112   F9 FB 38 66 24 7C 04 CD  13 72 3C 8A 46 10 98 F7   ùû8f$|.Í.r<▮F.▮÷
00000128   66 16 03 46 1C 13 56 1E  03 46 0E 13 D1 50 52 89   f..F..V..F..ÑPR▮
00000144   46 FC 89 56 FE B8 20 00  8B 76 11 F7 E6 8B 5E 0B   Fü▮Vþ¸ .▮v.÷æ▮^.
00000160   03 C3 48 F7 F3 01 46 FC  11 4E FE 5A 58 BB 00 07   .ÃH÷ó.Fü.NþZX»..
```

Figure 3 Using WinHex's *Interpret Image File as Disk* feature to examine a FAT file system and recover deleted files and directories (disk image available at http://www.disclosedigital.com/downloads.html).

WinHex. In the event that a different disk geometry needs to be set, WinHex provides a *Set Disk Parameters* option on the Tools—Disk Tools menu that can be used to specify different CHS values when viewing a physical disk or a different number of clusters when viewing a logical volume.

Recovery

In addition to examining active files, digital evidence examiners often need to salvage deleted data from a disk. WinHex can interpret FAT and NTFS file systems and can automatically recover deleted files and directories on these file systems. In some instances, it is necessary to instruct WinHex to parse the file system using the *Interpret Image File as Disk* option on the Specialist menu. Fig. 3 shows WinHex being used to recover the deleted file "greenfield.doc" and directory "april" from a FAT formatted disk.

Using the disk image from "FAT Undelete Test #1" (Carrier, 2004a), WinHex successfully reconstructed the directory structure and recovered deleted files that were not fragmented.

When recovering deleted files on FAT file systems, WinHex assumes that data is located in contiguous clusters. This assumption breaks down when dealing with fragmented files such as those in bold in Table 1, potentially causing clusters from other files to be associated with the undeleted file. For comparison, this author used WinHex and EnCase to recover 92 deleted photographs from a memory card. The MD5 hashes of the files recovered using both tools were calculated

Table 1 WinHex results for "FAT Undelete Test #1" (Carrier, 2004a)

File	Size	MD5
#ing.dat	780	59b20779f69ff9f0ac5fcd2c38835a79
#rag1.dat	**1584**	**c4f3b9a0f17d464f4fc61b94ecf6cc21**
#rag2.dat	**3873**	**965370acfa85aa7e26ab04bbd45cdcae**
#ult1.dat	3801	ffd27bd782bdce67750b6b9ee069d2ef
#ir1\Mult2.dat	1715	59cf0e9cd107bc1e75afb7374f6e05bb
#ir1\Dir2\Frag3.dat	**2027**	**a7a14ac62f79fdea4056ed5e8bcf97ef**

using Maresware's hash utility and then compared using hashcmp as shown here:

```
E:\>hashcmp encase.txt winhex.txt -h

     ╔══════════════════════════════════════════════════════════════════╗
     ║                              HASHCMP                             ║
     ║                     Registered to: Eoghan Casey                  ║
     ║                                                                  ║
     ║          Ver. 02.03.22.11.33√ (32 bit) ser no# 1031306760        ║
     ║                                                                  ║
     ║     Portions Copyright 1998-2002 by Mares and Company, LLC       ║
     ║              (770)237-8870  http://www.dmares.com                ║
     ╚══════════════════════════════════════════════════════════════════╝

Getting number of records in each file. This may take some time.

Processed     27
Based on field starting at 89, for 32 characters
Found the following entry in encase.txt not in winhex.txt
E:\encase\_1010100.JPG        554A921D1F30FFBB48DF8C6FD1165DE7
Processed     81
Based on field starting at 89, for 32 characters
Found the following entry in encase.txt not in winhex.txt
E:\encase\_1010018.JPG        E98A9E42E0E35B440504CB3A699303F7
Based on field starting at 89, for 32 characters
Found the following entry in encase.txt not in winhex.txt
E:\encase\_1010086.JPG        F02A9CBC19F0F67626DD0EFDDFD56C71
Processed     92   Compare file records in 1 not in 2

Processed      0
Based on field starting at 89, for 32 characters
Found the following entry in winhex.txt not in encase.txt
E:\winhex\#1010100.jpg        2FA63B26E866CDE8AAFC05147F260AA4

Based on field starting at 89, for 32 characters
Found the following entry in winhex.txt not in encase.txt
E:\winhex\#1010086.jpg        D04E616855C0B3F110CA6D777D83A10D

Based on field starting at 89, for 32 characters
Found the following entry in winhex.txt not in encase.txt
E:\winhex\#1010018.jpg        FC5945833F88C99930B251C0701536AC
Processed     92   Compare file records in 2 not in 1

Found 6 mismatches
```

This output indicates that there were differences between the same files recovered by both tools—these mismatches were due to different recovery methods. When undeleting files, EnCase skips clusters that are allocated to other files whereas WinHex copies contiguous clusters, including clusters that are allocated to active files. The approach implemented by EnCase also breaks down when portions of a deleted file are overwritten by another file that is subsequently deleted. Ultimately, no single method will always be successful in all circumstances. This emphasizes the importance of using multiple tools and being aware of the assumptions they make.

On NTFS, WinHex displays deleted files in the context of the file system and, for convenience, also assembles deleted files and directories that are still referenced by the file system in an area called "Lost & Found" as shown in Fig. 4.

Using the "NTFS Undelete (and leap year) Test #1" (Carrier, 2004b), WinHex correctly displayed dates as 02/29/2004 and recovered all files except the alternate data stream (ADS) associated with file "mult1.dat" (see Table 2). Although WinHex correctly associated the data in this ADS with the parent file "mult1.dat", it did not provide an automated way to recover the data in the ADS. However, this alternate data stream was detected using WinHex's *Create Drive Contents Table* feature with the ADS option enabled, and the sectors associated with this ADS were listed to help the examiner locate the associated data on disk.

Although WinHex was able to recover "sing2.dat"—a single cluster file in a directory whose MFT entry had been reallocated, it was not able to associate the file with its original directory. Other specialized forensic tools including EnCase

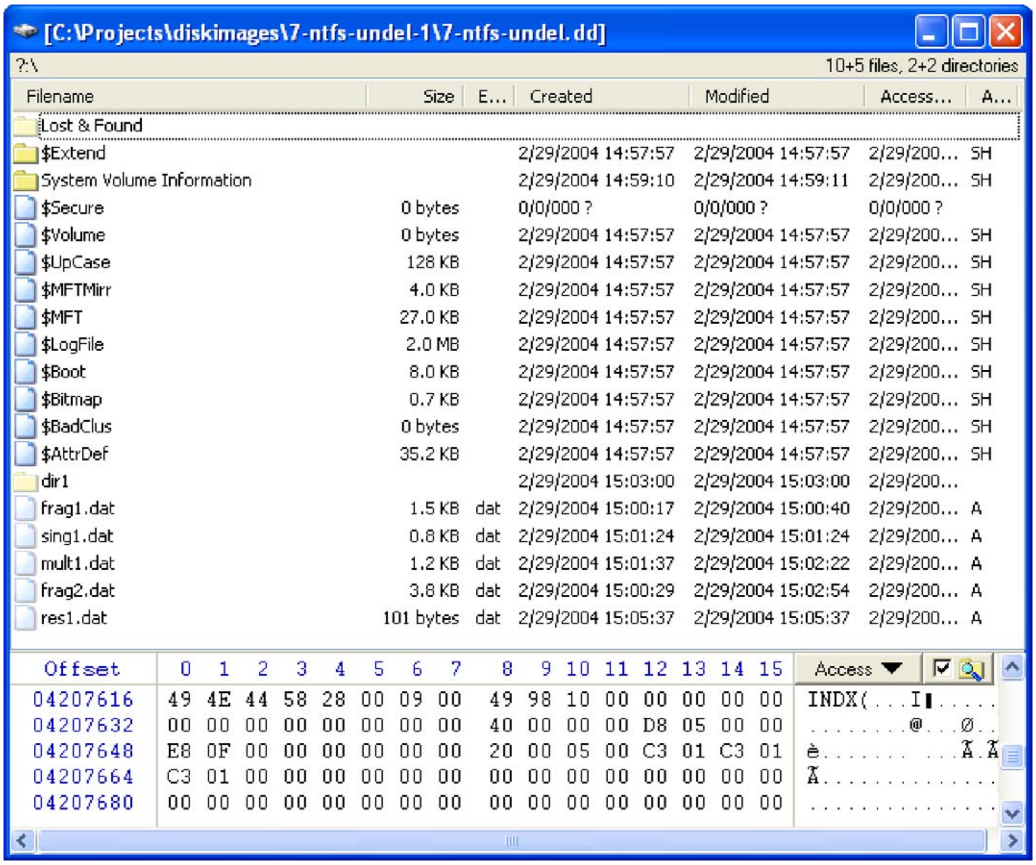

```
[C:\Projects\diskimages\7-ntfs-undel-1\7-ntfs-undel.dd]
?:\                                                        10+5 files, 2+2 directories
Filename                     Size   E...  Created             Modified            Access...      A...
Lost & Found
$Extend                                   2/29/2004 14:57:57  2/29/2004 14:57:57  2/29/200...  SH
System Volume Information                 2/29/2004 14:59:10  2/29/2004 14:59:11  2/29/200...  SH
$Secure                      0 bytes      0/0/000 ?           0/0/000 ?           0/0/000 ?
$Volume                      0 bytes      2/29/2004 14:57:57  2/29/2004 14:57:57  2/29/200...  SH
$UpCase                      128 KB       2/29/2004 14:57:57  2/29/2004 14:57:57  2/29/200...  SH
$MFTMirr                     4.0 KB       2/29/2004 14:57:57  2/29/2004 14:57:57  2/29/200...  SH
$MFT                         27.0 KB      2/29/2004 14:57:57  2/29/2004 14:57:57  2/29/200...  SH
$LogFile                     2.0 MB       2/29/2004 14:57:57  2/29/2004 14:57:57  2/29/200...  SH
$Boot                        8.0 KB       2/29/2004 14:57:57  2/29/2004 14:57:57  2/29/200...  SH
$Bitmap                      0.7 KB       2/29/2004 14:57:57  2/29/2004 14:57:57  2/29/200...  SH
$BadClus                     0 bytes      2/29/2004 14:57:57  2/29/2004 14:57:57  2/29/200...  SH
$AttrDef                     35.2 KB      2/29/2004 14:57:57  2/29/2004 14:57:57  2/29/200...  SH
dir1                                      2/29/2004 15:03:00  2/29/2004 15:03:00  2/29/200...
frag1.dat                    1.5 KB  dat  2/29/2004 15:00:17  2/29/2004 15:00:40  2/29/200...  A
sing1.dat                    0.8 KB  dat  2/29/2004 15:01:24  2/29/2004 15:01:24  2/29/200...  A
mult1.dat                    1.2 KB  dat  2/29/2004 15:01:37  2/29/2004 15:02:22  2/29/200...  A
frag2.dat                    3.8 KB  dat  2/29/2004 15:00:29  2/29/2004 15:02:54  2/29/200...  A
res1.dat                   101 bytes dat  2/29/2004 15:05:37  2/29/2004 15:05:37  2/29/200...  A

  Offset    0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15   Access
 04207616  49 4E 44 58 28 00 09 00  49 98 10 00 00 00 00 00   INDX(...I▮.....
 04207632  00 00 00 00 00 00 00 00  40 00 00 00 D8 05 00 00   ........@...Ø..
 04207648  E8 0F 00 00 00 00 00 00  20 00 05 00 C3 01 C3 01   è.......  ...Ã.Ã
 04207664  C3 01 00 00 00 00 00 00  00 00 00 00 00 00 00 00   Ã..............
 04207680  00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ...............
```

Figure 4 Using WinHex's *Interpret Image File as Disk* feature to examine a NTFS file system and recover deleted files and directories.

were similarly unable to associate this file with its original directory.

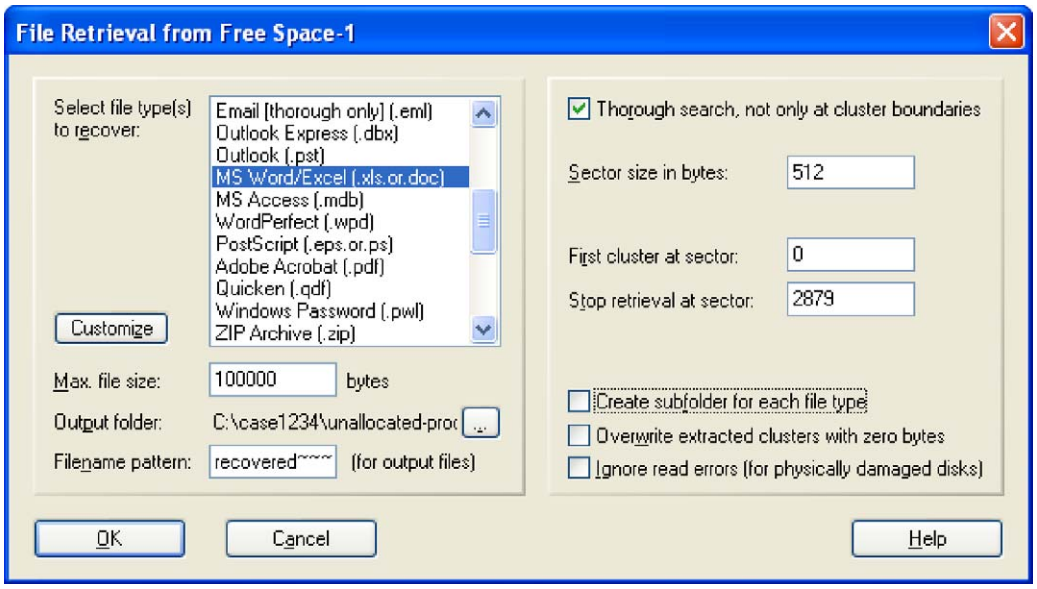
Notably, when automatically interpreting file systems, WinHex converts date–time stamps to the time zone of the examination system. Therefore, examiners must be cognizant of this adjustment as discussed in Boyd and Forster (2004). The next section presents a number of other ways that WinHex can be used to recover deleted data.

Unallocated and file slack space

Keep in mind that criminals often take steps to conceal their crimes, and deleted data often contains the most incriminating digital evidence. Therefore, it is often fruitful to scour unallocated and file slack space for useful data. WinHex can be used to recover both file slack and unallocated space using the *Gather Slack Space* and *Gather Free Space* options on the Specialist menu.

Table 2 WinHex results for "NTFS Undelete (and leap year) Test #1" (Carrier, 2004b)

File	Size	MD5
res1.dat	101	9036637712B491904CD0BFBDBE648453
sing1.dat	780	59B20779F69FF9F0AC5FCD2C38835A79
mult1.dat	3801	FFD27BD782BDCE67750B6B9EE069D2EF
mult1.dat:ADS	Not recovered	Not recovered
dir1\mult2.dat	1715	59CF0E9CD107BC1E75AFB7374F6E05BB
frag1.dat	1584	7A3BC5B763BEF201202108F4BA128149
frag2.dat	3873	0E80AB84EF0087E60DFC67B88A1CF13E
dir1\dir2\frag3.dat	2027	21121699487F3FBBDB9A4B3391B6D3E0
sing2.dat	1005	C229626F6A71B167AD7E50C4F2FCCDB1


File Retrieval from Free Space-1

Select file type(s) to recover:
Email (thorough only) (.eml)
Outlook Express (.dbx)
Outlook (.pst)
MS Word/Excel (.xls.or.doc)
MS Access (.mdb)
WordPerfect (.wpd)
PostScript (.eps.or.ps)
Adobe Acrobat (.pdf)
Quicken (.qdf)
Windows Password (.pwl)
ZIP Archive (.zip)

Max. file size: 100000 bytes
Output folder: C:\case1234\unallocated-proc
Filename pattern: recovered~~~~ (for output files)

[✓] Thorough search, not only at cluster boundaries
Sector size in bytes: 512
First cluster at sector: 0
Stop retrieval at sector: 2879
[ ] Create subfolder for each file type
[ ] Overwrite extracted clusters with zero bytes
[ ] Ignore read errors (for physically damaged disks)

Figure 5 File carving based on class characteristics.

Furthermore, WinHex can extract the space that is not assigned to any of the partitions on a disk if desired using the *Gather Inter-partition Space* option.

WinHex also has the capability to salvage deleted files of a certain type from unallocated space. Fig. 5 shows the *Recover File by Type* feature on the Tools—Disk Tools menu being used to extract Word documents from unallocated space. This approach to salvaging files is commonly referred to as "file carving" because it uses common header characteristics to identify the beginning of a file and then carves out the data in continuous clusters until it reaches a characteristic that marks the end of the file or until it reaches a predefined maximum file size.

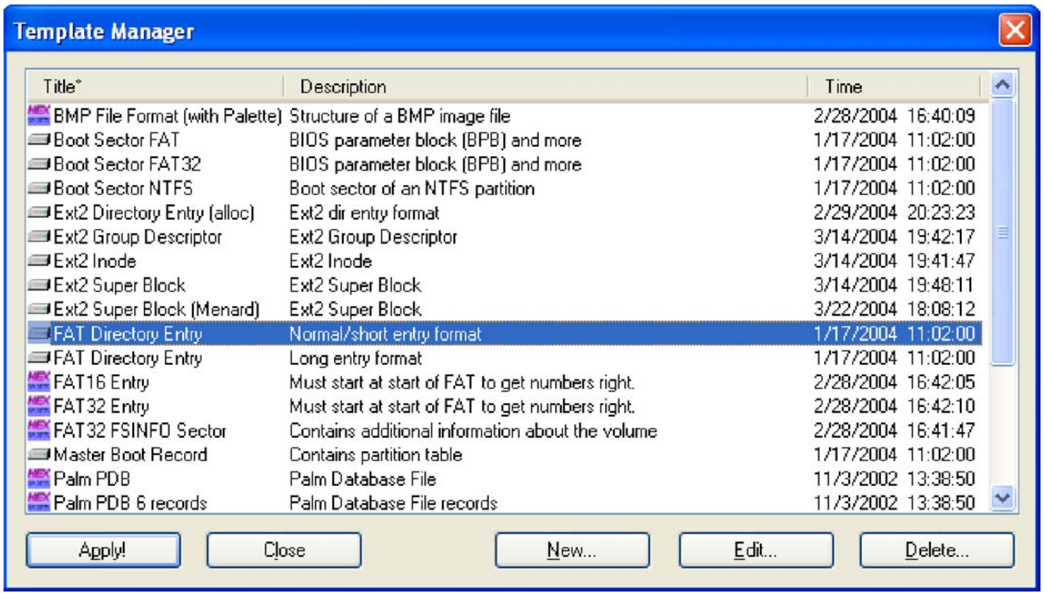
By default, WinHex's *Recover File by Type* file carving feature only searches the beginning of each cluster for headers. This approach is efficient, taking advantage of the fact that files usually start at cluster boundaries, but in some situations this assumption may not be suitable (e.g., when the desired files are embedded within other files). Therefore, the "Thorough search" option in Fig. 5 instructs WinHex to inspecting all data on the disk for file headers, ignoring cluster and sector boundaries. The results of this activity are summarized in the following log that is automatically generated by WinHex:

```
3/22/2004, 22:52:29
C:\case1234\prepare\unallocated-raw\Free Space-1
Scope: 499712 - 499711
Disk layout options: not used, examining all offsets
Fixed/maxmimum file size: 100000

MS Word/Excel (xls.or.doc), header: D0CF11E0

000000 - 099999: recovered0000.xls.or.doc
019968 - 119967: recovered0001.xls.or.doc
097280 - 197279: recovered0002.xls.or.doc
117248 - 217247: recovered0003.xls.or.doc

3/22/2004, 22:52:29
4 headers were found. 4 files were retrieved.
```

WinHex can also be used to recover files with specific patterns in their names using the *Recover File by Name* feature on the Tools—Disk Tools menu.

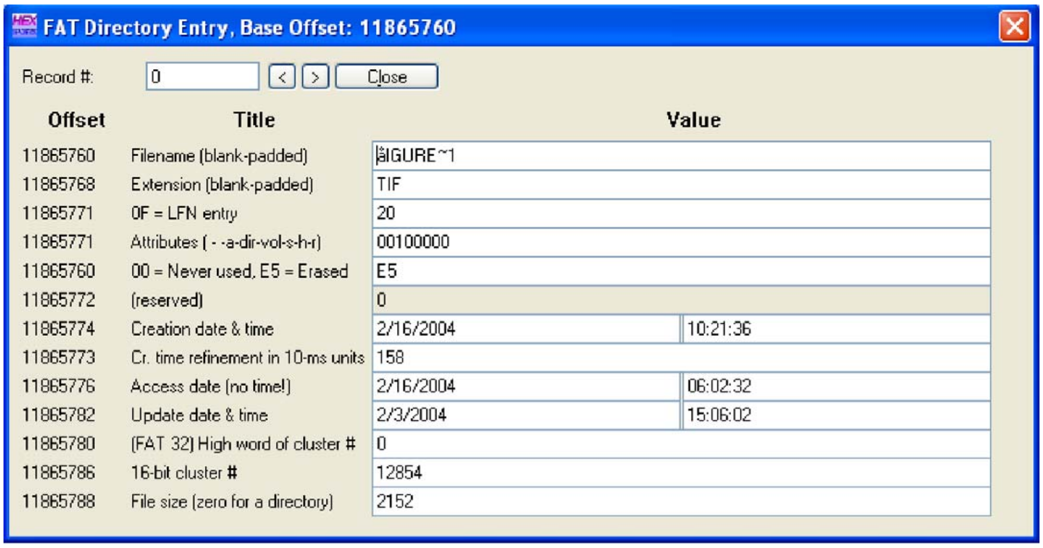
Deleted data can also be salvaged using WinHex's "template" feature. Users can create customized templates that tell WinHex how to interpret certain data structures. For example, take a directory entry found in unallocated space as shown here in hexadecimal:

```
Offset       0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15

011865664   E5 69 00 66 00 00 00 FF  FF FF FF 0F 00 9F FF FF   åi.f...ÿÿÿÿ..Ÿÿÿ
011865680   FF FF FF FF FF FF FF FF  FF FF 00 00 FF FF FF FF   ÿÿÿÿÿÿÿÿÿÿ..ÿÿÿÿ
011865696   E5 6E 00 61 00 6C 00 2D  00 73 00 0F 00 9F 74 00   ån.a.l.-.s...Ÿt.
011865712   6F 00 72 00 61 00 67 00  65 00 00 00 2E 00 74 00   o.r.a.g.e.....t.
011865728   E5 66 00 69 00 67 00 75  00 72 00 0F 00 9F 65 00   åf.i.g.u.r...Ÿe.
011865744   35 00 2D 00 65 00 78 00  74 00 00 00 65 00 72 00   5.-.e.x.t...e.r.
011865760   E5 49 47 55 52 45 7E 31  54 49 46 20 00 9E B2 52   åIGURE~1TIF .ž²R
```


Template Manager

Title	Description	Time
BMP File Format (with Palette)	Structure of a BMP image file	2/28/2004 16:40:09
Boot Sector FAT	BIOS parameter block (BPB) and more	1/17/2004 11:02:00
Boot Sector FAT32	BIOS parameter block (BPB) and more	1/17/2004 11:02:00
Boot Sector NTFS	Boot sector of an NTFS partition	1/17/2004 11:02:00
Ext2 Directory Entry (alloc)	Ext2 dir entry format	2/29/2004 20:23:23
Ext2 Group Descriptor	Ext2 Group Descriptor	3/14/2004 19:42:17
Ext2 Inode	Ext2 Inode	3/14/2004 19:41:47
Ext2 Super Block	Ext2 Super Block	3/14/2004 19:48:11
Ext2 Super Block (Menard)	Ext2 Super Block	3/22/2004 18:08:12
FAT Directory Entry	Normal/short entry format	1/17/2004 11:02:00
FAT Directory Entry	Long entry format	1/17/2004 11:02:00
FAT16 Entry	Must start at start of FAT to get numbers right.	2/28/2004 16:42:05
FAT32 Entry	Must start at start of FAT to get numbers right.	2/28/2004 16:42:10
FAT32 FSINFO Sector	Contains additional information about the volume	2/28/2004 16:41:47
Master Boot Record	Contains partition table	1/17/2004 11:02:00
Palm PDB	Palm Database File	11/3/2002 13:38:50
Palm PDB 6 records	Palm Database File records	11/3/2002 13:38:50

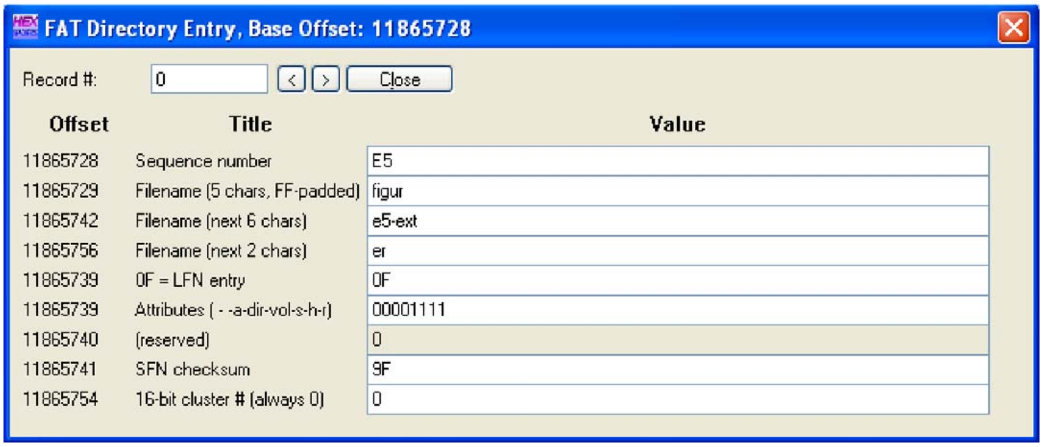
Figure 6 Applying a template using the WinHex *Template Manager*.


a

FAT Directory Entry, Base Offset: 11865760

Record #: 0

Offset	Title	Value	
11865760	Filename (blank-padded)	ßIGURE~1	
11865768	Extension (blank-padded)	TIF	
11865771	0F = LFN entry	20	
11865771	Attributes (- -a-dir-vol-s-h-r)	00100000	
11865760	00 = Never used, E5 = Erased	E5	
11865772	(reserved)	0	
11865774	Creation date & time	2/16/2004	10:21:36
11865773	Cr. time refinement in 10-ms units	158	
11865776	Access date (no time!)	2/16/2004	06:02:32
11865782	Update date & time	2/3/2004	15:06:02
11865780	(FAT 32) High word of cluster #	0	
11865786	16-bit cluster #	12854	
11865788	File size (zero for a directory)	2152	


b

FAT Directory Entry, Base Offset: 11865728

Record #: 0

Offset	Title	Value
11865728	Sequence number	E5
11865729	Filename (5 chars, FF-padded)	figur
11865742	Filename (next 6 chars)	e5-ext
11865756	Filename (next 2 chars)	er
11865739	0F = LFN entry	0F
11865739	Attributes (- -a-dir-vol-s-h-r)	00001111
11865740	(reserved)	0
11865741	SFN checksum	9F
11865754	16-bit cluster # (always 0)	0

Figure 7 (a) Applying the WinHex "FAT Directory Entry (Normal/short entry format)" template to a FAT directory entry found in unallocated space. (b) Applying the WinHex "FAT Directory Entry (Long entry format)" template to a FAT directory entry containing part of the filename "figure5-external-storage.tif".

An examiner can view this directory in a more readable form using a template packaged with WinHex by placing the cursor at the beginning of the data structure, opening the *Template Manager* on the View menu, and applying the appropriate template as shown in Fig. 6.

Using the WinHex templates for short and long file names shown in Fig. 6 and applying them to the appropriate directory entries results in the interpreted data shown in Fig. 7.

Although the focus of this section was on Windows systems, the file carving and template features can be applied to other file systems such as EXT2 and EXT3.

Extended file systems (EXT2/EXT3)

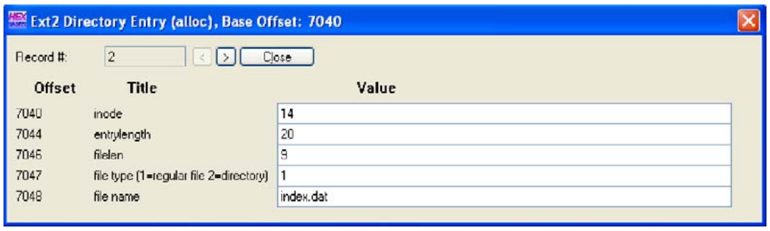
The EXT3 and EXT2 file systems are essentially the same except that EXT3 has an additional journal file that is used to expedite system recovery after a crash. Although WinHex does not directly support the EXT2 and EXT3 file systems, individuals can create custom templates to tell the program how to interpret any data structure. Like FAT file systems, EXT2/EXT3 file systems have a root directory and something like a file allocate table; called the inode table. For instance, the root directory of an EXT2 file system is shown here in hexadecimal format with inode numbers in bold:

```
Offset       0  1  2  3  4  5  6  7   8  9  A  B  C  D  E  F

00007000    02 00 00 00 0C 00 01 02  2E 00 00 00 02 00 00 00   ................
00007010    0C 00 02 02 2E 2E 00 00  0B 00 00 00 14 00 0A 02   ................
00007020    6C 6F 73 74 2B 66 6F 75  6E 64 00 00 0C 00 00 00   lost+found......
00007030    14 00 0A 02 64 69 72 65  63 74 6F 72 79 31 00 00   ....directory1..
00007040    0E 00 00 00 14 00 09 01  69 6E 64 65 78 2E 64 61   ........index.da
00007050    74 00 00 00 15 00 00 00  20 00 05 01 66 69 6C 65   t....... ...file
00007060    33 70 00 00 00 00 00 00  10 00 05 01 66 69 6C 65   3p..........file
00007070    32 32 2E 73 11 00 00 00  24 00 05 01 66 69 6C 65   22.s....$...file
00007080    34 34 2E 73 00 00 00 00  14 00 0A 01 68 61 6E 64   44.s........hand
00007090    6C 65 2E 70 64 66 00 00  13 00 00 00 68 03 0C 01   le.pdf......h...
000070A0    44 43 50 5F 31 37 32 32  2E 4A 50 47 00 00 00 00   DCP_1722.JPG....
000070B0    54 03 0A 01 2E 66 69 6C  65 33 2E 73 77 70 00 00   T....file3.swp..
000070C0    00 00 00 00 40 03 0B 01  2E 66 69 6C 65 33 2E 73   ....@....file3.s
000070D0    77 70 78 00 00 00 00 00  00 00 00 00 00 00 00 00   wpx.............
```

Using a WinHex template created by this author for interpreting EXT2 root directory entries (available at http://www.x-ways.net/winhex/templates/index.html) enables us to view this information in a more readable form as shown in Fig. 8.

The directory entry in Fig. 8 indicates that the file "index.dat" is assigned inode 14. Using another WinHex template created by this author for interpreting inode tables on EXT2/EXT3 systems enables us to view the data in a more readable form as shown in Fig. 9.

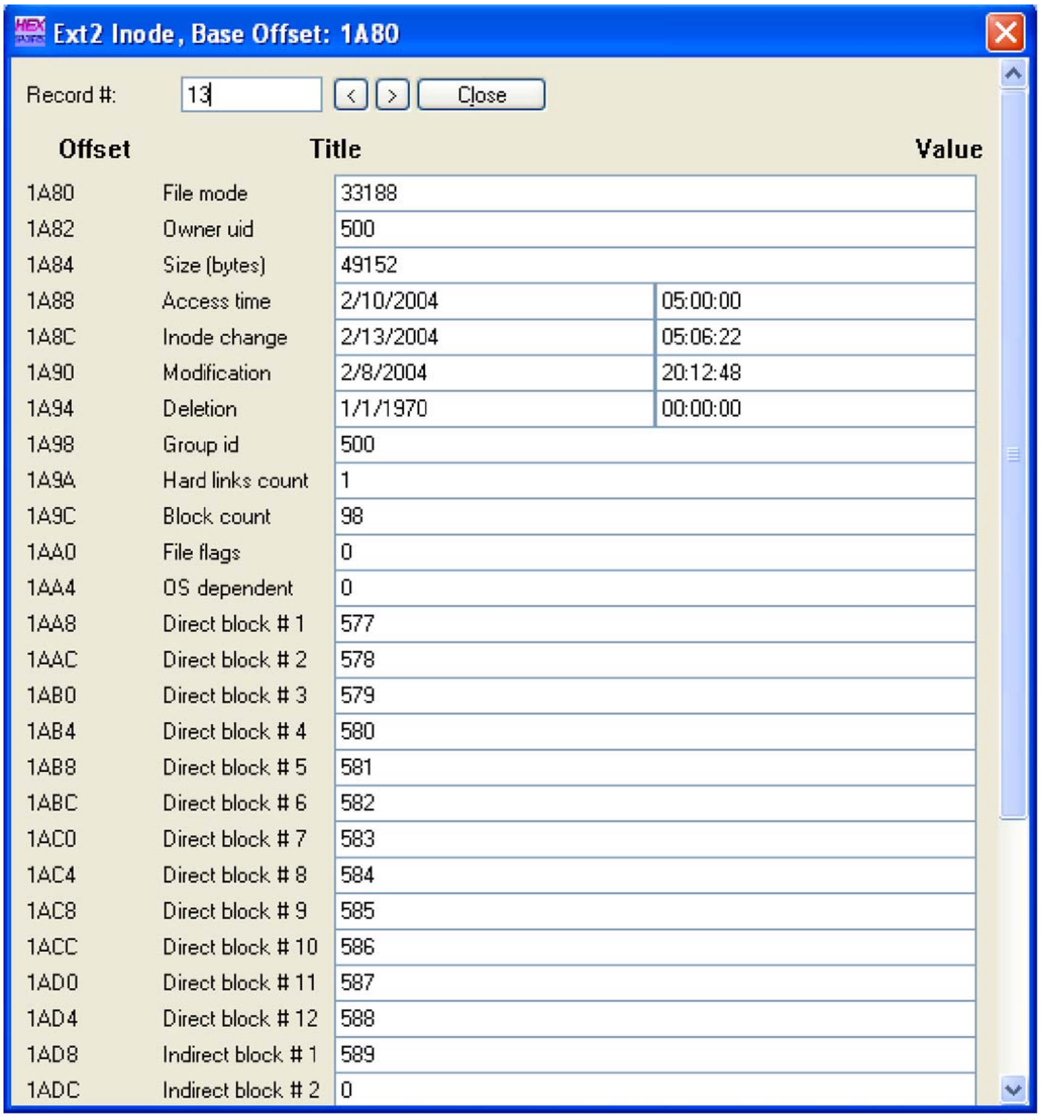
The inode shown in Fig. 9 indicates that the file starts at block 577 and that indirect block


Ext2 Directory Entry (alloc), Base Offset: 7040

Record #: 2 Close

Offset	Title	Value
7040	inode	14
7044	entrylength	20
7046	filelen	9
7047	file type (1=regular file 2=directory)	1
7048	file name	index.dat

Figure 8 Using WinHex's template feature to examine the Root directory of an EXT2 file system (disk image available at http://www.disclosedigital.com/downloads.html).



Ext2 Inode, Base Offset: 1A80

Record #: 13

Offset	Title	Value	
1A80	File mode	33188	
1A82	Owner uid	500	
1A84	Size (bytes)	49152	
1A88	Access time	2/10/2004	05:00:00
1A8C	Inode change	2/13/2004	05:06:22
1A90	Modification	2/8/2004	20:12:48
1A94	Deletion	1/1/1970	00:00:00
1A98	Group id	500	
1A9A	Hard links count	1	
1A9C	Block count	98	
1AA0	File flags	0	
1AA4	OS dependent	0	
1AA8	Direct block # 1	577	
1AAC	Direct block # 2	578	
1AB0	Direct block # 3	579	
1AB4	Direct block # 4	580	
1AB8	Direct block # 5	581	
1ABC	Direct block # 6	582	
1AC0	Direct block # 7	583	
1AC4	Direct block # 8	584	
1AC8	Direct block # 9	585	
1ACC	Direct block # 10	586	
1AD0	Direct block # 11	587	
1AD4	Direct block # 12	588	
1AD8	Indirect block # 1	589	
1ADC	Indirect block # 2	0	

Figure 9 Using WinHex's template feature to examine an inode on an EXT2 file system (disk image available at http://www.disclosedigital.com/downloads.html).

589 contains a list of additional blocks that are assigned to this file. The contents of block 589 (589 × 1024 = byte offset 603136) is shown here in hexadecimal format.

```
Offset       0   1   2   3   4   5   6   7    8   9  10  11  12  13  14  15

00603136    4E  02  00  00  4F  02  00  00   50  02  00  00  51  02  00  00    N...O...P...Q...
00603152    52  02  00  00  53  02  00  00   54  02  00  00  55  02  00  00    R...S...T...U...
00603168    56  02  00  00  57  02  00  00   58  02  00  00  59  02  00  00    V...W...X...Y...
00603184    5A  02  00  00  5B  02  00  00   5C  02  00  00  5D  02  00  00    Z...[...\...]...
00603200    5E  02  00  00  5F  02  00  00   60  02  00  00  61  02  00  00    ^..._...`...a...
00603216    62  02  00  00  63  02  00  00   64  02  00  00  65  02  00  00    b...c...d...e...
00603232    66  02  00  00  67  02  00  00   68  02  00  00  69  02  00  00    f...g...h...i...
00603248    6A  02  00  00  6B  02  00  00   6C  02  00  00  6D  02  00  00    j...k...l...m...
00603264    6E  02  00  00  6F  02  00  00   70  02  00  00  71  02  00  00    n...o...p...q...
```

This "alphabet soup" is simply a list of block numbers in little endian format which equate to blocks 590—625 (0000024E—00000271 hexadecimal). Once the data associated with a file has been located, a WinHex template may be available to interpret the data or the data can be marked in WinHex and exported to a file for further processing using other tools.

Although using templates to interpret file systems can be useful for certain tasks such as teaching novices and verifying important details at a low level, it is rarely feasible to perform a complete forensic examination in this manner. On large disks it would take an inordinate amount of time to extract all data and metadata from the file system using the techniques demonstrated in this section.

Harvesting

The aim of harvesting during the forensic examination process is to gather metadata relating to directories, files, and fragments salvaged during the recovery process. File system characteristics are the most common form of metadata. WinHex can generate a list of existing and deleted files and directories with associated characteristics, including date—time stamps, size, allocated clusters, and hash values using the *Create Drive Contents Table* option on the Specialist menu.

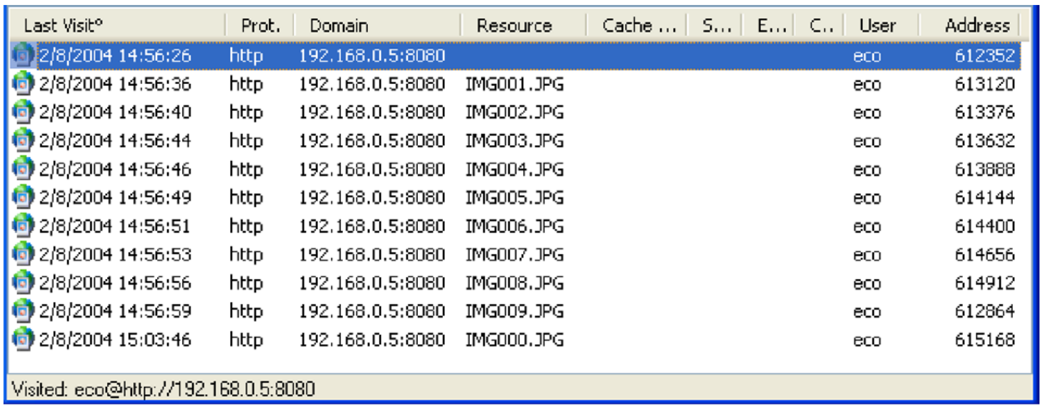
By default, the *Create Drive Contents Table* feature lists active and deleted files that are still referenced by the file system (i.e. in directories on FAT and in the $MFT on NTFS). Enabling the "Particularly thorough search" option instructs WinHex to look beyond the file system, and to methodically scan the entire disk for deleted directories or MFT entries. Specifically, on FAT systems, this thorough search feature searches the beginning of every sector for the pattern generally associated with a directory (". .."). According to the developer, using the "Particularly thorough search" option on NTFS causes WinHex to look for the pattern generally associated with MFT entries ("FILE") at the beginning of each cluster and at the beginning of each 1024-byte entity within that cluster, if a cluster is larger than 1024 bytes. These facilities for generating a list of deleted directory and MFT entries are useful for verifying the results of other tools such as EnCase's Recover Folders feature.

By design, WinHex does not calculate hash values of deleted files, making it necessary to save the recovered files to disk and calculate their MD5 values with a tool specifically designed for this purpose such as Maresware's `hash` utility as demonstrated in the recovery section of this paper. This approach to calculating hash values is recommended even when using integrated forensic media examination applications since internal MD5 calculations may not always be correct.

After the basic file characteristics are harvested, it is then necessary to look within files for additional metadata. Compound files such as Microsoft Office and Outlook files have date—time stamps and other metadata embedded in them. As noted earlier, custom WinHex templates can be created to interpret different data structures. In the case of index.dat and INFO files, WinHex can invoke the X-Ways Trace utility on the Tools menu to interpret the data as shown in Fig. 10.

Notably, the X-Ways Trace utility converts date—time stamps into the local time zone of the examination computer when the *Convert to Local Time* option on the Edit menu is enabled which can create confusion if not realized (Boyd and Forster, 2004).

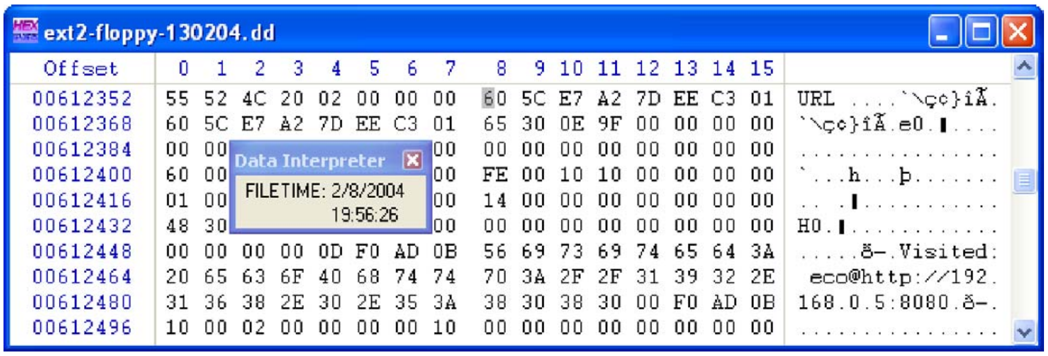
The *Data Interpreter* feature on WinHex's Options menu can also be used to interpret embedded date—time stamps as shown in Fig. 11. Note the discrepancy between the date—time stamps in Figs. 10 and 11 caused by the time zone adjustment (605CE7A27DEEC301 = February 8, 2004 19:56:26 GMT).



Last Visit°	Prot.	Domain	Resource	Cache ...	S...	E...	C..	User	Address
2/8/2004 14:56:26	http	192.168.0.5:8080						eco	612352
2/8/2004 14:56:36	http	192.168.0.5:8080	IMG001.JPG					eco	613120
2/8/2004 14:56:40	http	192.168.0.5:8080	IMG002.JPG					eco	613376
2/8/2004 14:56:44	http	192.168.0.5:8080	IMG003.JPG					eco	613632
2/8/2004 14:56:46	http	192.168.0.5:8080	IMG004.JPG					eco	613888
2/8/2004 14:56:49	http	192.168.0.5:8080	IMG005.JPG					eco	614144
2/8/2004 14:56:51	http	192.168.0.5:8080	IMG006.JPG					eco	614400
2/8/2004 14:56:53	http	192.168.0.5:8080	IMG007.JPG					eco	614656
2/8/2004 14:56:56	http	192.168.0.5:8080	IMG008.JPG					eco	614912
2/8/2004 14:56:59	http	192.168.0.5:8080	IMG009.JPG					eco	612864
2/8/2004 15:03:46	http	192.168.0.5:8080	IMG000.JPG					eco	615168

Visited: eco@http://192.168.0.5:8080

Figure 10 An IE History index.dat file interpreted using X-Ways Trace (http://www.x-ways.net/trace/).



```
Offset     0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15
00612352  55 52 4C 20 02 00 00 00  60 5C E7 A2 7D EE C3 01  URL ....`\çc}îÃ.
00612368  60 5C E7 A2 7D EE C3 01  65 30 0E 9F 00 00 00 00  `\çc}îÃ.e0.▌....
00612384  00 00                00  00 00 00 00 00 00 00 00  ................
00612400  60 00                00  FE 00 10 10 00 00 00 00  `...h...þ.......
00612416  01 00                00  14 00 00 00 00 00 00 00  .. .▌...........
00612432  48 30                00  00 00 00 00 00 00 00 00  H0.▌............
00612448  00 00 00 00 0D F0 AD 0B  56 69 73 69 74 65 64 3A  .....ð-.Visited:
00612464  20 65 63 6F 40 68 74 74  70 3A 2F 2F 31 39 32 2E   eco@http://192.
00612480  31 36 38 2E 30 2E 35 3A  38 30 38 30 00 F0 AD 0B  168.0.5:8080.ð-.
00612496  10 00 02 00 00 00 00 10  00 00 00 00 00 00 00 00  ................
```

```
FILETIME: 2/8/2004
          19:56:26
```

Figure 11 A Window 64 bit FILETIME date—time stamp in an IE History (index.dat) file viewed using WinHex's *Data Interpreter* feature.

The *Data Interpreter* feature can convert several time formats, including 64 bit Windows FILETIME, 32 bit Windows/DOS, and UNIX date—time stamps.

Search and organization

WinHex can search a disk for text or hexadecimal values, allows for case sensitivity and Unicode when performing text searches, and can be configured with a wildcard character (e.g., ?, 3F), but does not have a regular expression search feature. In addition to searching for individual keywords, a list of keywords can be provided in the *Simultaneous Search* feature on the Specialist menu and the output can be saved to a file or to the WinHex Position Manager which enables the user to locate specific search hits on the disk. The output of the *Simultaneous Search* feature includes the logical file that contains the keyword but not the cluster number. The cluster number and associated file can be obtained by going to each keyword and reading the information in the Details window as shown in Fig. 12.

Using the disk image "NTFS Keyword Search #1" (Carrier, 2003b) shown in Fig. 12, WinHex located all of the keywords except "n-frag" which was split between two non-contiguous clusters in an allocated file. The implication here is that WinHex does not search the logical file system on the physical disk. For instance, in a homicide investigation, if an incriminating file named "personal-diary.txt" contains the word "murdered" but this keyword is split between two non-contiguous clusters, a keyword search for "murdered" using WinHex will not find this occurrence. Although WinHex can perform a logical search of a mounted volume using the Open Folder option on the File menu, this does not work with forensic images.

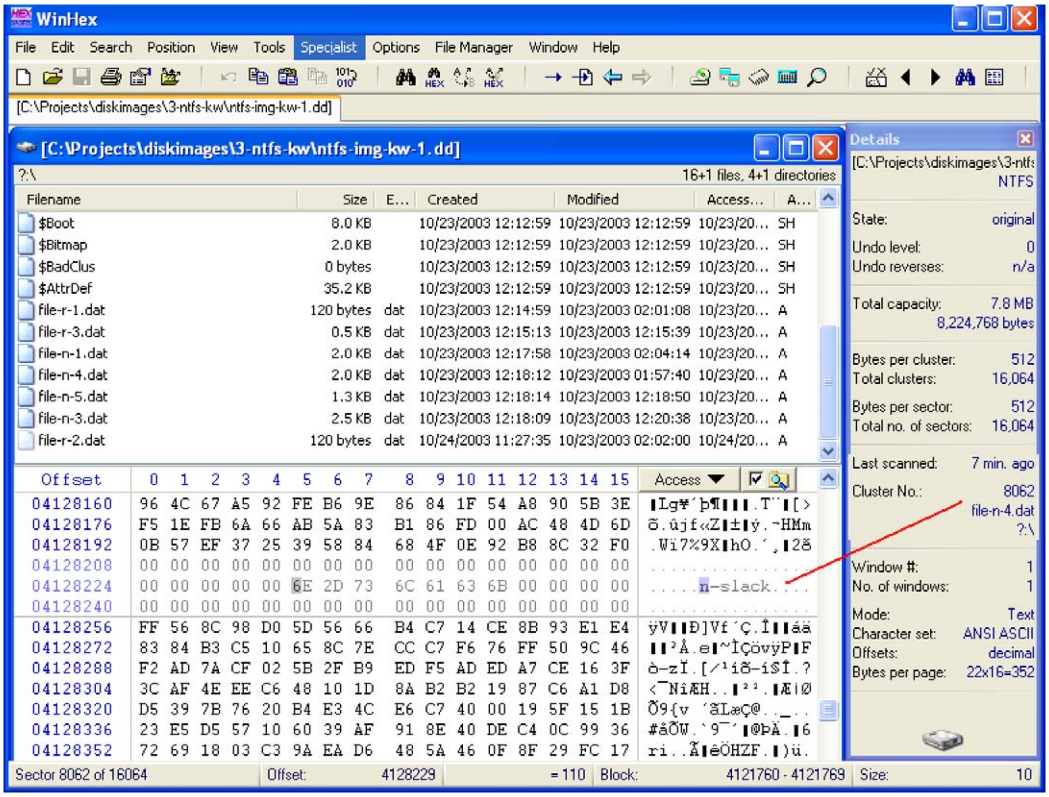
When a keyword is located in resident data (keywords 1—4 in Table 3), search results exported to a file or displayed in the Position Manager do not clearly indicate the associated file name, simply indicating that it was found in the MFT. To determine the file name it is necessary to place the cursor on the keyword in WinHex and click on the context sensitive Access button which will have an option to recover the file associated with the selected MFT entry as shown in Fig. 13. Notice that the Details window shows the number of the selected MFT entry.

Furthermore, for keywords in alternate data streams, WinHex associated the hit with the parent file or directory rather than with the ADS (keywords 3, 4, 9 and 10 in Table 3). Omitting the name of the ADS that contains a keyword could lead to misinterpretations of the search results. As an example, in a larceny investigation, if a file named "personal-finances.txt" had an ADS named "stolen-money.txt" containing a list of stolen sums of money, WinHex could be used to locate the list of stolen money but would not show the examiner that they were in the ADS named "stolen-money.txt", instead displaying the name of the parent file "personal-finances.txt".

Finally, WinHex did not recover the deleted file named "file-n-2.dat" and therefore did not indicate that the keyword "n-unalloc" was found in this deleted file. Actually, it would be a surprise if any tool could recover this file since the only reference to the filename is in the $Logfile and there is no way to associate specific clusters with that file name.

Be aware that WinHex does not provide the location of keywords relative to the beginning of logical files, only the offset from the beginning of the disk.

Using the "FAT Keyword Search #1" (Carrier, 2003c), WinHex located all of the keywords in the correct files except for those fragmented


WinHex
File Edit Search Position View Tools Specialist Options File Manager Window Help

```
[C:\Projects\diskimages\3-ntfs-kw\ntfs-img-kw-1.dd]
?:\                                                    16+1 files, 4+1 directories
Filename        Size       E...  Created              Modified             Access...     A...
$Boot           8.0 KB           10/23/2003 12:12:59  10/23/2003 12:12:59  10/23/20...   SH
$Bitmap         2.0 KB           10/23/2003 12:12:59  10/23/2003 12:12:59  10/23/20...   SH
$BadClus        0 bytes          10/23/2003 12:12:59  10/23/2003 12:12:59  10/23/20...   SH
$AttrDef        35.2 KB          10/23/2003 12:12:59  10/23/2003 12:12:59  10/23/20...   SH
file-r-1.dat    120 bytes  dat   10/23/2003 12:14:59  10/23/2003 02:01:08  10/23/20...   A
file-r-3.dat    0.5 KB     dat   10/23/2003 12:15:13  10/23/2003 12:15:39  10/23/20...   A
file-n-1.dat    2.0 KB     dat   10/23/2003 12:17:58  10/23/2003 02:04:14  10/23/20...   A
file-n-4.dat    2.0 KB     dat   10/23/2003 12:18:12  10/23/2003 01:57:40  10/23/20...   A
file-n-5.dat    1.3 KB     dat   10/23/2003 12:18:14  10/23/2003 12:18:50  10/23/20...   A
file-n-3.dat    2.5 KB     dat   10/23/2003 12:18:09  10/23/2003 12:20:38  10/23/20...   A
file-r-2.dat    120 bytes  dat   10/24/2003 11:27:35  10/23/2003 02:02:00  10/24/20...   A

  Offset    0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15   Access
04128160   96 4C 67 A5 92 FE B6 9E  86 84 1F 54 A8 90 5B 3E   ▮Lg¥´þ¶▮▮▮.T¨▮[>
04128176   F5 1E FB 6A 66 AB 5A 83  B1 86 FD 00 AC 48 4D 6D   õ.ûjf«Z▮±▮ý.¬HMm
04128192   0B 57 EF 37 25 39 58 84  68 4F 0E 92 B8 8C 32 F0   .Wï7%9X▮hO.´¸▮2ð
04128208   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
04128224   00 00 00 00 00 6E 2D 73  6C 61 63 6B 00 00 00 00   .....n-slack....
04128240   00 00 00 00 00 00 00 00  00 00 00 00 00 00 00 00   ................
04128256   FF 56 8C 98 D0 5D 56 66  B4 C7 14 CE 8B 93 E1 E4   ÿV▮▮Ð]Vf´Ç.Î▮▮áä
04128272   83 84 B3 C5 10 65 8C 7E  CC C7 F6 76 FF 50 9C 46   ▮▮³Å.e▮~ÌÇövÿP▮F
04128288   F2 AD 7A CF 02 5B 2F B9  ED F5 AD ED A7 CE 16 3F   ò-zÏ.[/¹íõ-í§Î.?
04128304   3C AF 4E EE C6 48 10 1D  8A B2 B2 19 87 C6 A1 D8   <¯NîÆH..▮²².▮Æ¡Ø
04128320   D5 39 7B 76 20 B4 E3 4C  E6 C7 40 00 19 5F 15 1B   Õ9{v ´ãLæÇ@.._..
04128336   23 E5 D5 57 10 60 39 AF  91 8E 40 DE C4 0C 99 36   #åÕW.`9¯'▮@ÞÄ.▮6
04128352   72 69 18 03 C3 9A EA D6  48 5A 46 0F 8F 29 FC 17   ri..Ã▮êÖHZF.▮)ü.
Sector 8062 of 16064    Offset:    4128229    = 110    Block:    4121760 - 4121769    Size:    10

Details
[C:\Projects\diskimages\3-ntfs
                          NTFS
State:                original
Undo level:                  0
Undo reverses:             n/a
Total capacity:         7.8 MB
               8,224,768 bytes
Bytes per cluster:         512
Total clusters:         16,064
Bytes per sector:          512
Total no. of sectors:   16,064
Last scanned:       7 min. ago
Cluster No.:              8062
                  file-n-4.dat
                           ?:\
Window #:                    1
No. of windows:              1
Mode:                     Text
Character set:      ANSI ASCII
Offsets:               decimal
Bytes per page:      22x16=352
```

Figure 12 The Details window provides information about data at the current cursor position, including the cluster and name of the associated file when applicable.

across non-contiguous clusters (in bold in Table 4). Again, this indicates that WinHex only performs a physical search of the disk and not a logical search of files. The fact that WinHex found the three "cross" strings indicates that it will find strings that cross file boundaries. Specifically, it will find keywords that start in a file and end in slack space, start in one file and end in another, and start in a file and end in an unallocated sector.

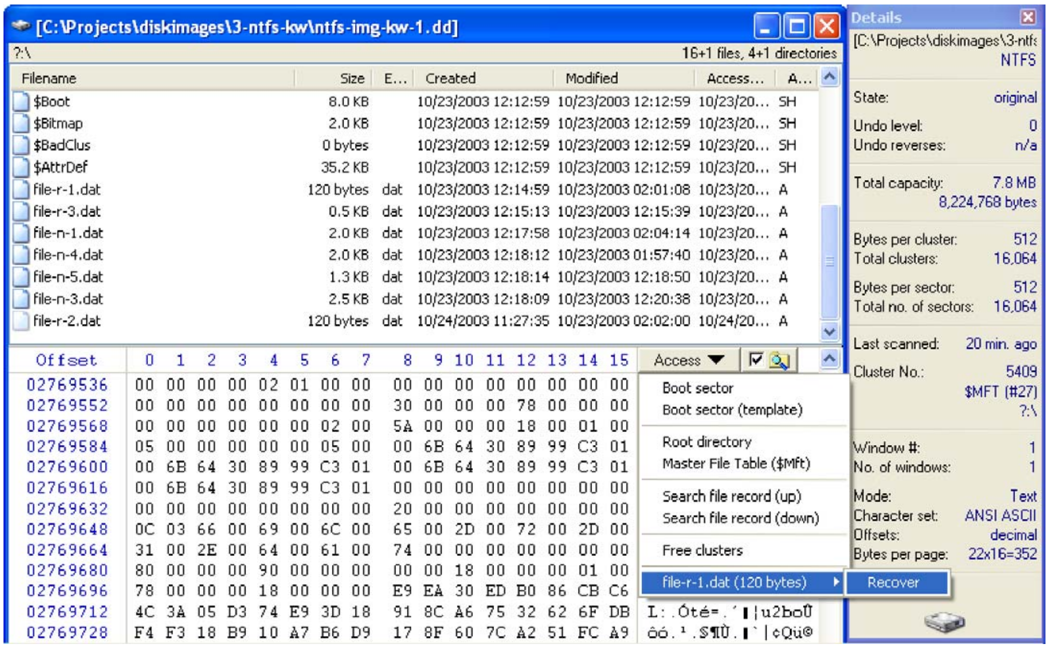
The *Gather Text* feature on the Specialist menu is useful for extracting human readable data on a disk. For instance, this feature can be applied to unallocated or file slack space to find useful text fragments. The *Text Passages* feature on the Search menu performs searches for a specified number of sequential letters, numbers, or punctuation marks or spaces.

Table 3 WinHex results for "NTFS Keyword Search #1" (Carrier, 2003b)

Num	String	Cluster	Cluster allocation
1	r-alloc	1342	?:\$LogFile
	r-alloc	5409	?:\$MFT #27 (file-r-1.dat)
2	r-unalloc	1350	?:\$LogFile
	r-unalloc	1915	?:\$LogFile
	r-unalloc	5423	?:\$MFT #34 (file-r-2.dat)
3	r-fads	1391	?:\$LogFile
	r-fads	5414	?:\$MFT #29 (file-r-3.dat)
4	r-dads	1528	?:\$LogFile
	r-dads	5415	?:\$MFT #30 (dir-r-4)
5	n-alloc	8050	?:\file-n-1.dat
6	n-unalloc	8053	Unallocated space
7	**n-frag**	**Not found**	**Not found**
8	n-slack	8062	?:\file-n-4.dat (slack)
9	n-fads	8067	?:\file-n-5.dat (slack)
10	n-dads	8068	?:\dir-n-6

Conclusions and recommendations

WinHex Specialist Edition is a valuable tool for experienced forensic examiners to validate findings obtained using other applications, and is useful for students to learn about file systems and data structures. In addition to most of the file inventory and data recovery capabilities necessary in a forensic examination tool, WinHex facilitates low-level examination of digital evidence using the *Data Interpreter* and template features. The template feature can also be used to edit structures such as a partition table or file allocation table, which can be



[C:\Projects\diskimages\3-ntfs-kw\ntfs-img-kw-1.dd]

?:\ 16+1 files, 4+1 directories

Filename	Size	E...	Created	Modified	Access...	A...
$Boot	8.0 KB		10/23/2003 12:12:59	10/23/2003 12:12:59	10/23/20...	SH
$Bitmap	2.0 KB		10/23/2003 12:12:59	10/23/2003 12:12:59	10/23/20...	SH
$BadClus	0 bytes		10/23/2003 12:12:59	10/23/2003 12:12:59	10/23/20...	SH
$AttrDef	35.2 KB		10/23/2003 12:12:59	10/23/2003 12:12:59	10/23/20...	SH
file-r-1.dat	120 bytes	dat	10/23/2003 12:14:59	10/23/2003 02:01:08	10/23/20...	A
file-r-3.dat	0.5 KB	dat	10/23/2003 12:15:13	10/23/2003 12:15:39	10/23/20...	A
file-n-1.dat	2.0 KB	dat	10/23/2003 12:17:58	10/23/2003 02:04:14	10/23/20...	A
file-n-4.dat	2.0 KB	dat	10/23/2003 12:18:12	10/23/2003 01:57:40	10/23/20...	A
file-n-5.dat	1.3 KB	dat	10/23/2003 12:18:14	10/23/2003 12:18:50	10/23/20...	A
file-n-3.dat	2.5 KB	dat	10/23/2003 12:18:09	10/23/2003 12:20:38	10/23/20...	A
file-r-2.dat	120 bytes	dat	10/24/2003 11:27:35	10/23/2003 02:02:00	10/24/20...	A

```
  Offset     0  1  2  3  4  5  6  7   8  9 10 11 12 13 14 15
02769536    00 00 00 00 02 01 00 00  00 00 00 00 00 00 00 00
02769552    00 00 00 00 00 00 00 00  30 00 00 00 78 00 00 00
02769568    00 00 00 00 00 00 02 00  5A 00 00 00 18 00 01 00
02769584    05 00 00 00 00 00 05 00  00 6B 64 30 89 99 C3 01
02769600    00 6B 64 30 89 99 C3 01  00 6B 64 30 89 99 C3 01
02769616    00 6B 64 30 89 99 C3 01  00 00 00 00 00 00 00 00
02769632    00 00 00 00 00 00 00 00  20 00 00 00 00 00 00 00
02769648    0C 03 66 00 69 00 6C 00  65 00 2D 00 72 00 2D 00
02769664    31 00 2E 00 64 00 61 00  74 00 00 00 00 00 00 00
02769680    80 00 00 00 90 00 00 00  00 00 18 00 00 00 01 00
02769696    78 00 00 00 18 00 00 00  E9 EA 30 ED B0 86 CB C6
02769712    4C 3A 05 D3 74 E9 3D 18  91 8C A6 75 32 62 6F DB  L:.Óté=.´▌¦u2boÛ
02769728    F4 F3 18 B9 10 A7 B6 D9  17 8F 60 7C A2 51 FC A9  ôó.¹.§¶Ù.▌`|¢Qü©
```

Details
[C:\Projects\diskimages\3-ntfs
NTFS

State:	original
Undo level:	0
Undo reverses:	n/a
Total capacity:	7.8 MB
	8,224,768 bytes
Bytes per cluster:	512
Total clusters:	16,064
Bytes per sector:	512
Total no. of sectors:	16,064
Last scanned:	20 min. ago
Cluster No.:	5409
	$MFT (#27)
	?:\
Window #:	1
No. of windows:	1
Mode:	Text
Character set:	ANSI ASCII
Offsets:	decimal
Bytes per page:	22x16=352

Figure 13 Displaying the file name associated with MFT entry #27 using WinHex's context sensitive Access button which facilitates the recovery of the associated file.

useful for repairing damage and recovering data. A number of enhancements are proposed here to address shortcomings described in this paper.

Preservation

One proposed enhancement to the *Clone Disk* feature is error handling to inform the user when the destination disk has insufficient capacity. Alternately, prompting the user for additional storage when the destination drive is full would be an effective way to respond to a full disk. It would also be prudent to prevent a forensic image from being saved to the same disk that it is being copied from. One approach to preventing users from accidentally overwriting evidence is to make all disks read only by default and to require users to designate a disk as writable before WinHex will save data onto it.

Since the testing in this review was not exhaustive, the *Clone Disk* feature could benefit from more testing to determine if it has problems under certain circumstances. In any event, given the importance of the acquisition process, it is advisable to duplicate an evidentiary disk using at least two tools and to verify that the MD5 value of both images match that of the original.

Table 4 WinHex results for "FAT Keyword Search #1" (Carrier, 2003c)

Num	String	Cluster allocation	Sector
1	first	file1.dat	271
2	SECOND	file2.dat	272
	SECOND	Root directory	239
3	1cross1	file1.dat	271
4	2cross2	file3.dat	273
5	3cross3	unallocated space	283
6	1slack1	file2.dat	272
7	2slack2	file3.dat	274
8	3slack3	file4.dat	277
9	**1fragment1**	**Not found**	**Not found**
10	**2fragment sentence2**	**Not found**	**Not found**
11	deleted	Unallocated space	276
12	a?b\c*d$e#f[g^	file7.dat	279

Recovery

One issue uncovered while testing WinHex's file recovery capabilities on FAT file systems was that clusters allocated to active files were included in undeleted files. Since there are advantages to different recovery methods, it would benefit the user to have an option to choose which method to implement. The method not currently employed by WinHex is to skip allocated clusters during the undeletion process. Another proposed enhancement is an automated feature to recover alternate data streams.

Search

One proposed enhancement to WinHex's search functionality is to perform both logical and physical searches of forensic images. A logical search capability would enable WinHex to find keywords that are fragmented across non-contiguous clusters allocated to a file. A logical search would also enable WinHex to calculate the offset of the keyword from the beginning of the file.

When using the *Simultaneous Search* feature to export search results to a file or WinHex's Position Manager, it would be helpful to include the file name associated with resident data of MFT entries. When a keyword is found in ADS, the search results should provide the name of the stream in addition to that of the parent file. Additionally, a regular expression search feature would enable more flexible searches.

Documentation

Although WinHex creates a log file when certain actions are performed, it does not log all actions that a forensic examiner takes. A log of an examiner's actions can help others assess the work, reproduce the results, and determine if anyone viewed private data without authorization or exceeded their authorization by overstepping the bounds of an organization's privacy policy or a search warrant. When using tools that do not create this type of audit record, it is necessary to make detailed written notes. Although it is always a good practice to keep written notes when processing evidence, they rarely have the level of detail of computer generated audit logs.

Classification

Currently WinHex has limited data reduction capabilities that could be improved by classifying files, detecting file signature mismatches, and by using the NSRL hash database to identify known files. According to the developer, a WinHex Forensic Edition will be available later this year that can automatically classify files (e.g., Microsoft Office documents, e-mail, images, pictures with a high percentage of skin color) and detect file signature mismatches.

Acknowledgements

Thanks to Stefan Fleischmann for explaining some of the nuances of WinHex, to Brian Carrier for providing the tool testing images and helpful suggestions, and to Troy Larson for his invaluable feedback and enduring wit.

References

Boyd C, Forster P. Time and date issues in forensic computing—a case study. Digital Invest 2004;1(1).

Carrier B. Extended DOS partition test, 2003a. Available from: http://dftt.sourceforge.net/test1/index.html.

Carrier B. NTFS keyword search #1, 2003b. Available from: http://dftt.sourceforge.net/test3/index.html.

Carrier B. FAT keyword search #1, 2003c. Available from: http://dftt.sourceforge.net/test2/index.html.

Carrier B. FAT undelete test #1, 2004a. Available from: http://dftt.sourceforge.net/test6/index.html.

Carrier B. NTFS undelete (and leap year) test #1, 2004b. Available from: http://dftt.sourceforge.net/test7/index.html.

Casey E, Palmer G. The investigative process in Digital evidence and computer crime: forensic science, computers and the Internet. 2nd ed. Academic Press; 2004.

USDOD. National industrial security program operating manual (NISPOM / 5220.22-M), 1995. Available from: http://www.dss.mil/isec/nispom_0195.htm.