Virtualizing the CIC Floppy Disk Project: an Experiment in Digital Preservation Using Emulation

Geoffrey Brown
Indiana University

ABSTRACT

The CIC floppy disk project (FDP) is a partnership between the U.S. Government Printing Office and Indiana University Bloomington to make publications that were distributed to federal depository libraries on floppy disk available on the Internet. The FDP does not provide software to access these publications many of which depend upon obsolete application and operating system software. This paper describes an experiment to provide support for the publications in the FDP using obsolete software executing within off-the-shelf virtualization tools. These tools provide for the creation of "software images" consisting of an operating system and application software which can be executed in conjunction with data files by an application program to provide an "original" execution environment within a modern operating system. Virtualization is a special case of emulation which has frequently been discussed in the context of digital preservation, but has not been applied on any large heterogeneous data sets such as those in the FDP. The results of this paper apply equally to the emulation.

1. INTRODUCTION

Preserving access to digital objects in the face of rapid technological obsolescence is a challenging problem which can be addressed through four obvious strategies: (1) maintain the hardware and software used to create or capture the object, (2) maintain hardware and software capable of viewing the record in its native format, (3) ensure backward compatibility when the hardware and software are updated, or (4) migrate the object to a new format before the old format becomes obsolete[1]. This paper describes an experiment that utilizes commercial virtualization software to realize strategy (2) for the relatively large collection of diverse digital objects constituting the Floppy Disk Project [9].

Virtualization provides a means to execute an obsolete software environment consisting of an operating system and application programs on modern hardware. Virtualization (more generally emulation) provides a software layer on a modern computer system that allows the obsolete software environment to execute on a "virtual" hardware platform. This virtual platform provides an accurate model of a specific hardware configuration that is indistinguishable from actual hardware except in performance critical applications. The further removed a modern environment is from the hardware required by the obsolete software, the more hardware features must be emulated to provide this illusion. In the limit even the central processor must be emulated. The difference between virtualization and emulation is in the degree of mismatch between the virtual hardware and actual hardware that must be bridged by software. This difference is largely irrelevant to the work described in this paper – I assume the existence of an emulation platform and focus upon the issues required to create suitable software environments to access obsolete documents.

Emulation has frequently been discussed in the context of digital preservation [32, 13, 33, 15, 34, 21, 14]. While emulation has been used for a few specific preservation projects such as the BBC "Doomsday" book [23] and multimedia artwork [36], the strategy has not been tested on large diverse document collections. The FDP provides a substantial test case for emulation based preservation with a relatively large heterogeneous document collection. The FDP preserved all of the known U.S. Government Printing Office documents issued to federal depository libraries on floppy disk. The practical reasons for the original project were that the media were becoming obsolete and had poor long term archival properties, and represented an increasing burden on the depository libraries which had a legal responsibility to preserve them until officially obsoleted by the Government Printing Office. The documents include data such as agriculture census results, geological and geochemical data, and government statistics relating to the energy industries. The FDP created a web archive that provides basic documentation for the preserved document collections (each originally consisting of one or more floppy disks) along with 335 "zip" archives each of which preserves the contents of a single document collection. These zip archives contain further archives in a variety of formats which, when fully expanded include nearly 8000 individual files in a large number of formats. The files of the FDP were known to require IBM PC compatible hardware and required various versions of MS DOS

[1]These strategies are a paraphrase of C2.2.12.3 of DoD 5015.2-STD which specifies the requirements for record management by the Department of Defense and has been endorsed by the National Archives and Record Administration [7, 26].

and Windows operating systems, but little was documented about the application software requirements.

The process of virtualizing the FDP consisted of three major steps – analysis of the documents for software requirements, creation of a software "image" including both the operating system and required applications, and testing the document collection with this software image on an emulator. While the basic tools needed to build an emulation environment are available both commercially and through open source software projects [39, 38, 29, 1, 31, 44, 28], the technology needed to define a suitable environment for a heterogeneous document collection has not been developed. For the work described in this paper, I created tools to automatically evaluate the documents of the FDP to determine software requirements. Based upon these software requirements, I acquired the necessary application and operating system software which I then used to build a emulation environment. The remainder of this paper is organized as follows. I begin with a discussion of my broader research objective – providing networked access to "virtualized" document collections. I discuss my use of automated tools to evaluate the requirements of the FDP documents in Section 3, the virtual machine I created based upon these requirements and testing in Sections 4 and 5. The use of emulation to support digital preservation inevitably raises the question of who preserves the emulator; I address this issue in Section 6. I close with a discussion of results and plans for future work.

2. VIRTUALIZED DOCUMENT COLLECTIONS

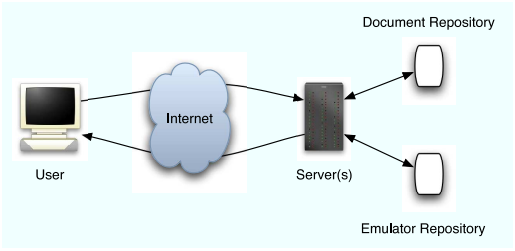

Figure 1: Model of Document Access

The model of document access that I aim to support is illustrated in Figure 1. A user's electronic request to access a document collection causes a pre-configured emulator to be dynamically customized, linked to the document collection, and made available through the Internet as a working emulator for exploring and interacting with the requested collection.

Tools to support emulation and virtualization are readily available. For most of the work described in this paper I used VMware Server [40]; I have also used Parallels [28], Bochs [1], and VirtualPC [38]. I do not have direct experience with Qemu [31], Xen [44], or Plex86 [29]. [2] My experience is that most of these tools are used in quite similar ways; the primary differences being the host hardware and guest software supported, the user interface tools supplied, and whether the tool has a scripting interface.

Key technical requirements for the underlying emulation platforms are a software API that enables the control and configuration of virtual machines (emulators), a way to deliver "console" access over the Internet, and mechanisms to enable file transfer with and printing from a virtual machine. In this study, I utilized an open source implementation of the VNC protocol [41] to provide web access. I found that VNC access could be provided in a relatively secure manner using dynamically configured passwords. One limitation of VNC is that it does not provide remote audio. I supported printing by installing a generic Postscript printer driver from Adobe configured to print to files. To support file transfer I created a "shared folder" on the guest virtual machine. I have experimented with using the Samba [35] open source tools to access this shared folder from the host. My intention is to provide users access to this shared folder by utilizing the WebDAV support in the Apache web server.

3. SOFTWARE REQUIREMENTS

A key enabling technology for the application of emulation to document preservation is the development of tools to identify the software required for the emulation environment – in this project, a collection consisting of thousands of distinct files. It is impractical to manually examine each file for name and contextual information that might reveal the required software.

Identifying file types is as much art as science. There are no standards that allow completely reliable identification of file types by examining either the file name or the file contents. However, there are imperfect techniques based upon file naming conventions (i.e suffixes) and upon examining file contents which, when combined with contextual information, appear to give good results. Many of the FDP collections contain files from a similar time period and are intended to be executed on MSDOS or Windows 3.1 operating systems – we can eliminate from consideration file types associated with Unix or Macintosh operating systems. Similarly, we know that most of these document collections contain government data in database or spreadsheet form along with text documents describing the collections. This contextual information allowed me to eliminate many clearly spurious file type attributions; by slightly customizing the tools to eliminate file type identification information that is clearly inconsistent with the context, I believe the accuracy of identification could be greatly improved.

I used three primary techniques for identifying individual files – a standard Unix utility called `file` [10] which compares various internal file features against a database of known patterns, a Perl command which tests to see if a file is "binary", and finally a list of well-known file naming suffixes. In the following I discuss each of these techniques in more detail. In addition, the FDP project consists of a large number of compressed "Zip" archives many of which contain other archives. I wrote a Perl script that recursively expands archives identified with the Unix `file` utility; the root files are identified using the three techniques mentioned previously.

The Unix `file` utility attempts to identify file formats based upon "magic" numbers which, in well designed file formats, may consist of a small number of binary digits at the beginning of a file. Unfortunately, many file types are

[2]Xen and Plex86 do not support Windows guest operating systems.

```
Library_1.zip : Zip archive data, at least v2.0 to extract
        INSTALL.DAT : ASCII C program text, with CRLF line terminators
        INSTALL.EXE : MS-DOS executable, MZ for MS-DOS
        DDB.001 : data (binary)
        DISK.ID : ASCII text, with CRLF line terminators
Library_2.zip : Zip archive data, at least v2.0 to extract
        INSTALL.EXE : MS-DOS executable, MZ for MS-DOS
        INSTALL.DAT : ASCII C program text, with CRLF line terminators
        DISK.ID : ASCII text, with CRLF line terminators
        DDB.001 : data (binary)
Library_3.zip : Zip archive data, at least v2.0 to extract
        FLIC_94F.DBF : DBase 3 data file (1234 records)
Library_4.zip : Zip archive data, at least v2.0 to extract
        UMINSTR.DOS : (Corel/WP)
        UMINSTR.WIN : (Corel/WP)
        UMWP51.DOS : (Corel/WP)
        UMWP61.WIN : (Corel/WP)
        VOCED.EXE : MS-DOS executable, MZ for MS-DOS, PKLITE compressed, \
                    ZIP self-extracting archive
                vl_help.dbf : DBase 3 data file (209 records)
                vl.exe : MS-DOS executable, MZ for MS-DOS
                vl_descr.ndx : DBase 3 index file
                vl_keycd.ndx : VMS Alpha executable
                vl_keywd.dbf : DBase 3 data file with memo(s) (219 records)
                vl_keywd.dbt : data (binary)
                vl_keywd.ndx : data (binary)
                ... (roughly 150 files elided)
...
```

Figure 2: Document Analysis

not designed for ease of identification and, in the general case, magic numbers may consist of patterns including embedded strings and sequences of numbers at variable locations in the file. The basic algorithm is to apply the patterns in a database sequentially until a match occurs. In practice, `file` is easy to use. For example, when executed on the WordPerfect file "unwitmit.wp5", the program responds with a correct identification (though the software version isn't identified).

```
> file unwitmit.wp5
unwitmit.wp5: (Corel/WP)
```

The database distributed with `file` contains many 100's of patterns; with so many formats and a sequential search strategy, false identifications frequently occur. However, the program is configurable and it is relatively simple to prune the database of formats that are inappropriate to the context (e.g. binaries for Alpha processors in the FDP document collection). Although I have not done so for the work described. Furthermore, the algorithms of the `file` utility are available as a code library which is easily integrated into other programs. A Perl library, `File::MMagic` [24], provides similar functionality.

The second approach that I used when `file` failed was to compare the file suffix against a database of known suffixes. Unfortunately, the result of such a comparison is often at best a hint. For example, the ".DOC" suffix was used in the files of the FDP for WordPerfect, Microsoft Word, and ASCII text. There are a large number of web sites that provide lists of suffixes and the known applications that generate them. Many of these lists contain multiple identifications for some common suffixes.

The third identification technique I used, the Perl command to determine whether a file is text or binary, applies a relatively simple algorithm; it examine the first few hundred characters of a file and if most are text characters, then the file is determined to contain text.

These three identification techniques were combined in a set of scripts that recursively expand the archives of the FDP project. The output of this process is a text report illustrated in Figure 2.

The "root" level Zip files from the FDP are named "Library_... ." Whenever an archive is extracted, the files within that archive are indented in the generated report. Each line of the report is of the form "file-name : file-type." The file types are determined from the analysis process and are not always accurate. For, example, we can be reasonably certain that the file "vl_keycd.ndx" is not an executable for an Alpha processor. As mentioned, I have not yet attempted to prune the database used by `file`. Furthermore the context gives clues to more appropriate identification: while the fragment displayed for Library_4.zip contains three files with the suffix ".ndx" one of which is correctly identified as a dBASE III (Dbase 3) file, one incorrectly identified as an Alpha executable, and the third unidentified. The complete report includes approximately 150 dBASE III files in Library_4.zip, thus we can be relatively certain that all three files are related to dBASE III.

Returning to the example, note that Library_1.zip and Library_2.zip each contain an MS-DOS executable binary with some indeterminate binary data; Library_3.zip appears to contain a dBASE III file, Library_4.zip contains documen-

```
    elsif ($info =~ m/Zip/)
    {
        $tmpdir = tempdir( CLEANUP => 1 );
        system(``unzip $f -d $tmpdir > /dev/null'');
        $recursecnt++;
    }
    elsif ($info =~ m/ARC archive/)
    {
        $tmpdir = tempdir( CLEANUP => 1 );
        pushdir($tmpdir);
        system(``arc x \"$d/$f\" > /dev/null'');
        $recursecnt++;
        popdir();
    }
...
    elsif (($info =~ m/SFX|PKLITE/) && !($info =~ m/PKUNZIP.EXE/))
    {
        $tmpdir = tempdir( CLEANUP => 1 );
        pushdir($tmpdir);
        system(``cp \"$d/$f\" .'');
        system(``echo \"\n\" | dosemu -dumb $f > /dev/null'');
        system(``rm $f'');
        $recursecnt++;
        popdir();
    }
```

Figure 3: File Type Processing

tation in WordPerfect format and an "executable archive," VOCED.EXE. By executing VOCED.EXE in an emulation environment (DosEMU [8]), a further set of files was extracted including various dBASE III files.

Even with this brief example, it should be clear that identifying file types is a challenging problem. There are some clear miss-identifications as well as some unidentified file types. Furthermore, document collections contain compressed archives in a variety of formats including "self-extracting" files which must be uncompressed to recursively identify the types of their contents. Within the FDP files I found ZIP, ARC, MSCompress, PKLite, LHa, and CAB archives each of which requires a separate extraction tool or technique.

As described above, my approach involves recursively evaluating the files of the FDP project – each time an archive file was encountered, I extracted the contents of that archive and then evaluated those contents. This evaluation was performed with a short Perl script in conjunction with the `file` utility discussed above, a set of open source utilities for extracting archives, and a DOS emulator (dosemu[8]) which I used to extract the executable archives. The core of the evaluation process utilizes the "info" extracted using `file` and decides upon further processing. Figure 3 illustrates three such rules for processing ZIP, ARC, and PKLITE or SFX executable archives. The first two cases can be evaluated in the native Unix environment, while the third utilizes dosemu. In each case, a temporary directory is created and the archive extracted within that archive. Not shown is the recursive step which evaluates each of the newly extracted files. The described technique is similar to that used at NIST to generate file hash sets for forensic computing [42].

In summary, while the described techniques could benefit from fine tuning, I found them to be extremely useful in practice. The set of applications I identified and the virtual machine I created based upon these results are discussed in Section 4.

There are other research efforts to perform file identification such as Droid [2, 30] and JHove [17]. In comparison with Droid, I found `file` is far faster, easier to use, and appears to use similar magic number patterns. JHove currently supports only a small number of the formats I have encountered in this study. There are several other ongoing projects to collect file format information in preservation repositories including the Global Digital Format Registry [12] and Fred, the Format Registry Demonstrator [11]. Other projects are described on the Library of Congress digital preservation site [19].

4. AN EMULATION ENVIRONMENT

Using the tools described in Section 3 I analyzed 335 Zip files containing 7717 files. In this section I describe the results of this analysis, the software image I built to support access to the FDP files.

My analysis identified 127 distinct file types of which the top 20 most frequent are listed in Figure 4. Of these, only one type seems to be a clear miss-identification. I was able to make reasonable guesses for about half the 551 "binary" files with a simple suffix analysis tool. Based upon these results I built a software image with the Windows 98 operating system and applications supporting WordPerfect, Lotus 1-2-3, dBASE III, MS Word, and various archive expansion tools. While I also found individual collections requiring TurboPascal, fortran, SAS, and arc/info I did not include supporting tools in my emulation environment because the expense was not justified for this experiment. My selection of Windows 98 was based upon the fact that this was the last windows release built on MSDOS. I knew both from the FDP documentation and my analysis that many of the documents

```
2323  data (ASCII)
 858  Lotus 1-2-3 wk1 document data
 851  ASCII text, with CRLF line terminators
 551  data (binary)
 532  ASCII English text, with CRLF line terminators
 508  DBase 3 data file
 361  Zip archive data, at least v2.0 to extract
 228  binary Computer Graphics Metafile
 206  MS-DOS executable, MZ for MS-DOS
 166  Zip archive data, at least v1.0 to extract
 166  MS Compress archive data
 119  VMS Alpha executable
 118  directory
 101  MS-DOS executable, NE for MS Windows 3.x (driver)
  86  Lotus 1-2-3 wk3 document data
  82  MS-DOS executable, NE for MS Windows 3.x
  78  ARC archive data, dynamic LZW
  64  Lotus 1-2-3
  57  (Corel/WP)
  47  Non-ISO extended-ASCII English text, with CRLF line terminators
```

Figure 4: Top 20 file types from Magic Number Analysis

and applications were originally created for MSDOS. While it is likely that later versions of Windows could support some of these applications, such analysis is beyond the scope of this experiment.

5. TESTING

The FDP archive consists of 335 zip files organized as 108 document groups. The groups correspond to the original "SUDOC" classification provided by the GPO and are each described by a web page containing basic cataloging information and limited notes on software requirements. To test the emulation environment defined in Section 4, I attempted to install each of the 108 document groups. This included executing applications and accessing documents where appropriate.

The most significant issue I encountered with testing involved the various installer programs included with the documents. Many of these depended upon the original floppy organization to operate correctly. I was able to overcome these problems in most cases. The two primary approaches I used were to expand archives without the installer and to create multiple copies of files in specially named directories (e.g. `c:disk1`). There were two programs which I managed to install, but which did not execute correctly due to apparent "path" problems. It is possible that these could be resolved with further work. One document collection contained MSDOS, 32-bit windows, and 16-bit windows versions of the same program. I only succeeded in installing the 32-bit windows and MSDOS versions.

I found only one file type that I could identify but hadn't seen in the software analysis stage – Microsoft Access. There were two archives with Access files encapsulated in proprietary binaries that were extracted during installation. I found one archive containing six binary files in an unrecognized format. There was no documentation with this archive to assist in identification.

In addition to the install problems described above, I found three applications that appeared to install correctly, but failed when executed. I found a third program which appeared to execute correctly but depended upon MSDOS printing support which I had not provided.

Most of my testing was performed using a VNC connection over a home DSL connection to an Indiana University server. I encountered no performance issues with this approach; however, I had some difficulty sending F1-F10 keys from a Macintosh computer using VNC.

The most disturbing issue I encountered in testing was a large number of errors in the FDP archive. There were 16 missing zip files (in addition to the 335 I tested). At least two other zip files had either missing components or spurious extra files.

6. PRESERVING THE EMULATOR

Emulation as a preservation strategy would appear to have a potentially fatal weakness – continued access to documents supported by an emulator requires preserving the viability of the emulator across generations of machines. This fear has led to the development of emulators based upon the idea of a Universal Virtual Computer (UVC) which could be defined in a highly portable manner that can be more easily maintained across generations [20, 21, 16]. While research on more portable emulation architectures is important to improve the viability of emulation as a long term preservation strategy, I believe the underlying fear – that the emulator itself may be lost – is overblown.

To explore the complexity of emulators compared with other software systems, I obtained the source code or used published data for four categories of software – emulators, office applications, user interfaces, and operating systems [3, 4, 5, 6, 22]. I used the tool sloccount for those packages that I analyzed [37]. The data are illustrated in Figure 5. The metric that I use in comparing software complexity is lines of source code (LOC). The key observations are that the emulator packages that I analyzed – including complete machine emulators capable for executing Windows XP such as Qemu – are of the same order of complexity as xpdf [45]

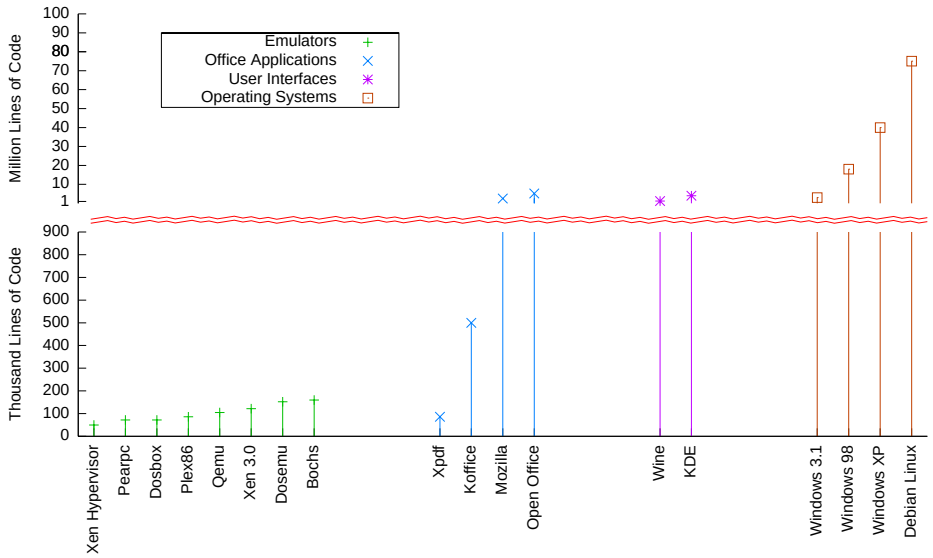

Figure 5: Lines of Code for Software Packages

which is a commonly used open source implementation of a pdf interpreter (105,000 LOC for Qemu vs 86,000 LOC for xpdf). In contrast, emulators are 25 to 50 times simpler than office tools such as Mozilla [25] or OpenOffice [27], 10 to 40 times simpler than user interface packages such as KDE [18] or Wine [43] (a windows application interface emulator). Finally, emulators are incomparable to operating systems in complexity. Not illustrated on the graph is Jhove, a tool developed for validating files for preservation [17]. Even this basic preservation tool, with 57,000 LOC, is not significantly less complex than an emulator. Thus, I assert that preserving an emulation environment over time appears to be a relatively manageable problem.

7. DISCUSSION

I have described an experiment to virtualize the large and diverse collection of digital documents constituting the Floppy Disk Project. This work has helped to define the technical problems that must be solved in any such virtualization effort. I am currently working to extend the tools I developed to the larger class of documents published by the United States Government Printing Office on CD-ROM. As mentioned, My long term goal is to develop the technologies to deliver emulator supported access to collections such as the FDP to patrons distributed across the Internet. While I have experimented with the tools needed to realize this goal, I have not begun to construct a web site that would be accessible outside my research group.

Emulation is not the only solution for archiving digital documents; however, I believe it may be necessary even if migration tools exist. Migration (indeed all forms of digital preservation) has a risk of information loss and cannot be depended on absolutely. Thus, it is important to preserve access to original documents even if more convenient renditions are available. Emulation may also prove to be a useful technology to support migration; it may be convenient to use the original application API to implement migration tools and it may be simpler to preserve migration tools within emulation environments. Furthermore some of the work described in this paper would be required even if the objects were to be preserved through migration. For example, a necessary step for both migration and virtualization is the evaluation of the objects to determine their software requirements. Ironically, in order to perform this evaluation, I found it necessary to evaluate some objects in an emulation environment.

I haven't addressed the preservation of application software and the preservation of instruction manuals. Emulation assumes the availability of the necessary application software; however, it is increasingly difficult to find some applications. For this project I resorted to ebay in order to find some software. Even the operating systems are beginning to be hard to find. Microsoft recently dropped support for Windows 98; the practical effect is that the software is no longer available through MSDN – the Microsoft developer network. The free reader I used for Lotus files is no longer distributed, though I found a copy on the Internet.

The use of obsolete software assumes that people will continue to know how to use the software. At the least this requires preservation of instruction manuals. I believe that scripting virtual machines can help – for example, the installation steps can be automated and the key files can be opened in a reader.

There are also legal and copyright issues associated with making copies of digital documents available on the web. With United States Government documents there are no such copyright restrictions; however, in the general case these issues must be explored.

Acknowledgments

I am grateful to Lou Malcomb and Julianne Bobay of the Indiana University Libraries for suggesting this project, for providing data enumerating the digital documents in the library government document collections, and for facilitating

my access to the collections.

8. REFERENCES

[1] The cross platform ia-32 emulator. http://bochs.sourceforge.net/. Accessed September, 2006.

[2] A. Brown. Digital preservation technical paper 1: Automatic format identification using PRONOM and DROID. Technical report, National Archives, UK, March 2006.

[3] koders: Searching 225,816,744 lines of code. http://www.koders.com. Accessed September, 2006.

[4] http://physos.net/fun/kde_LOC_statistics.txt. Accessed September, 2006.

[5] Xen source. http://www.xensource.com/products/xen/. Accessed September, 2006.

[6] SLOCCount home page. Counted using SLOCCcount, http://www.dwheeler.com/sloccount/. Accessed September 19, 2006.

[7] Design criteria standard for electronic record management software applications (DoD 5015.2-std). http://www.dtic.mil/whs/directives/corres/html/50152std.htm. Accessed September, 2006.

[8] Dosemu. http://www.dosemu.org. Accessed September, 2006.

[9] CIC floppy disk project. http://www.indiana.edu/~libgpd/mforms/floppy/floppy.html. Accessed September, 2006.

[10] Fine free file command. http://www.darwinsys.com/file. Accessed September, 2006.

[11] Fred: A format registry demonstration. http://tom.library.upenn.edu/fred/. Accessed November, 2006.

[12] Global digital format registry. http://hul.harvard.edu/gdfr/. Accessed November, 2006.

[13] S. Gilheany. Preserving digital information forever and a call for emulators. In *Digital Libraries Asia 98: The Digital Era: Implications, Challenges, and Issues*, 1998.

[14] M. Hadestrom, editor. *It's About Time: Research Challenges in Digital Archiving. Final Report Workshop on Resaerch Challenges in Digital Archiving and Long-Term Preservation*. National Science Foundation, August 2003.

[15] A. R. Heminger and S. Robertson. The digital rosetta stone: a model for maintaining long-term access to static digital documents. *Communications of AIS*, 3(1es):2, 2000.

[16] J. R. V. D. Hoeven, R. J. V. Diessen, and K. V. D. Meer. Development of a universal virtual computer (uvc) for long-term preservation of digital objects. *Journal of Information Science*, 31(3):196–208, 2005.

[17] JHOVE - JSTOR/Harvard object validation environment. http://hul.harvard.edu/jhove/. Accessed September, 2006.

[18] K desktop environment. http://www.kde.org/. Accessed September, 2006.

[19] Sustainability of digital formats planning for Library of Congress collections. http://www.digitalpreservation.gov/formats/index.shtml. Accessed November, 2006.

[20] R. A. Lorie. Long term preservation of digital information. In *JCDL '01: Proceedings of the 1st ACM/IEEE-CS joint conference on Digital libraries*, pages 346–352, New York, NY, USA, 2001. ACM Press.

[21] R. A. Lorie. A methodology and system for preserving digital data. In *Proceedings of the second ACM/IEEE-CS joint conference on Digital libraries*, pages 312–319. ACM Press, 2002.

[22] G. McGraw. From the ground up: The dimacs software security workshop. *IEEE Security and Privacy*, 01(2):59–66, 2003.

[23] P. Mellor. CaMiLEON: emulation and BBC doomsday. *RLG DigiNews*, 7(2), 2003.

[24] File::mmagic. http://search.cpan.org/~knok/File-MMagic-1.16/MMagic.pm. Accessed September, 2006.

[25] Mozilla. http://www.mozilla.org/. Accessed September, 2006.

[26] Endorsement of DoD electronics records management application (RMA) design criteria standard, version 2. NARA Bulletin 2003-03 http://www.archives.gov/records-mgmt/bulletins/2003/2003-03.html. Accessed September, 2006.

[27] Openoffice: Free office suite. http://www.openoffice.org. Accessed on September, 2006.

[28] Parallels. http://www.parallels.com. Accessed September, 2006.

[29] The new Plex86 x86 virtual machine project. http://plex86.sourceforge.net/. Accessed September, 2006.

[30] The technical registry Pronom. http://www.nationalarchives.gov.uk/pronom/. Accessed November, 2006.

[31] QEMU open source processor emulator. http://fabrice.bellard.free.fr/qemu/. Accessed September, 2006.

[32] J. Rothenberg. Ensuring the longevity of digital information. *Scientific American*, 272(1):42–47, January 1995.

[33] J. Rothenberg. *Avoiding Technological Quicksand: Finding a Viable Technical Foundation for Digital Preservation*. Council on Library & Information Resources, 1999.

[34] J. Rothenberg. Using emulation to preserve digital documents. Technical report, Koninklijke Bibliotheek, July 2000.

[35] Samba opening windows to a wider world. http://www.samba.org. Accessed October, 2006.

[36] Seeing double emulation theory and practice. http://www.variablemedia.net/e/seeingdouble/home.html. Accessed November, 2006.

[37] SLOCCount web for debian sarge. http://libresoft.dat.escet.urjc.es/debian-counting/sarge/. Accessed September, 2006.

[38] Microsoft virtual PC 2004. http://www.microsoft.

com/windows/virtualpc/default.mspx. Accessed September, 2006.

[39] Vmware home page. http://www.vmware.com Accessed September 19, 2006.

[40] Vmware server. http://www.vmware.com/products/server/. Accessed September, 2006.

[41] About RealVNC. http://www.realvnc.com/. Accessed September, 2006.

[42] D. White and J. Tebbutt. Digital forensics – using perl to harvest hash sets, June 2004. http://www.nsrl.nist.gov/documents/yapc2004/index.html. Accessed September, 2006.

[43] About wine. http://www.foolabs.com/xpdf/. Accessed September, 2006.

[44] The xen virtual machine monitor. http://www.cl.cam.ac.uk/Research/SRG/netos/xen/. Accessed September, 2006.

[45] Xpdf. http://www.foolabs.com/xpdf/. Accessed September, 2006.