code{4}lib JOURNAL

Issue 34, 2016-10-25 ISSN 1940-5758

Digital Archaeology and/or Forensics: Working with Floppy Disks from the 1980s

While software originating from the domain of digital forensics has demonstrated utility for data recovery from contemporary storage media, it is not as effective for working with floppy disks from the 1980s. This paper details alternative strategies for recovering data from floppy disks employing software originating from the software preservation and retro computing communities. Imaging hardware, storage formats and processing workflows are also discussed.

John Durno, University of Victoria Libraries

Introduction

In this paper I will discuss tools and techniques I have employed to retrieve content on floppy disks from various 8- and 16-bit computer systems dating from the 1980s. In the context of several projects completed over the past two years I have recovered data from approximately 500 floppy disks. Disks from different sources exhibited many unique characteristics and required specialized processing to extract the content in a usable form.

Specifically, this paper describes the tools and techniques I have used to recover content saved to 5.25″ and 3.5″ floppy media from the following systems:

- IBM PC MS-DOS
- Kaypro II CP/M
- Mac OS 7
- Apple II ProDOS
- Apple II p-System & IBM PC p-System
- Atari 130 XE

Floppy disks were sourced by way of donations to our Archives, from materials in the Library collection, and from researchers seeking to access materials on early computing media from their own collections. The different requirements of these use cases also affected the processes employed to retrieve information. The researchers' interests were best served by simple file recovery, sometimes involving format conversion, while materials destined for archival preservation required a more structured and consistent process.

While my main reason for writing this paper is the hope that my methods may prove useful to others confronted with similar problems, these case studies are also intended to support a larger argument. As evidenced by disk imaging toolkits such as those found in Archivematica and BitCurator much of our current practice reflects the assumption that tools derived from the domain of digital forensics can productively be applied to all types of digital media, from 1980s floppy disks through contemporary hard drives. I would argue instead that floppy disks from the 1980s are a separate problem domain requiring specialized tools from outside the digital forensics community.

Relative to earlier eras, today we enjoy a high level of standardization in our computing environments and routinely expect digital content to be intelligible across multiple platforms. The major challenge is managing the sheer volume of data that is stored on modern systems. In the 1980s however these conditions were reversed. While the volumes of data were often miniscule by contemporary standards, a wide range of computing systems developed by competing vendors implemented proprietary technologies up and down the stack, from disk encodings to file systems to applications and their associated file formats. Far more than in our own time, digital information from the 1980s was deeply enmeshed in the particular platforms, operating systems and applications that created it. We should not therefore expect to approach the very small amounts of boutique data found on floppy disks with the same tools we use to wrangle the vast amounts of content on newer hard drives.[1]

The observation “Forensic techniques and tools will not eliminate the problems presented by older media, but they can make certain parts of the preservation process more efficient and more secure”[2] points in the direction my argument will take, but I would argue it does not sufficiently stress the distinction between forensic tools and forensic techniques. While forensic techniques are still, broadly speaking, applicable, in most cases it does not make sense to apply tools developed by the digital forensics community to the problem of data retrieval from floppy disks. Often it is not even possible. Tools developed by the software preservation and retro-computing communities are generally far more useful as the cases described below will illustrate.

Disk Imaging

The first step in any kind of data recovery from floppy media is to create a disk image, a sector by sector copy of the entire disk, including the parts of the disk not normally seen by users (boot sector, file table, sector numbering, data marked deleted but not yet overwritten). Any further manipulation of the disk contents (such as content extraction) is done from a copy of the disk image. This has the benefit of minimizing the handling of the original floppy media, and provides a baseline known good state to control for inadvertent changes that could be introduced by subsequent manipulation of the contents. Write blocking is critical in the imaging process in order to prevent inadvertent alterations to the source media. In this my approach differs not at all from standard digital forensics methodology.

In most cases, the right tools make imaging disks relatively straightforward. In addition to drives of the requisite size (typically 3.5″ and 5.25″), one also needs a controller (circuitry to control the drive) and appropriate software to run the imaging process. Historically, floppy drive circuitry was often built into computer motherboards but most modern motherboards have limited capability in this regard. Devices like the Kryoflux and FC5025 supply external USB-attached drive controllers, with sophisticated circuitry enabling them to bypass the limitations of older disk controllers (for example, they can read disks with both GCR and MFM encodings).

At the University of Victoria we have three devices capable of imaging floppy disks, each with their own strengths and weaknesses.

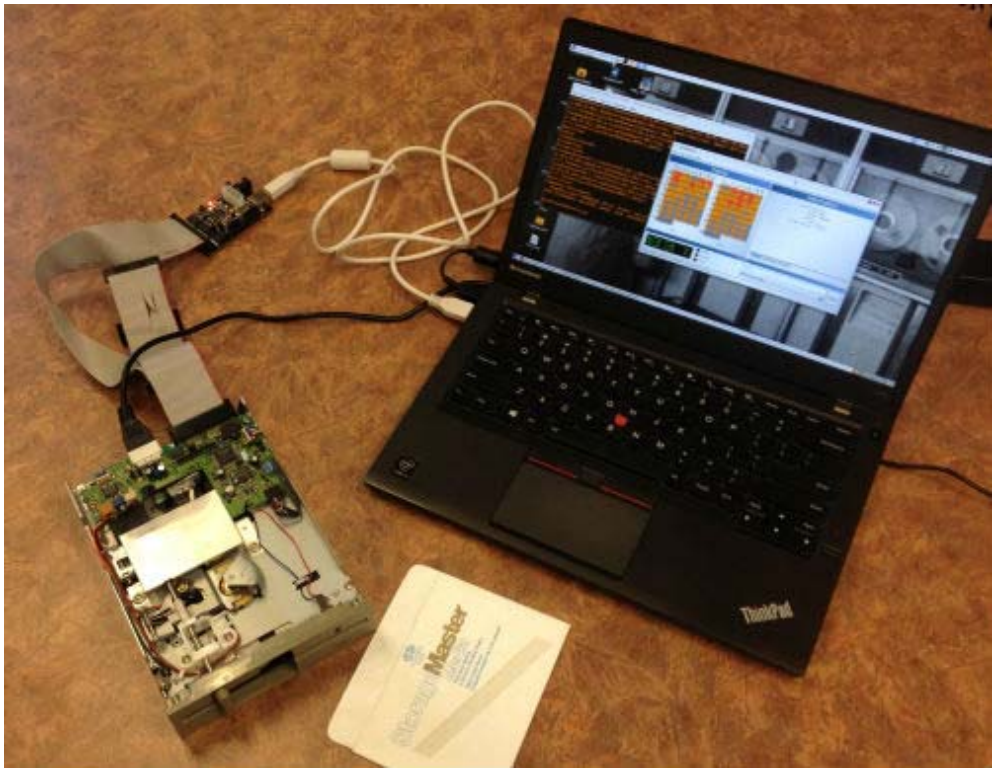


Figure 1. Kryoflux controller connected to 5.25′ Floppy Drive and Laptop (via USB)

1. Kryoflux. Without a doubt, the Kryoflux is the most versatile device we have for imaging floppy disks. It consists of a controller, cabling, and software. With appropriate drives it is capable of imaging both 3.5″ and 5.25″ disks and can handle a wide range of encodings and sector formats. It has hardware-level write-blocking. The software has both GUI and command line modes (the command line is harder to use but more flexible). The Kryoflux is also the most capable of our three devices for recovering data from damaged disks, and is the only device we have that is capable (with a modified drive) of recovering data from ‘flippy’ disks

(single-sided floppies with data written on both sides). However it is easy to damage and can only be handled by experienced technical staff.

2. Windows 7 workstation. This is a purpose built workstation with a DeviceSide Data FC 5025 USB floppy controller (for 5.25″ disks), a motherboard-attached 3.5″ floppy drive, and standard optical media drive. The FC 5025 controller has proven to be surprisingly capable given its low cost (relative to the Kryoflux) and can handle a wide range of disk formats. The 3.5″ floppy drive has been under-utilized but will work with any Windows imaging software; for example, FTK Imager. The FC5025 unit employs hardware level write blocking, while write blocking for other devices is enabled at the level of the Windows registry. This approach is also useful for acquiring content from non-floppy devices, such as external hard drives, CDs and thumb drives.

3. Industrial motherboard workstation. Another purpose-built workstation, this one has a 32 bit industrial motherboard and can boot into DOS, Windows XP, and Linux (CentOS). 3.5″, 5.25″ and optical drives are all directly attached to the motherboard. Of our three workstations we have used it the least for imaging, but it has come in handy for edge cases (see the section on PC p-System disks below). Imaging software includes standard open source tools like Guymager and dd on the Linux partition, the versatile OmniFlop[3] on the Windows XP partition, and Dave Dunfield's ImageDisk[4] in DOS. Write blocking is implemented at the application level and by using write protection features on the disks themselves.[5]

Given its versatility, the Kryoflux is the generally preferred option for floppy disk imaging. However, the other two devices have occasionally provided better results in accessing the contents of copy-protected disks and disks with unusual sector formats. And one should always be conscious that floppy drives can silently go out of alignment, resulting in disk images that cannot be read, or worse still, contain corrupted files. Having multiple drives and devices available makes it possible to periodically run sanity checks on your imaging output.[6]

Most imaging software comes with built-in settings for known disk types. However it is not always easy – particularly in the case of posthumous donations of archival materials – to determine what particular combination of hardware and operating system was used to write to the disk in the first place. In some cases the relevant information is noted on the disk label, but in others a certain amount of trial and error may be required.

Image formats

Tools developed by the digital forensics community typically save disk images as either AFF (Advanced Forensics Format) or EWF (Encase/Expert Witness Format). The main advantages of these formats are seen to be that they store metadata, including checksums, within the disk image file itself. That provides some assurance that the metadata is accurate and the image has not subsequently been tampered with.

Unfortunately, AFF up to version 3 has been deprecated by its creator[7] , and EWF is proprietary (albeit well-documented and with some open source support)[8] . For these reasons, and because neither format is well supported outside the forensics community, there is some cause to doubt their suitability as long-term preservation formats.

Sector image formats would appear to be more suitable for floppy disk preservation, for the following reasons:

- They are the standard output of imaging tools such as the Kryoflux and FC5025.
- They are widely supported by a range of open source content extraction utilities,
- Their contents are accessible under emulation, and
- They have been in existence longer than most preservation formats, which bodes well for their continued longevity.

The forensics community designs tools for criminal investigators, not for digital archaeologists. Tools that support AFF and EWF (Guymager, fiwalk and The Sleuth Kit for example), while undoubtedly valuable in their problem domain, were not designed to recover and preserve content from computer media from the 1980s. Most of the tools that can recover content from 1980s floppy disks either output or expect to work with standard sector image formats.

Sector images vs. stream files

Stream files are a proprietary but publicly-documented format output by the Kryoflux controller. They contain a great deal of information and are the easiest to acquire: they contain all the low-level information the Kryoflux generates while reading flux transitions, and can be made without knowing any technical details (encodings, sector geometries) about the source media beforehand. However, stream files are quite large in comparison to sector images (roughly 100 times larger, so a stream file for a 360K disk will come out to around 40 MB), and need to be converted to sector images in order to access their data in human-

parsable format. The Kryoflux software, DTC, can be used to create images from stream files. They are not a preservation format, as noted in the documentation: 'Stream files are hardware specific (to the KryoFlux device) and therefore are not intended for long term preservation.'[9]

It is worth noting that acquiring only stream files would be a workable strategy for obtaining floppy disk images in situations where it is necessary to use semi-skilled labour to do the imaging work. In such cases the work of converting streams to sector images could be deferred until later.

Sector images invert the characteristics of stream files: they are typically the same size as the original disk, accessible to emulators and content extraction tools, non-proprietary in most cases, and are not a complete recording of all the low level data the controller was able to read from the disk, such as flux reversals and index blocks.

Processing images for information storage and access

While creating disk images is relatively straightforward, making their contents intelligible to humans can be more challenging. Fortunately a wide range of readily available tools exist to facilitate the process. I will be describing some of them below.

Except where otherwise noted, all of the work was done on a modern Lenovo Thinkpad 450s running Ubuntu Linux 15.10. Most open source content extraction tools and emulators work well in Ubuntu. Many of these tools have a command line interface, which makes it possible to script the repetitive parts of the job.

Also, working in Linux provides added security against computer viruses. Floppies were an attack vector for viruses in their day and many of the viruses from 25 years ago are still perfectly capable of infecting a modern Windows computer. For this reason, files recovered from floppy disk images should be immediately scanned for viruses prior to further handling. I use the open source ClamAV[10] for this purpose. The images themselves should also be scanned to check for boot sector viruses.

While one could preserve disk images only and not extract their contents until such time as they were required, running the extraction process up front facilitates an examination of the image contents in order to ensure the imaging process was successful.

Structured output

As noted above, the desired end product of the content extraction process varies depending on the nature of the project. For digital preservation, one reasonable option would be creating a structured directory containing the disk image, the extracted contents of the disk image, and metadata files:

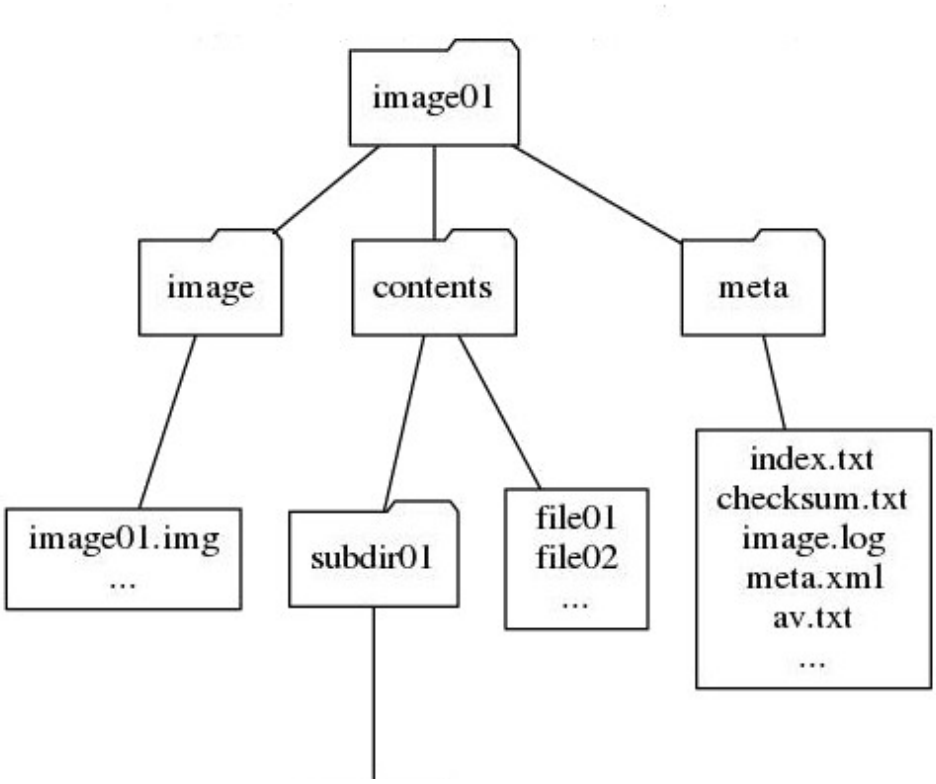


Figure 2. An example data structure for the image and output of a floppy disk

Non-exhaustively, the meta folder may contain:

- index.txt: A recursive directory listing of the contents of the disk image
- checksum.txt: An MD5 checksum of the disk image, generated immediately after the image was acquired
- image.log: A log of the image creation process generated by the Kryoflux or other imaging software, noting the condition of disk sectors
- meta.xml: Metadata recorded in accordance with archival policy, for example control numbers, date the image was made, text from the disk label, disk image format, tools used to extract the files and so on.
- av.txt: Output from the antivirus scanner

Time and resources permitting, photos of the disks could also be included in the meta folder. Floppy disk labels often contain useful metadata.

Case Study 1: PC/MS-DOS Floppy Disks (3.5″/5.25″)

Floppy disks created on PC/MS-DOS systems were not as prevalent in our processing queues as we originally anticipated. Due to their ubiquity in other contexts, these are the easiest kinds of disks to work with, and the only one of the case studies in this paper that digital forensics tools like guymager, The Sleuth Kit, and fiwalk can handle.

Disk imaging: All standard imaging tools come with settings for PC/MS-DOS disks. Because the Kryoflux works at the level of the disk encoding rather than the file system, the MFM setting (i4) is appropriate here. For double density disks, the step setting -k2 (indicating 40 cylinders) should be used, for example:
dtc -fimage.img -k2 -i4

File Extraction: Again, multiple options exist for copying individual files from the disk image to the local file system. I typically use the set of utilities collectively known as mtools, which enable Unix systems to work with MS-DOS file systems.

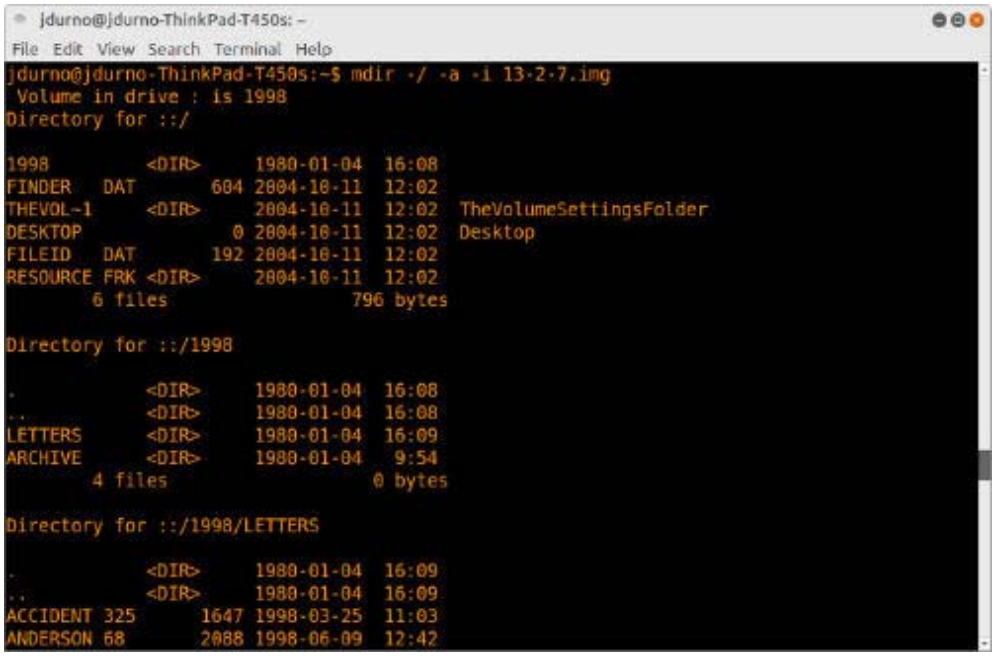


Figure 3. Partial output of `mdir` command

The following commands produce the desired output, for a disk image with the name “image.img”:

Recursive directory listing including hidden files, output to the file “index.txt”:

```
mdir -/ -a -i image.img > index.txt
```

Recursive content extraction output to subfolder “content”, preserving modification times and file attributes:

```
mcopy -p -s -m -i image.img ::* content
```

In a small number of cases where mdir and mcopy failed to read a disk image one may mount the image in the linux file system and extract the contents that way. Use:

```
sudo mount -t msdos -o ro,loop image.img mountpoint/
```

… where “mountpoint/” is the name of the empty directory where you want the image to mount. Standard unix utilities (ls, cp) can be used to work with the files once the image has been mounted.

Case Study 2: Kaypro II CP/M (5.25″)

Disk imaging: The Kryoflux MFM setting (i4) works for Kaypro II CP/M disks. The FC5025 floppy controller has a setting specifically for KayPro II CP/M disks.

File extraction: I used Michael Haardt’s cpmtools package[11] , which functions similarly to mtools, but for CP/M file systems. One significant difference is that the disk format must be specified, because the CP/M operating system ran on a variety of systems, not just PC-compatibles. In this example, the command line switch `-f kpii` (for Kaypro II) is added.

Disk formats are defined in a disk definitions file which, on Linux systems is usually located in /usr/local/share/diskdefs. The file contains human-readable text specifying floppy disk geometries (sector lengths, number of tracks, block sizes, and so on) employed by the many different different systems that ran CP/M. A comment at the beginning of most stanzas identifies the system to which it applies. For example, the disk definition for the Kaypro II is contained in the following stanza:

```
#Kaypro II
diskdef kpii
seclen 512
    tracks 40
      sectrk 10
      blocksize 1024
      maxdir 64
      skew 0
      boottrk 1
      os 2.2
end
```

Most versions of CP/M did not implement directories for file storage, so recursive output is not necessary. Instead, CP/M structured file storage around what were called user spaces, numbered areas where different users could store their files. As most CP/M home computers were single user systems usually there will only be one user area, numbered 0 indicating that files were accessible to all users.

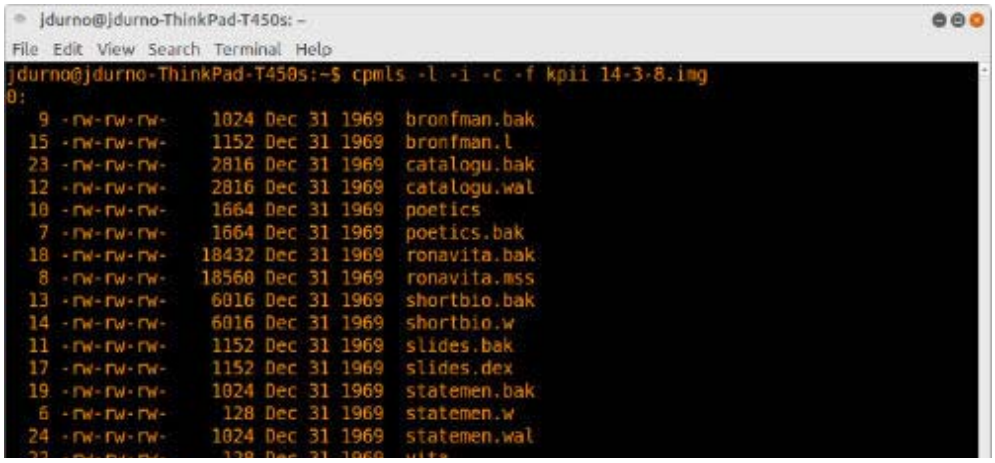


Figure 4. Output of `cpmls` command

For a directory listing a number of different output options are available. The "-l" flag produces Unix style output, with permissions, file modification times (not always trustworthy, as illustrated above) and file sizes.

```
cpmls -l -i -c -f kpii image.dsk > index.txt
```

This command extracts all the files on "image.dsk" in user area 0 to the subdirectory "content", preserving original time stamps.

```
cpmcp -p -f kpii image.dsk 0:* content/
```

Case Study 3: Mac OS, 3.5″ Double sided, double density (800K)

Mac floppy disks from the mid-1980s through the mid-1990s differed from PC disks in two significant ways:

1. The file system, HFS (Hierarchical File System)
2. The disk encoding scheme, Apple GCR

There are a couple of exceptions: the earliest Mac 400K floppies had a different file system (MFS, for "Macintosh File System") and the later high density floppies used MFM encoding, not GCR, for PC compatibility. Another aspect which may cause confusion is Apple's successor to HFS, developed for OS X, called HFS Plus (or sometimes, incorrectly, "HFS Extended") and sometimes referred to as HFS. Software like The Sleuth Kit that supports the later variant of HFS do not necessarily support the earlier one.[12]

These complications notwithstanding, if you need to work with double density 3.5″ Mac floppy disks you will require tools capable of handling GCR encoding and HFS file systems.[13]

Disk imaging

The Kryoflux can image 3.5″ double density disks using the "Apple DOS 400/800K sector image" setting. As PC disk controllers cannot read GCR encodings and the FC5025 can only handle 5.25″ disks, the Kryoflux is the only option for imaging Mac 3.5″ disks up to 800K unless you have a vintage Mac with a working floppy drive.

Because the later Mac HD (1.44K) floppies were MFM-encoded, there is a much broader range of options for obtaining Mac HD floppy disk images. Most PC floppy disk controllers can handle them, and most PC-compatible disk imaging software (eg. Omniflop, FTKImager, or Guymager in Linux) includes the appropriate settings.

File extraction

My toolkit of choice for working with HFS file systems is hfsutils, a collection of utilities for Unix systems that has been in development since 1996[14] . It works similarly to mtools and cpmtools, except that the disk image needs to be mounted before it can be read. The following command mounts an HFS disk image named "image.dmg":

```
hmount image.dmg
```

If the command was successful, hmount will return some information about the disk including the volume name, time created and last modified, and number of bytes free. One can capture that output by modifying the above command to:

```
hmount image.dmg > hmount.txt
```

To recursively list directory contents, including hidden files and catalog ids, in long format, output to the file "index.txt", use:

```
hls -i -a -l -R > index.txt
```

Note that the disk image file does not need to be specified, because it was mounted in the previous step.

The HFS file system includes resource forks and data forks. Every file in the file system may have both. Data forks contain unstructured data, while resource forks include structured data relating to the file, including icon bitmaps, positioning of windows, and so on. While much of this information is irrelevant to viewing standard file types on modern systems, data in the resource fork sometimes extends beyond system-specific metadata including, for example, the embedded images in a word processing document.[15]

While it is therefore important to preserve resource forks, it is debatable whether resource forks need to be included among the files extracted from the disk image as a routine procedure. If the goal is to read the extracted files on non-Mac systems, then resource forks are unnecessary as other systems cannot use them. If the goal is to read the files on a Mac, the best approach is probably to mount the disk image and access the files that way.

To extract files without the resource fork, to a subdirectory named ‘content’, use:

```
1 hcopy -r :* content/
```

To extract files with resource forks, use:

```
1 hcopy -m :* content/
```

After working with a disk image, one must unmount it before mounting another one. The following command unmounts the disk:

```
1 humount
```

Case Study 4: Apple II

The Apple II series was in production from 1977 through 1993, during which time between five and six million were sold.[16] Apple II hardware and software evolved significantly during that period, and even models of the same era could boot into different operating systems. No single approach works for all variants when it comes to processing Apple II disks. In this section I discuss two variants that I have come across.[17]

Apple II ProDOS (5.25″)

ProDOS was first introduced in late 1983 to overcome a number of shortcomings associated with the previous Apple DOS version, 3.3. It was eventually renamed ‘ProDOS 8’ after a 16 bit version was released.

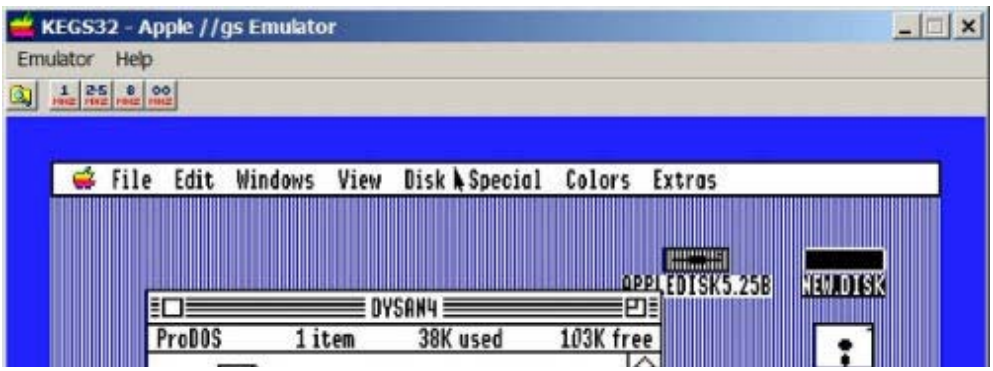
Disk imaging

Apple DOS 3.3 and ProDOS use the same 16 sector format for 140K disks, so the Kryoflux setting for “DSK, DOS 3.3 Interleave” was appropriate here. The FC5025 has an “Apple ProDOS” setting. The Kryoflux command line requires the following parameters for an optimal read of Apple DOS 3.3 disks:

```
1 dtc -f<imagename> -x0 -i8 -l8 -dd1p
```

File extraction

As HFS was specific to Macs, hfsutils cannot be used to extract content from Apple II disks, which had their own file system. Initially I mounted a few sample disk images in the KEGS Apple IIgs emulator to examine their contents, which as it turned out consisted primarily of files written in an early version of Appleworks.[18] While it was possible to read the Appleworks files in the emulator, the process was awkward and slow. The sheer volume of content (hundreds of files on more than 90 floppy disk images) would have rendered that approach problematic for anyone wishing to read more than a handful of documents.



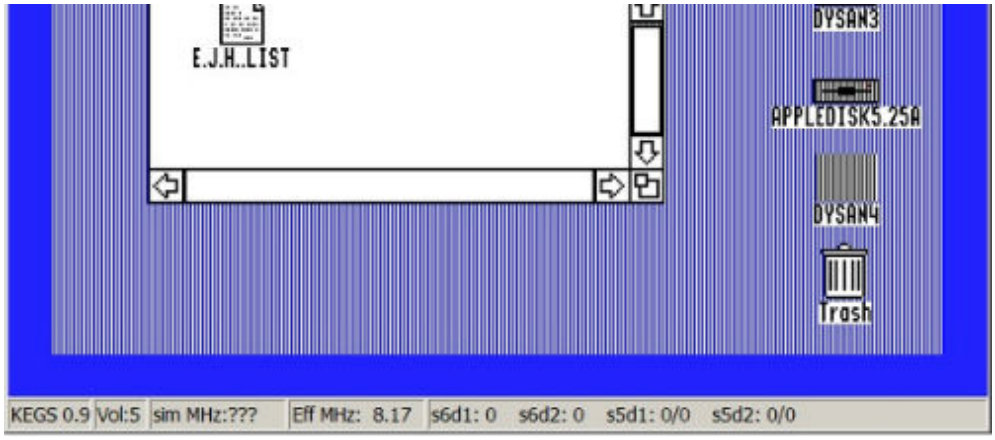


Figure 5. Viewing files in the KEGS Apple IIgs Emulator

Instead I used AppleCommander to extract and convert the contents of the disk images[19] . AppleCommander is an open source java program with both GUI and command line interfaces. In addition to ProDOS it can handle a range of common Apple II disk image formats, versions of DOS 3.3, Apple Pascal, and CP/M among them.

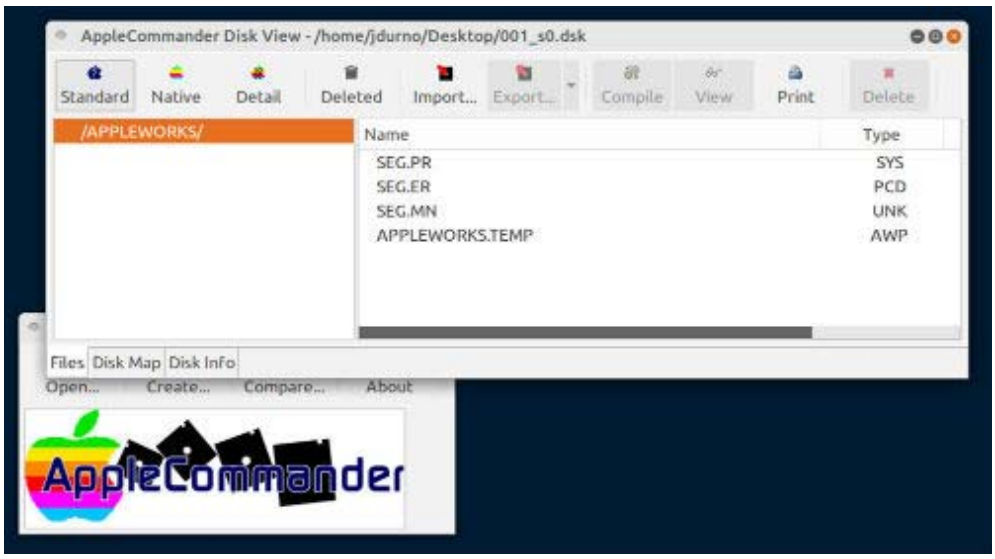


Figure 6. AppleCommander GUI

To write a directory listing into 'index.txt', use:

```
1 java -jar ac.jar -ll disk.dsk > index.txt
```

… where "ac.jar" is the name of the Apple Commander jar file and "disk.dsk" is the name of the disk image you want the directory listing from. The switch "-ll" outputs a long file listing with file name, creation and modification dates, and file sizes.

The current stable release version of AppleCommander (1.3.5) does not have the ability to extract all the files from a disk image via the command line interface, but the interim release candidate (1.3.5.13) does, using the command:

```
1 java -jar ac.jar -x disk.dsk content
```

This command extracts all the files on the disk image to the subdirectory "content." It automatically applies a conversion filter to certain types of documents; for example AppleWorks documents are extracted as text files. While text conversion was acceptable for our project, it would be problematic in cases where it was important to retain formatting. In a curious reversal of the norm, the AppleCommander GUI provides a greater range of bulk extraction options than the command line, including text, RTF, HTML, and no conversion (binary extraction).

Case Study 5: Apple II p-System & IBM PC p-System 5.25″

The p-System OS was developed in the late 1970s at the University of California, San Diego. Like CP/M, was designed to be portable across different varieties of hardware, including the Apple II, the PDP 11, and later the IBM PC. Two variants of p-System disks (Apple II and IBM PC) were donated to our archives, corresponding to two phases of a project undertaken by our Computer Science department collaborating with the Victoria-based artist Glenn Howarth in the first half of the 1980s.

Disk imaging

p-System disks used the encoding scheme appropriate for the hardware on which they were running, MFM on PC and GCR on Apple.

For imaging the Apple disks, the built-in “DSK, DOS 3.3 Interleave” Kryoflux setting produced disk images readable by the file extraction utilities described below.

Imaging the PC disks proved to be more of a challenge. The PC version of the p-System OS was a rarity in its day. As a consequence of its relative obscurity, there are no built-in settings for it in any of my imaging tools. For example, it is not included in the list of ~150 formats supported by Omniflop, nor is it a known format that is supported by the FC5025 and Kryoflux controllers. The Kryoflux was, of course, able to obtain stream files but my attempts to produce sector images resulted in files that were oddly twice the size they should have been (720K images for a 360K floppy).

Further analysis of these disks using a program called Anadisk indicated the sector numbering was unusual. Following a recommendation on the Kryoflux forums I reimaged the disks using a DOS utility called ImageDisk and converted the resulting IMD files to IMG.[20] These worked somewhat better, allowing me to extract some, but not all, of the files on the disks. I believe the remaining problems have to do with how the UCSD p-System handled sector interleaving, but have not yet confirmed this.

File extraction

As with other test cases, specialized tools were required to process the p-System disks. In this case, the same tools and settings could be used for both the Apple II and IBM PC disk images. I used Peter Miller’s ucsd-psystem-fs (http://ucsd-psystem-fs.sourceforge.net/), a set of utilities for manipulating UCSD p-System disk images.

For both PC and Apple disk images, the following command outputs a directory listing from the disk image “disk.img” into the text file “index.txt”:

```
1 ucsdpsys_disk -f disk.img -l > index.txt
```

The following command extracts the files on the disk image “disk.img” into the subfolder “contents”:

```
1 ucsdpsys_disk -f disk.img -g contents/
```

Case Study 6: Atari 130XE 5.25″

Atari manufactured their line of 8-bit home computers from 1979 through 1992. The mass-market 130XE model dates from the latter half of that period. Atari systems were popular for gaming but were also used for office applications. In this case the disks came from the personal collection of a researcher, not from our Archives. The researcher was able to tell us both the model of computer and the software (AtariWriter) that was used to create the files on the disks.

Disk imaging

Early Atari floppy disks were typically FM encoded, single sided, and had 40 tracks with 18 sectors each. Later disk drives used MFM encoding and additional tracks per sector, permitting increased storage density. The Kryoflux has settings for both types. The “MFM XFD, Atari 8-bit” GUI setting worked for these disks.

File extraction

I was unable to locate command line tools to work with Atari disk images. Instead, we configured an instance of the Atari800 emulator, and used AtariWriter to read the files from the disk images mounted as virtual floppies.[21] It proved possible to send files as postscript from AtariWriter to a print queue on the host system. In this case the emulator was installed on the researcher’s OS X laptop and configured to print files to PDF. This would not be an optimal technique for converting hundreds of files as it involves touching each file individually, but it addressed our requirements in this particular case.

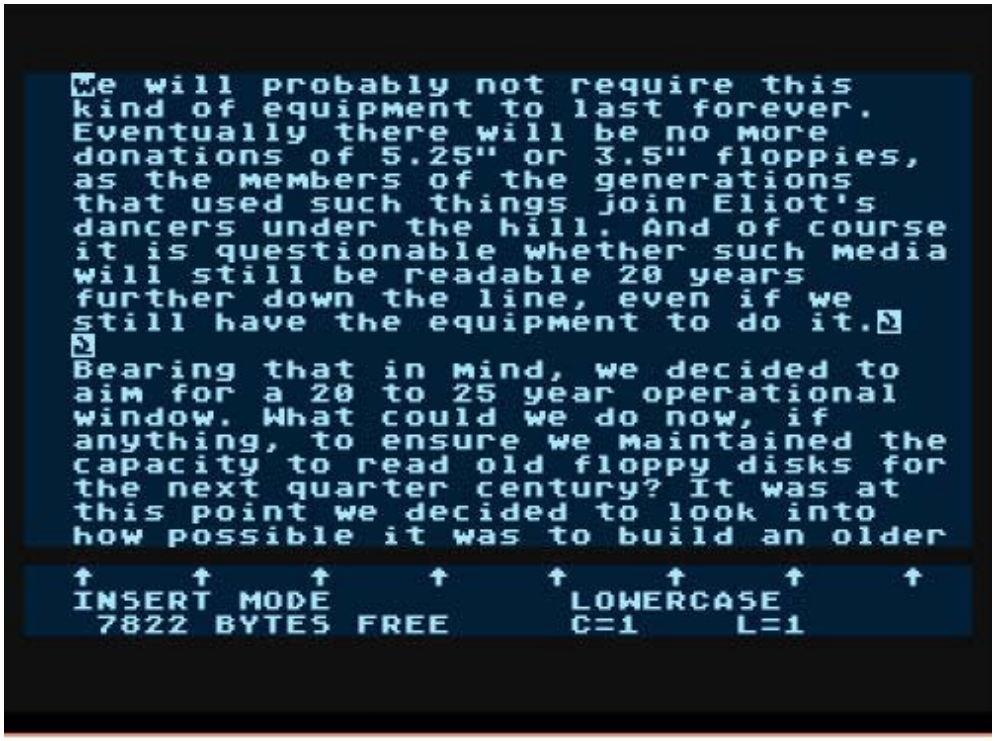


Figure 7. AtariWriter document rendered in Atari800 emulator

Automating the process

The work of imaging and processing floppy disks typically falls into two distinct phases. In the first phase, the disks are examined in order to determine the optimal imaging and content extraction strategies. Once the technical analysis phase is over, however, the work quickly becomes rote: issuing the same commands over and over again as the individual disks are processed. Further analysis and decision making is typically only required in the case of damaged disks or in the event anomalous disks are encountered.

It is therefore advantageous to script as much of the work as possible. To automate the workflow described above, I have written scripts that reduce the effort required per disk to two brief commands, one to create the image and associated log file; and the other to create the directory structure (as shown in 'Structured output' above), move the disk image and log to their correct locations, checksum the disk image, extract the contents and directory listing, and run the antivirus scan and log its output to the correct location.

The sections above outline the basic building blocks of these scripts. Examples of my CP/M and DOS content extraction scripts are found in Github[22] . They are not in any way sophisticated examples of the art of programming; the good news is they don't need to be.

Rendering files on modern systems

The topic of rendering obsolete file formats on modern systems is vast and the challenges are many. The following are general observations from my own experience.

Speaking very broadly, there are two ways to go about accessing files with formats that are not supported on modern systems. One may convert the obsolete formats to their contemporary equivalents, or else recreate a period software environment either on old hardware or more usually via emulation. These approaches are not mutually exclusive; in practice they may be combined as in the Atari example above.[23]

One of the common characteristics I have observed across multiple sets of floppies is that disks from the same source typically exhibit little variety in terms of file formats. Unlike the computers of our own time with their plethora of different applications and associated file formats, personal computers of the 1980s were typically used for a much smaller range of tasks and ran a much smaller range of productivity software. In my experience, it is not uncommon for dozens of disks from the same source to contain nothing but files from a single word processing application.

Here again, the tools developed for enumerating file types in the context of a forensic investigation are of little use. They typically have problems identifying file types from 1980s systems, even common formats like Wordstar and AppleWorks. There is also less utility in enumerating file types in a set of floppy disks that contain only one or two types of files.

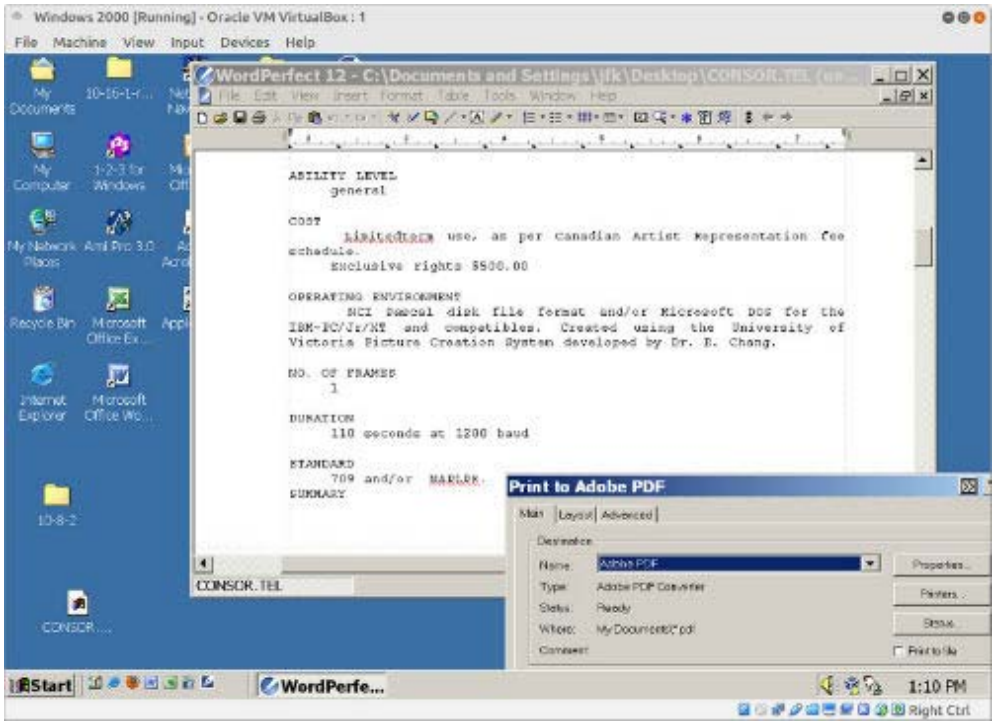


Figure 8. Converting Wordstar to PDF via WordPerfect 12 running on Windows 2000 in Virtualbox

Bulk format migration tools are also problematic. In many cases they don't exist for the formats in question or are expensive and proprietary. This is particularly troublesome in the case of Wordstar files, since Wordstar was widely used on multiple computing platforms in the first half of the 80s. Our current strategy for converting Wordstar documents to PDF involves opening them in WordPerfect 12 installed in a virtualized (Virtual Box) instance of Windows 2000 and rendering them as PDF files via Adobe Acrobat Professional 8. WordPerfect 12 has Wordstar conversion capabilities back to version 3.3. This approach respects the formatting of the original document but is time-consuming as documents must be converted individually by hand. It is far simpler and faster to script a bulk conversion from Wordstar to plain text (8-bit ASCII) if that will suffice.[24]

Damaged/Problem disks

In general, the 5.25′ floppy disks I have encountered have held up well over time, calling into question some of the more alarmist claims about the rate of deterioration of data on floppy media. I have seen numerous examples of disks stored in very average conditions that were readable 30+ years after they were first created. The exceptions have largely been disks containing application files, presumably because these were used heavily, whereas disks meant primarily for saving data would have been used less frequently. The relative proportion of damaged 3.5′ floppies has been higher, despite being on average 10 years newer than their 5.25′ predecessors.

The disposition of damaged disks requires careful consideration, not least because disks may be damaged in multiple ways:

1. The disk itself may be damaged beyond the point where it can be imaged
2. The protective case may be damaged or bent, but the disk inside might still be viable
3. The disk can be imaged, but with errors indicating bad sectors, leaving the resulting image in one of two states:
    1. It is possible to extract files from the disk image, but some of the contents might be corrupted
    2. It is not possible to extract files using the standard tools
4. The disk might be copy-protected, have an unknown sector format (as with the PC p-System disks mentioned above), or be a hard sector disk

With enough effort many of these problems can be overcome or at least mitigated. However doing so requires expertise that might not be readily at hand, and time, which is always in short supply. For example, if the protective case is damaged one may try cutting it open and carefully transferring the disk inside to an undamaged case. If the disk is copy protected or has an unknown sector format, there are tools that can be used to analyze its idiosyncrasies and help determine a solution. But, if the data on the disks is of uncertain value, it may not be worthwhile to expend the time necessary to address these problems, and in some cases problems will persist no matter how much effort is expended. Here it becomes a case of preserving what one can.

If a disk is truly unreadable, the only question is whether to preserve the physical media against the day when another attempt might be made. If the disk can be imaged but with bad sectors, then it seems reasonable to preserve that image along with a log file indicating where damage was encountered. If it is possible to extract files, that should also be done, however the possibility of damage should be clearly indicated in the metadata accompanying the image and extracted files.

Conclusion: Forensics or Archaeology?

This paper was originally envisioned as a technical note exploring the disk image ingestion capabilities of our library's digital preservation storage system, Archivematica. However during the early stages of research it became apparent that Archivematica could only perform a bit-level ingestion of the floppy disk images I had been acquiring in the course of my data recovery projects. Also, even had it been able to extract individual files, Archivematica did not recognize common 1980s formats like Wordstar and Appleworks, so was limited in its capacity to migrate their contents to archival formats.[25]

Archivematica relies heavily on open source tools developed by the digital forensics community. As noted above, these tools (for example, The Sleuth Kit and fiwalk) are not designed to handle the peculiarities of 1980s floppy disk image formats. The comparative recency of the emergence of digital forensics as a professional discipline seems a likely explanation. Most digital forensics software in common use dates from the late-1990s onward and reflects a disciplinary focus on retrieving data from contemporary storage media, not media long considered obsolete.

Archivematica's disk image ingestion capabilities appear to have been optimized to work with images created by BitCurator[26] , or more accurately with its forensic imaging software Guymager. Guymager is limited to imaging within the constraints of the disk controller on the physical hardware of the underlying system, which in a typical configuration would be a PC controller. This means it lacks a number of the specialized abilities of the Kryoflux mentioned in the section "Disk Imaging" above, making it a less than optimal choice for floppy disk imaging.

It should be noted that in addition to the standard suite of forensics tools, BitCurator includes a number of tools that originated outside the digital forensics community. Specifically, it includes mtools and hfsutils, both mentioned above, and HFS Explorer for accessing the contents of older Mac floppies. It would be straightforward to further expand BitCurator to encompass most of the other software discussed in this paper, and this could in fact be done by the end user.

Building similar content handling capabilities into Archivematica would be considerably more effort given the more tightly coupled nature of its tool chain, and it is possible to question whether it would be worth the effort. In my experience, recovering content from floppy disks involves enough variables to sufficiently confound any attempt to fully automate the process.

As the case studies above illustrate, employing a variety of specialized tools represents a more viable approach to content recovery from 1980s floppy disks than relying on the software found in archival toolkits. Given the long involvement of the retro-computing and software preservation communities with this problem domain, it is not surprising that the most effective tools originate from those communities as distinct from the realm of digital forensics.

About the Author

John Durno is Head of Library Systems at the University of Victoria (UVic), where he manages the team responsible for building and maintaining the Libraries' computing environment. For the past two years he been working with UVic Archives and Special Collections to develop preservation strategies for at risk digital materials on volatile media. Prior to joining UVic in 2006 John coordinated province-wide library technology projects for the British Columbia Electronic Library Network.

Footnotes

[1] Limitations of digital forensic software have been noted within the forensics community itself. See Beebe, N. Digital Forensic Research: The Good, The Bad and the Unaddressed. Advances in Digital Forensics V, IFIP AICT 306. 2009. doi:10.1007/978-3-642-04155-6_2: 'One of the successes identified in the previous section was the collective ability to archaeologically identify, excavate and examine digital artifacts. The problem, however, is that knowledge and expertise are

heavily biased toward Windows, and to a lesser extent, standard Linux distributions. The FAT12/16/32, NTFS and EXT2/3 file systems, the operating systems that implement them (Windows 9X/ME/NT/XP/Vista and various Linux distributions), and common user applications installed on them (Microsoft Internet Explorer and Outlook, Mozilla Firefox and Thunderbird, etc.) have been well studied. Researchers have paid insufficient attention to other operating systems, file systems and user applications, especially in light of current market trends.'

[2] In Kirschenbaum M, Ovenden R, Redwine G. 2010. Digital Forensics and Born-Digital Content in Cultural Heritage Collections. Available from https://www.clir.org/pubs/reports/pub149/pub149.pdf

[3] Omniflop disk imaging tool available from: http://www.shlock.co.uk/Utils/OmniFlop/

[4] Dunfield, D. Disk Image/Software Image Archive. Available from: http://www.classiccmp.org/dunfield/img/index.htm

[5] For a full description of this workstation and its construction, see Durno, J, & Trofimchuk, J. 2015. Digital forensics on a shoestring: a case study from the University of Victoria. code{4}lib journal 27. Available from: http://journal.code4lib.org/articles/10279

[6] See, for example, Formatted HD DOS floppies reading as unformatted. 2016. Kryoflux Support Forums. Available from: http://forum.kryoflux.com/viewtopic.php?f=3&t=1166

[7] AFF format deprecated. Available from: https://sourceforge.net/p/guymager/wiki/AFF%20format%20deprecated/

[8] Forensics Wiki. Encase image file format. Available from: http://www.forensicswiki.org/wiki/Encase_image_file_format

[9] Louis-Guérin, J. 2013. Kryoflux Stream File Documentation. Revision 1.1. Available from: http://www.kryoflux.com/download/kryoflux_stream_protocol_rev1.1.pdf

[10] ClamAV. Available from https://www.clamav.net

[11] Haardt, M. Cpmtools 2.0. Available from: http://www.moria.de/~michael/cpmtools/

[12] SleuthKitWiki. HFS. Available from: http://wiki.sleuthkit.org/index.php?title=HFS

[13] For further discussion of the challenges of working with early Mac floppies, see Purity, S. Working with Macintosh floppy disks in the new millenium. Available from: http://siber-sonic.com/mac/newmillfloppy.html

[14] Leslie, R. HFS Utilities. Available from: http://www.mars.org/home/rob/proj/hfs/

[15] Wikipedia. Resource fork. Available from https://en.wikipedia.org/wiki/Resource_fork

[16] 'Apple II' series, available from https://en.wikipedia.org/wiki/Apple_II_series

[17] Information about the evolution of Apple II operating systems is widely available, see for example 'DOS 3.3, ProDOS & Beyond' available from http://apple2history.org/history/ah15/

[18] KEGS Apple IIgs emulator. Available from: http://kegs.sourceforge.net

[19] Green, R. AppleCommander. Available from http://applecommander.sourceforge.net/

[20] For more information on this topic, see UCSD p-System disks (PC version). 2016. Kryoflux Support Forums. Available from http://forum.kryoflux.com/viewtopic.php?f=3&t=1153

[21] Atari800. Available from http://atari800.sourceforge.net/

[22] Sample CP/M and DOS ingestion scripts, see https://github.com/jdurno/floppy-utils

[23] For an excellent discussion of the challenges and opportunities afforded by emulation as an access strategy, see Dietrich D, Kim J, McKeehan M, & Rhonemus A. 2016. How to Party Like it's 1999: Emulation for Everyone. code{4}lib journal 32. Available from: http://journal.code4lib.org/articles/11386

[24] The basic priniciple is described in: Just solve the file format problem. WordStar. Available from http://fileformats.archiveteam.org/wiki/Wordstar. . Also see my implementation of the principle at: https://github.com/jdurno/floppy-utils/blob/master/wsconv

[25] While Wordstar is listed in the PRONOM registry that Archivematica relies on for file format identification, its ability to recognize the Wordstar format appears to be based on the file extension. Unfortunately, appending file extensions to identify file

types was not as common in the 1980s: while I have encountered hundreds of Wordstar files in my work to date I have never seen one with a .ws, .ws3 or equivalent extension.

[26] Reference “Wherever possible, use BitCurator packages for forensics tools”. Archivematica Wiki. https://wiki.archivematica.org/Digital_forensics_image_ingest

Works Cited

Atari800. Available from http://atari800.sourceforge.net/

Beebe, N. 2009. Digital Forensic Research: The Good, The Bad and the Unaddressed. Advances in Digital Forensics V, IFIP AICT 306. doi:10.1007/978-3-642-04155-6_2

Chessman S. 1996. dd. Linux Journal 32. Available from: http://www.linuxjournal.com/article/1320

Diamond E. 1994. The Archivist as Forensic Scientist – Seeing ourselves in a different way. Archivaria 38. Available from: http://journals.sfu.ca/archivar/index.php/archivaria/article/view/12031/1300

Dietrich D, Kim J, McKeehan M, & Rhonemus A. 2016. How to Party Like it’s 1999: Emulation for Everyone. code{4}lib journal 32. Available from: http://journal.code4lib.org/articles/11386

Durno J. 2016. Floppy-utils. Available from: https://github.com/jdurno/floppy-utils

Durno J, Trofimchuk J. 2015. Digital forensics on a shoestring: a case study from the University of Victoria. code{4}lib journal 27. Available from: http://journal.code4lib.org/articles/10279

Forensics Wiki. Encase image file format. Available from: http://www.forensicswiki.org/wiki/Encase_image_file_format

Green, R. AppleCommander. Available from http://applecommander.sourceforge.net/

Guymager Wiki. AFF format deprecated. Available from: https://sourceforge.net/p/guymager/wiki/AFF%20format%20deprecated/

Haardt, M. Cpmtools 2.0. Available from: http://www.moria.de/~michael/cpmtools/

Just solve the file format problem. WordStar. Available from http://fileformats.archiveteam.org/wiki/Wordstar

Kirschenbaum M, Ovenden R, Redwine G. 2010. Digital Forensics and Born-Digital Content in Cultural Heritage Collections. Available from https://www.clir.org/pubs/reports/pub149/pub149.pdf

KEGS Apple IIgs emulator. Available from: http://kegs.sourceforge.net

Kryoflux Support Forums. 2016. Formatted HD DOS floppies reading as unformatted. Available from: http://forum.kryoflux.com/viewtopic.php?f=3&t=1166

Kryoflux Support Forums. 2016. UCSD p-System disks (PC version). Available from: http://forum.kryoflux.com/viewtopic.php?f=3&t=1153

Leslie, R. HFS Utilities. Available from: http://www.mars.org/home/rob/proj/hfs/

Louis-Guérin, J. 2013. Kryoflux Stream File Documentation. Revision 1.1. Available from: http://www.kryoflux.com/download/kryoflux_stream_protocol_rev1.1.pdf

Pollitt M. 2010. A History of Digital Forensics. Advances in Digital Forensics VI, IFIP AICT 337

Purity, S. Working with Macintosh floppy disks in the new millenium. Available from: http://siber-sonic.com/mac/newmillfloppy.html

Ross S, Gow A. 1999. Digital Archaeology: Rescuing Neglected and Damaged Data Resources. Available from http://www.ukoln.ac.uk/services/elib/papers/supporting/pdf/p2.pdf

SleuthKitWiki. HFS. Available from: http://wiki.sleuthkit.org/index.php?title=HFS

Weyhrich, Steven. DOS 3.3, ProDOS & Beyond. Available from: http://apple2history.org/history/ah15/

Wikipedia. Resource fork. Available from https://en.wikipedia.org/wiki/Resource_fork

Wolverton, M. 2016. Digital Forensics in the Library. Nature. Vol 534: 139-140. Available from: http://www.nature.com/polopoly_fs/1.19998!/menu/main/topColumns/topLeftColumn/pdf/534139a.pdf