Practical Floppy Disk Recovery Study

Digital Archeology on BTOS/CTOS Formatted Media

Dirk von Suchodoletz, Richard Schneider
University Computer Center
Freiburg, Germany

Euan Cochrane
Archives New Zealand
Department of Internal Affairs
Wellington, New Zealand

David Schmidt
RetroFloppy
North Carolina, USA

ABSTRACT

This paper provides a practical example of digital archeology and forensics to recover data from floppy disks originally used by CTOS, now an obsolete computer operating system. The various floppy disks were created during the period between the mid 1980s to the mid 1990s containing different types of text, data and binary files. This paper presents practical steps from two different approaches, the tools and workflows involved which can help archivists and digital preservation practitioners recover data from outdated systems and media. While the floppy disk data recovery was a full success, issues remain in filetype detection and interpretation of non-ASCII data files of unknown or unsupported types.

1. INTRODUCTION

Archives New Zealand and the University of Freiburg cooperated on a data recovery project in 2011 and 2012. The archive received a set of 66 5.25 inch floppy disks from the early 1990s that contained records of a public organization dating back to the mid 1980s. These floppies were not readable using any standard DOS-based personal computer with a 5.25 inch floppy drive attached to it. There was very little information available in the beginning about the contents or technical structure of the floppy disks from the organisation that owned them. Because of the age of the disks and this lack of information about their contents the organisation was eager to retrieve all files that could be read from the disks and get all available information from those files. This is an ideal use case for digital archeology workflows using forensic methods [4, 2, 1] as archives may receive objects quite some time after they have been created (20 years or more later). To be able to recover raw bit streams from obsolete floppies, the archive purchased a special hardware device with the ability to make digital images of the floppy disks. The team from Archives NZ and the University of Freiburg was joined later by a digital archivist from RetroFloppy who had been working on a similar challenge from the same system after he discovered the discussion about their work on the Open Planets Foundation (OPF) blog.[1]

2. STUDY ON DATA RECOVERY

The digital continuity team at Archives NZ thought it would be a great opportunity to demonstrate the practical use of the KryoFlux device, a generic floppy disk controller for a range of original floppy drives offering a USB interface to be connected to a modern computer. In addition, more information on the work required to incorporate it into archival processes was to be gathered and documented.

2.1 First Step – Bit Stream Recovery

The first step in the process after receiving the physical media was to visually examine the disks to find out any technical metadata that was available. The disks had labels that identified them as DS QD 96 tpi disks, which refers to Double Sided, Quad Density, with 96 tracks per inch. A 5.25 inch drive was attached to the KryoFlux which itself was connected to a modern Windows PC using a USB connection. The KryoFlux works by reading the state of the magnetic flux on the disk and writing that signal into a file on the host computer. Different output options are possible: A proprietary KryoFlux stream image formatted file, a RAW formatted file, and an MFM sector (BTOS/CTOS) formatted image file were all created from the disks.

A major component besides the hardware device is the interpretation software to translate the recorded signal into image files that are structured according to various floppy disk-formatting standards. After recovering a few disks it became clear that they were not following any known filesystem standard supported by today's operating systems. Thus it was impossible to directly mount them into the host filesystem and read the files from them. But nevertheless it was possible to analyse the images with a hexidecimal editor. Visual inspection of the resulting data showed that the reading process was producing some meaningful data. Several "words" like *sysImage.sys* were repeated in all readable disk images, thus seeming to represent some structural filesystem data. By searching the internet for this string and others it was possible to deduce that the disks were likely created on a computer running the Burroughs Technologies Operating System (BTOS) or its successor the Convergent Technologies Operating System (CTOS) [5]. Fortunately more in-depth information could still be found on various sites

[1]See the discussion on *http://openplanetsfoundation.org/blogs/2012-03-14-update-digital-archaeology-and-forensics*.

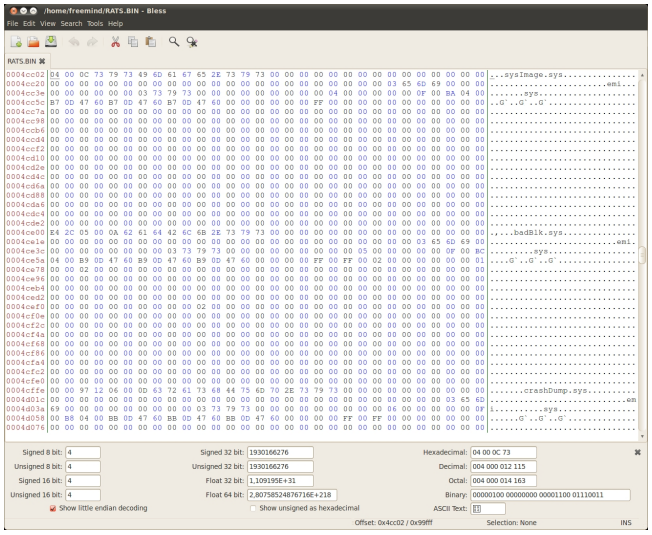

Figure 1: Hexdump image analysis revealing some hints about the original platform

describing the file system. After more research it was concluded that there is currently no software available to properly interpret disks or disk images formatted with this file system aside from the original software and its (obsolete) successors. As there are no emulators available for this system, an emulation approach was not a viable option either. At this point the image investigation was handed over to the computer science department of the Freiburg University to dig into the problem. An application was written to interpret the file system on the disks using the information available on the internet.

2.2 Second Step: Directory Reader

The preservation working group in Freiburg was able to attract a bachelor student for the task to write an interpreter and file extractor for the image files. This is a nice challenge for a computer scientist, as knowledge of operating systems and filesystem concepts are required and could be used practically. There is no demand for a whole filesystem driver, as the image does not need to be mountable on modern operating systems and no new files need to be written. Thus, a bitstream interpreter is sufficient. The Python programming language was used to write a first prototype of the interpreter as there were no performance requirements and it is very well suited for rapid development. By the end of the year a tool was produced that was able to read the filesystem headers and produce directory listings from them (Fig. 2).

In this example the volume header block (VHB) produces a checksum failure, but with the correct File Header Block the simple directory structure is readable. The listing seems to be correct as it reproduces the filenames like sysImage.sys which was readable in the hex editor. With this listing at least some information might be read from the filenames itself. The next stage was the file extraction feature which could extract single files from the image or dump all contained files into a folder on the host system. These could then be inspected further, to gather more knowledge of their original purpose.

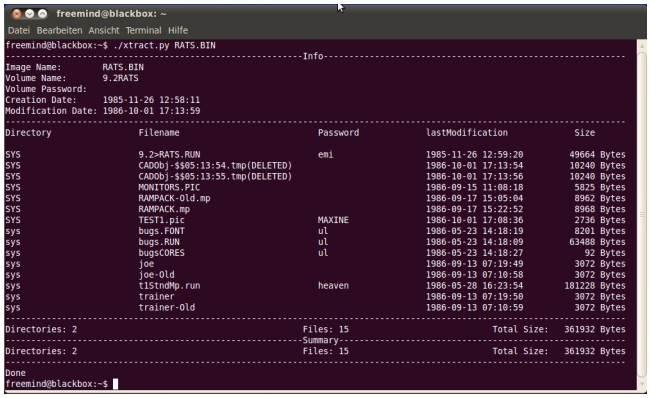

Figure 2: Python extractor file list output

2.3 Filesystem Interpretation

Of course it was possible to sneak a peek at the probable file contents before, by opening the floppy image file in a hex editor. But this made it very complicated, especially for non-text files to distinguish between file boundaries. Depending on the filesystem used and if fragmentation occured a single file is not necessarily contained in consecutive blocks on the storage medium. For the preservation and access needs of the Archive and the public institution donating the data, it was not necessary to re-implement the filesystem driver of the old platform for some recent one as most likely nobody will want to write files on floppy disks for this architecture again. But nevertheless a thorough understanding of the past filesystem is required to write a tool that can at least perform some basic filesystem functionality like listing the content of a directory and reading a specific file.

Fortunately the project was started early enough so that all relevant information that was coming from one specific site[2] on the net was copied locally in time. This site went offline and did not leave relevant traces either in the Internet Archive nor in the publicly accessible cache of search engines. This was a nice example of the challenges digital archaeologists face. Collecting institutions are advised for the future to store all relevant information on a past computer architecture on-site and not to rely on the permanent availability of online resources.

2.4 File Extraction

The extractor program knows the two possible base addresses of the volume header blocks (VHB), as there are an active VHB and one backup VHB defined by the CTOS specification. It selects by the checksum an intact VHB to find the File Header Blocks (FHB). If there is no correct checksum it looks at probable positions for suitable FHB addresses. In the next step the FHB are traversed sequentially to extract the contained file. It will also recover deleted files, files with an inactive header or password secured files. If there are several different, plausible file headers for a file, both files will be saved under different names. After storing the file, its name, directory, password, date and size are displayed as program output. If files cannot be properly

[2]The site *http://www.ctosfaq.com* went offline permanently, but was replaced later by work done by *http://OoCities.org* archivists at *http://www.oocities.org/siliconvalley/pines/4011*.

identified because of age and wear of the disk images, it will interpret the character encoding as ASCII-encoded strings, which can be extracted easily.

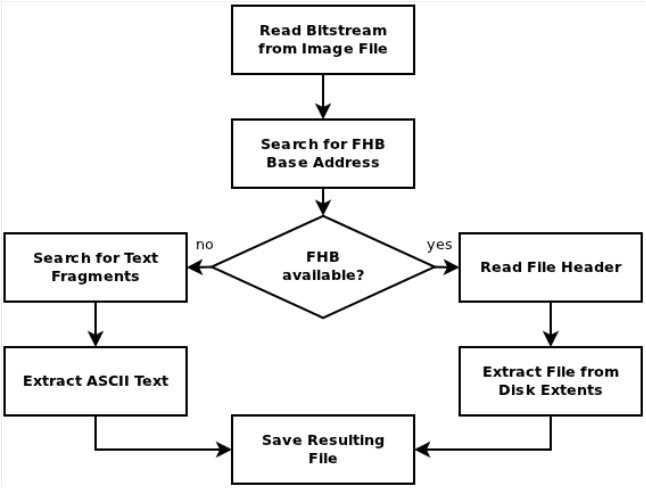

Figure 3: Python extractor for CTOS files and ASCII fragments

3. ALTERNATE APPROACH – FC5025

The RetroFloppy approach used the FC5025 floppy controller, currently available from DeviceSide Data. The FC-5025 is a USB-attached circuit board that interfaces to a number of once-common 360 KByte and 1.2 MByte floppy drives. Similar to the KryoFlux device, it reads flux transitions, but exposes much less detail to the user. Designed as a self-contained hardware and software package, it can read multiple disk formats from many different disk systems using a single floppy drive, and includes capabilities to extract entire disk images. Individual files can be extracted from a subset of the supported image formats of the FC5025 driver. In the CTOS case, though, there was no support built in. Fortunately, the FC5025 comes with C-language source code for the formats that are supported. Collaborating with DeviceSide and the other archivists, RetroFloppy wrote code that enabled the FC5025 device to read and interpret the CTOS filesystem, including extraction of individual files (Fig. 4). That support has been contributed back to the vendor for inclusion in future versions of their software package.

As the teams collaborated on filesystem interpretation, differences in strategies emerged and were shared. This ultimately strengthened both approaches. For example, one team felt it was important to extract even deleted files; the other team found significance in file timestamps and passwords. The filesystem interpretation by both teams ultimately relied heavily on available documentation of the CTOS system that would not be discernible by visual inspection. The timestamps, for example, were stored as an offset from May 1, 1952 – presumably a date that was somehow significant to the CTOS project itself, but was not discoverable simply given the disk image data.

4. RESULTS AND EVALUATION

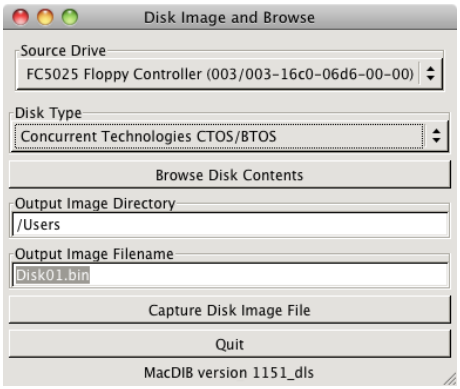

Figure 4: The user interface of the FC5025 solution, showing integrated CTOS support

The first round of the recovery experiment was run on 62 disk images created by the team in New Zealand from the received floppies. In three of those 62 images the File Header Block was unreadable. Two of the failing images had just half the size of the rest of them (320 KBytes instead of 640 KBytes). This issue led to missing file information like file address on the image and file length. For the third failing case it is still unclear why the File Header Block is unreadable. This comes to a total of 59 readable images with a total of 1332 identifiable files in them. The text content of the failing disk images was transferred to a single text file per image. At the moment the issues are being investigated together with the manufacturer of the reading device. It might be possible to tweak the reading process and extract more information to add the missing pieces for the failing images. This might lead to some deeper insight into the procedure and some best practice recommendations.

Filetype	Number of Files
No further recognized data	1635
ASCII text	106
ASCII English text	15
ISO-8859 text	11
XPack DiskImage archive data	7
DBase 3 data file (error)	5
FORTRAN program	2
Lisp/Scheme program text	2
Non-ISO extended-ASCII text	2
8086 relocatable (Microsoft)	1
ASCII C program text	1
Emacs v18 byte-compiled Lisp data	1
MS Windows icon resource (error)	1

Figure 5: Types of the recovered files from the floppy disks by *file* interpretation.

As the files of the New Zealand test set were mostly of ASCII text, a second set of floppies from the US coast guard was tested at RetroFloppy. This set spanned a longer period and contained many more non-ASCII files. Finally, there were 1889 files successfully extracted from the years 1983 through 1993. Of those, 1789 files with an active file header and 100 deleted files were recovered. The file type detection with the Linux file utility identified most files as ”unknown

binary data" (1635). Eighty-two of them could be identified as CTOS executables by the file extension "run". 838 could be attributed as Word Processor files by extension or manual inspection. Several files got categorized by the *file* utility (5), some of the attributions were simply wrong. In general the results were not widely cross-checked with other utilities as this was not the focus of this study.

5. CONCLUSION

The floppy disk recovery of physically intact media was a full success for both test sets, as it was possible to properly read files from outdated media without any original hardware available. Each disk took approximately five minutes to image and the research and initial forensic work added some additional hours to make up a total of one week of full time work for the initial imaging phase. Less than a month was required for a junior developer to write and test the Python code and less than a week for a seasoned C developer to produce the FC5025 driver code. The future per disk effort should now be very small with the tools available. The most troublesome part of the study was that the only way to understand the file system was to use documentation that has subsequently disappeared from where it was originally found on the internet. This highlights the need for some "body" to independently – not just on some site on the internet – preserve and make available all the system and software documentation for old digital technologies. Without that documentation this kind of work would be much more difficult if not impossible, and at least for the considered platform the documentation is rapidly disappearing.

There are at least two hardware solutions available today, providing an interface between outdated hardware and today's platforms. The KryoFlux device is shipped with proprietary software helping to fine-tune the image extraction. The Device Side USB floppy disk controller is priced very well below $ 100 and offers the source code of driver.[3] This is definitely a big plus in long-term access. A new controller is currently being developed[4] that will do similar work to both KryoFlux and DeviceSide FC5025, but is fully open source. So there is clearly interest in the industry in keeping a bridge to older devices open. Both approaches include the ability to extract entire disk images or browse disk directory contents. The two different hardware and software approaches taken here helped to validate and improve the results of both – primarily due to the fact that there were two independent teams working towards the same goal. In the end, the steps taken were the same and would need to be taken by anyone undertaking a project to decode disks of unknown origin:

1. Deduce whatever is possible by visual inspection of the pyhsical media (identifying marks on the disks themselves – bit density, sided-ness, even handwritten clues)

2. Employ a hardware solution to read the bits – Kryoflux is better at configuration "on the fly", DeviceSide FC5025 is simpler to use but requires *a priori* knowledge of and preparation for the format

3. Decode the resultant image and retrieve files by following the filesystem conventions of the system that created it.

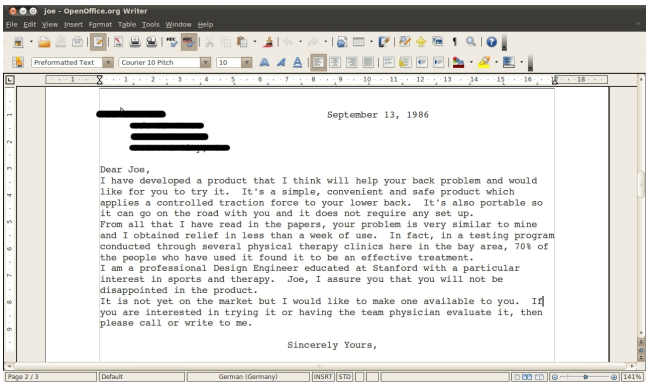

Figure 6: Interpretation of an ASCII text document in OpenOffice

The filetype detection test once again demonstrated the shortcomings of some of the existing tools. It would be great to add some of those files as well as some reference floppy images to a test set of files [3] as especially very old filetypes are under-represented in the existing detection libraries.[5] The study was brought to a point where some of the files – the ASCII text documents and fragments – could be interpreted, but the content and meaning of the binary data files remains mostly opaque. Another major challenge is the unavailability of software to properly run or render some of the extracted files. Emulation would have been the proper strategy to handle them, but neither a functional emulator nor the required additional software components are available for this computer architecture.

6. REFERENCES

[1] Florian Buchholz and Eugene Spafford. On the role of file system metadata in digital forensics. *Digital Investigation*, 1(4):298–309, 2004.

[2] Brian Carrier. *File System Forensic Analysis*. Addison Wesley Professional, 2005.

[3] Andrew Fetherston and Tim Gollins. Towards the development of a test corpus of digital objects for the evaluation of file format identification tools and signatures. *International Journal of Digital Curation*, 7(1), 2012.

[4] Matthew G. Kirschenbaum, Richard Ovenden, and Gabriela Redwine. *Digital Forensics and Born-Digital Content in Cultural Heritage Collections*. Council on Library and Information Resources, Washington, D.C., 2010.

[5] Edna I. Miller, Jim Croock, and June Loy. *Exploring CTOS*. Prentice Hall, 1991.

[3]The driver page, *http://www.deviceside.com/drivers.html*, gives a list of supported host operating systems and original environments.

[4]The DiscFerret controller, currently under development: *http://discferret.com/wiki/DiscFerret*.

[5]The Archive ran a couple of filetype detection experiments on their holdings showing a high failure rate for files dating before the mid 1990ies.