Data-Driven Reporting and Processing of Digital Archives with Brunnhilde

Tim Walsh

University of Florida

Introduction

Archivists are now several decades in to appraising, arranging, describing, preserving, and providing access to digital archives and have developed and adopted a number of tools to aid in specific tasks along the way. This article discusses Brunnhilde, a new tool developed to address one of the first steps in working with born-digital materials: characterizing the overall contents of directories or disks to enable smart evidence-based decision-making in the appraisal, arrangement, and description processes.

Among the tools archivists have developed or adopted are a number of very robust file format identification programs, including DROID[1], Siegfried[2], and Apache Tika.[3] These tools, often relying on the PRONOM[4] file format registry, precisely identify the formats and versions of files by comparing parts of a file's bit stream (i.e., code) against known signatures. Although work remains in refining these tools and expanding the breadth and depth of the registries upon which they rely, these tools very adequately fulfill the function of taking a single digital file and accurately identifying what it is.

In his daily work as Digital Archivist at the Canadian Centre for Architecture (CCA), a research museum in Montreal "with the specific aim of increasing public awareness of the role of architecture in contemporary society and promoting scholarly research in the field[5]," the author identified a perceived gap in the tools available to archivists. Despite the existence of the aforementioned tools that can precisely identify a singular file or small collection of files, there seemed to be no existing tools that would succinctly but comprehensively summarize the contents of entire directories or disks in a way that would help archivists establish intellectual control of these materials and make informed decisions about their treatment.

The author first conducted some informal research to ensure that no existing tools already met this need. BitCurator[6] offered a number of forensic reporting capabilities, built around the use of DFXML[7], Bulk Extractor[8], Sleuth Kit[9], and scripts developed by the BitCurator team. Though these tools are powerful and

fulfill their purpose well, they do not quite cover the identified gap. The simplest reason is because the BitCurator reporting tools work exclusively with disk images, not directories, and a significant number of files in CCA's digital archives arrive in a form – e.g., a Dropbox transfer, or as files copied onto an external hard drive – that are not amenable to or do not warrant forensic imaging. In addition, the BitCurator reports do not include detailed or robust file format identification – much needed information at CCA, which has in its collections a number of obscure file formats, including computer-aided design formats from the 1980s and 1990s. In short, despite all of their advantages and utility, the existing tools offered in BitCurator could not be applied to the entirety of CCA's holdings, nor do they contain all of the information needed to make informed decisions around appraisal, arrangement, and description.

The research then led to another, more fruitful avenue: Ross Spencer's DROID analysis engine.[10] This tool, initially developed to work solely with the DROID file format identification tool, has a simple but powerful premise: take the outputs of a DROID scan of a directory and load them into a small portable database from which useful queries can be run to generate information about the directory as a whole. This is a significant improvement on DROID's built-in reporting functionalities, which were used at CCA prior to the author's arrival as Digital Archivist.

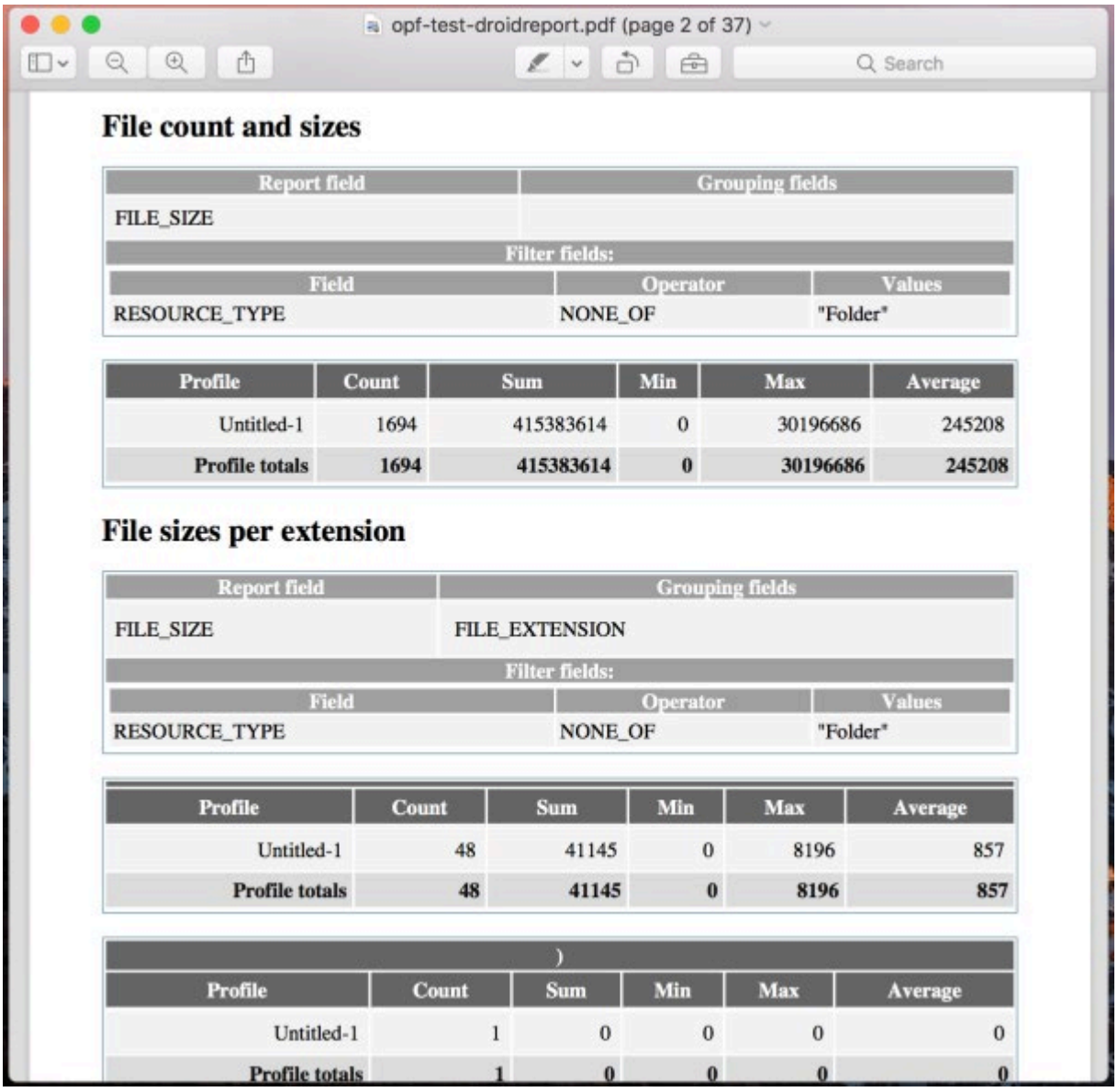

Figure 1: An example of a standard Comprehensive report from DROID

Testing at CCA had shown Siegfried to be more accurate than DROID in identifying a number of obscure file formats present in large quantities in CCA's archival collections. This prompted the author to embark on a programming project to implement Ross Spencer's powerful idea with Siegfried to neatly fit the gap identified in CCA's workflows.[11]

After a few months of on-again, off-again development and a few early releases, Brunnhilde 1.0.0 (named as a companion to Siegfried, as Roy was already taken) was released in August 2016. As of the final revisions of this article in July 2017, the latest version of Brunnhilde is Brunnhilde 1.5.0. A graphical user interface wrapper for Brunnhilde designed to be easily installed in the BitCurator environment was released in September 2016. An improved version of this GUI is forthcoming.

Tool Description

Brunnhilde is a command-line and graphical interface tool written in Python that builds on Siegfried to provide useful reporting for the contents of directories and disk images. It is designed to be usable on any Linux or macOS machine with minimal configuration, with a particular emphasis toward ease of use in the BitCurator environment, where Brunnhilde is natively installed as of v1.8.0. The user identifies a source directory or disk image, selects a destination for outputs, inputs an identifier for the material being scanned, and optionally passes some arguments to modify the script's behavior. Brunnhilde then:

- Creates a new directory for reports
- If source is a disk image, exports files from the image to a processing directory using SleuthKit's tsk_recover[12] or HFSExplorer[13], depending on the file system and options passed to Brunnhilde
- Conducts a virus scan using ClamAV[14] (this can optionally be skipped by the user)
- Runs Siegfried against the content and writes results to a "siegfried.csv" file.
- Loads the Siegfried CSV output into a SQLite3[15] database, stored as a file in the report directory
- Queries the database to generate CSV reports on:
    - File formats, sorted by count
    - File format versions, sorted by count
    - MIME types, sorted by count
    - Last modified dates (years), sorted by count
    - Unidentified files in the source
    - Files in the source that raised a warning in Siegfried
    - Files in the source that raised an error in Siegfried
    - Duplicate files (identified and grouped by checksum)
- Optionally, runs Bulk Extractor against the content and stores results in a new directory
- Uses the "tree" utility[16] to create a graphical depiction of the source's existing hierarchical arrangement
- Writes a human-readable HTML report intended for use by archivists

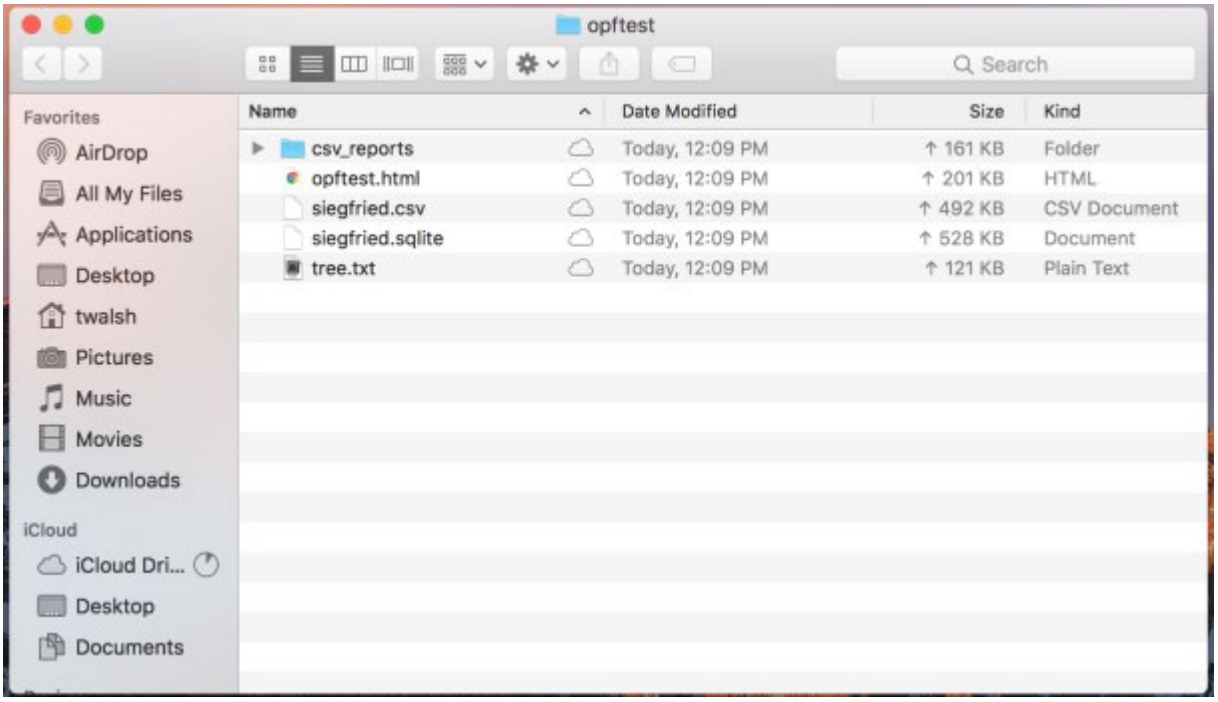

Figure 2: Brunnhilde outputs

The reports generated serve several functions. The CSV reports, named and stored in a consistent manner, are machine-actionable; they can be used as sources of information for other tools, such as CCA Disk Image Processor[17] and Folder Processor[18] to perform tasks such as pre-populating information such as common file formats into file inventory description spreadsheets.

The HTML report is intended to be read by archivists with the aim of helping them to understand the contents and characteristics of a source directory or disk as a whole and support better decision-making. This report is worth exploring in a little more detail.

HTML report

The HTML report generated by Brunnhilde has a few sections:

Descriptive information

The report starts with two basic pieces of information: the filepath of the source scanned and the accession number/identifier passed to Brunnhilde by the user.

Provenance information

The next section contains provenance information about the scan itself: the versions of Brunnhilde and Siegfried utilized, the exact Siegfried command called by Brunnhilde, and the time the scan commenced.

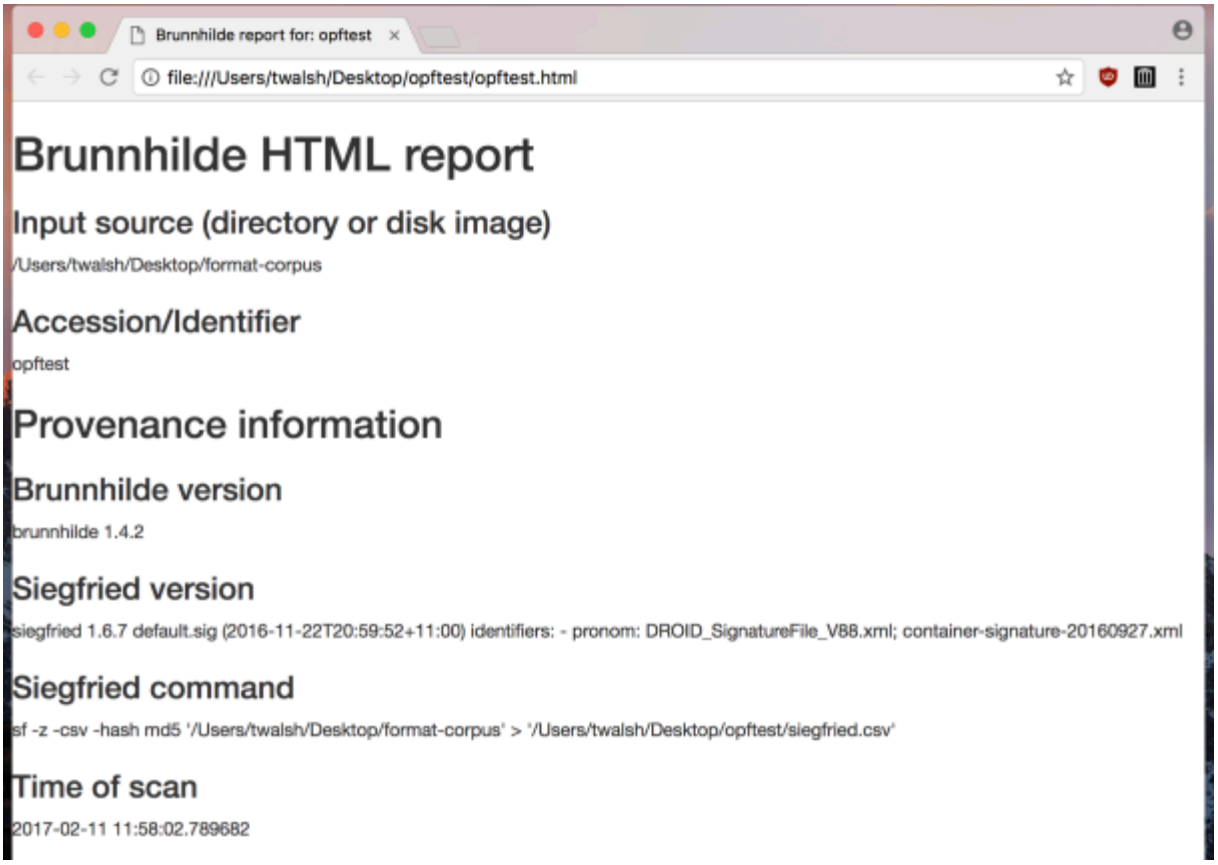

Figure 3: HTML report – Provenance information

Aggregate stats

The next section provides a high-level overview of the source as a whole. This includes the number of files and total size of the source, a date range and earliest and latest dates based on last modified dates in the file system, and information about the number of distinct, duplicate, and empty files. The section concludes with a high-level overview of Siegfried's file identification efforts, giving the number of identified file formats, number of unidentified files, and number of files that triggered either a warning (e.g. a mismatch between determined file format and the file's file extension) or error (e.g. file is empty) during the Siegfried scan.

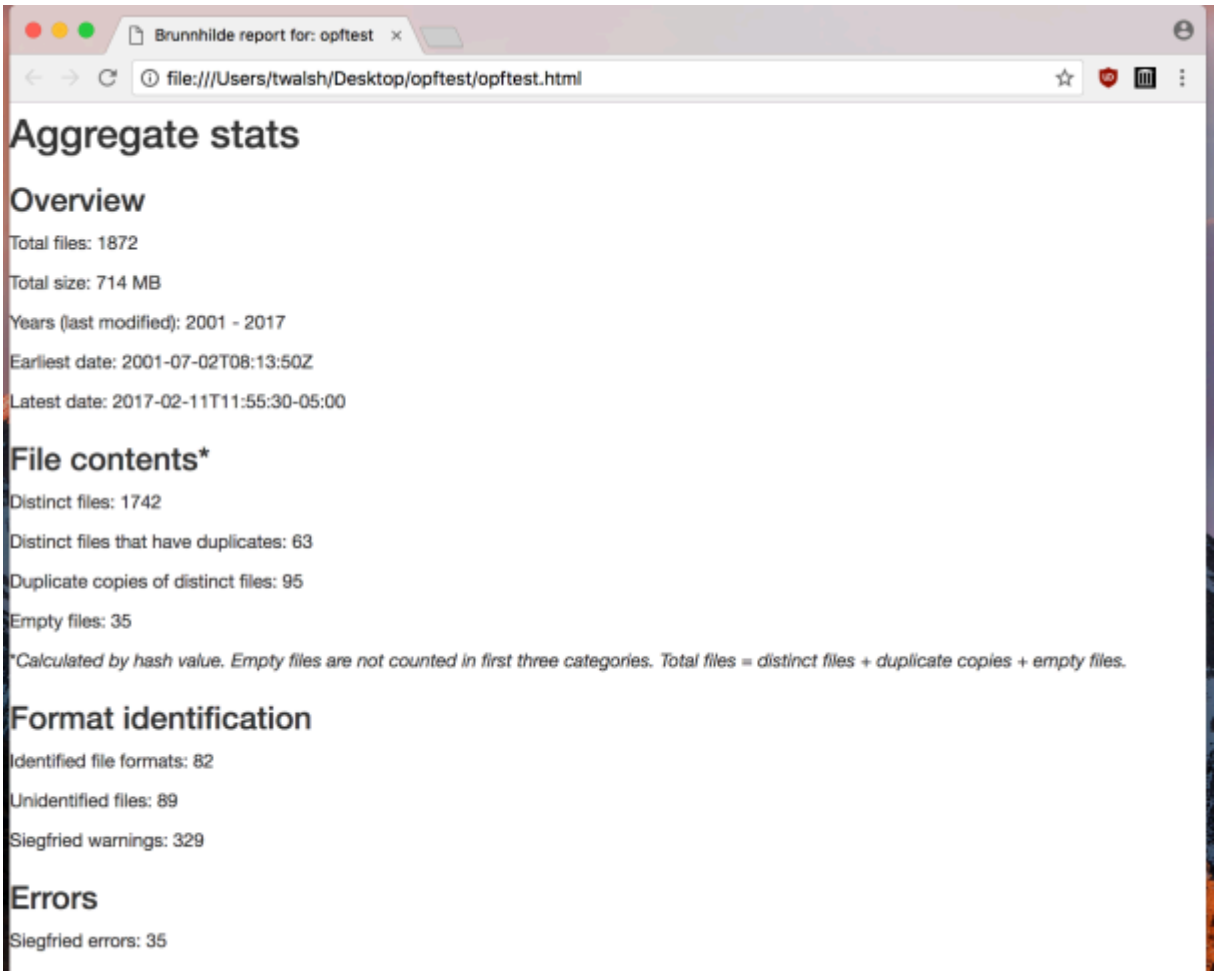

Figure 4: HTML report – Aggregate stats

Virus scan report

In Brunnhilde 1.2.0 and later, the HTML report contains a report of the ClamAV virus scan undertaken by default in Brunnhilde. If the user chooses not to run a virus scan, this section of the report reads “Virus scan skipped.”

Detailed reports

The report concludes with a number of detailed reports based on the CSV reports queried from the SQLite database and, optionally, Bulk Extractor.

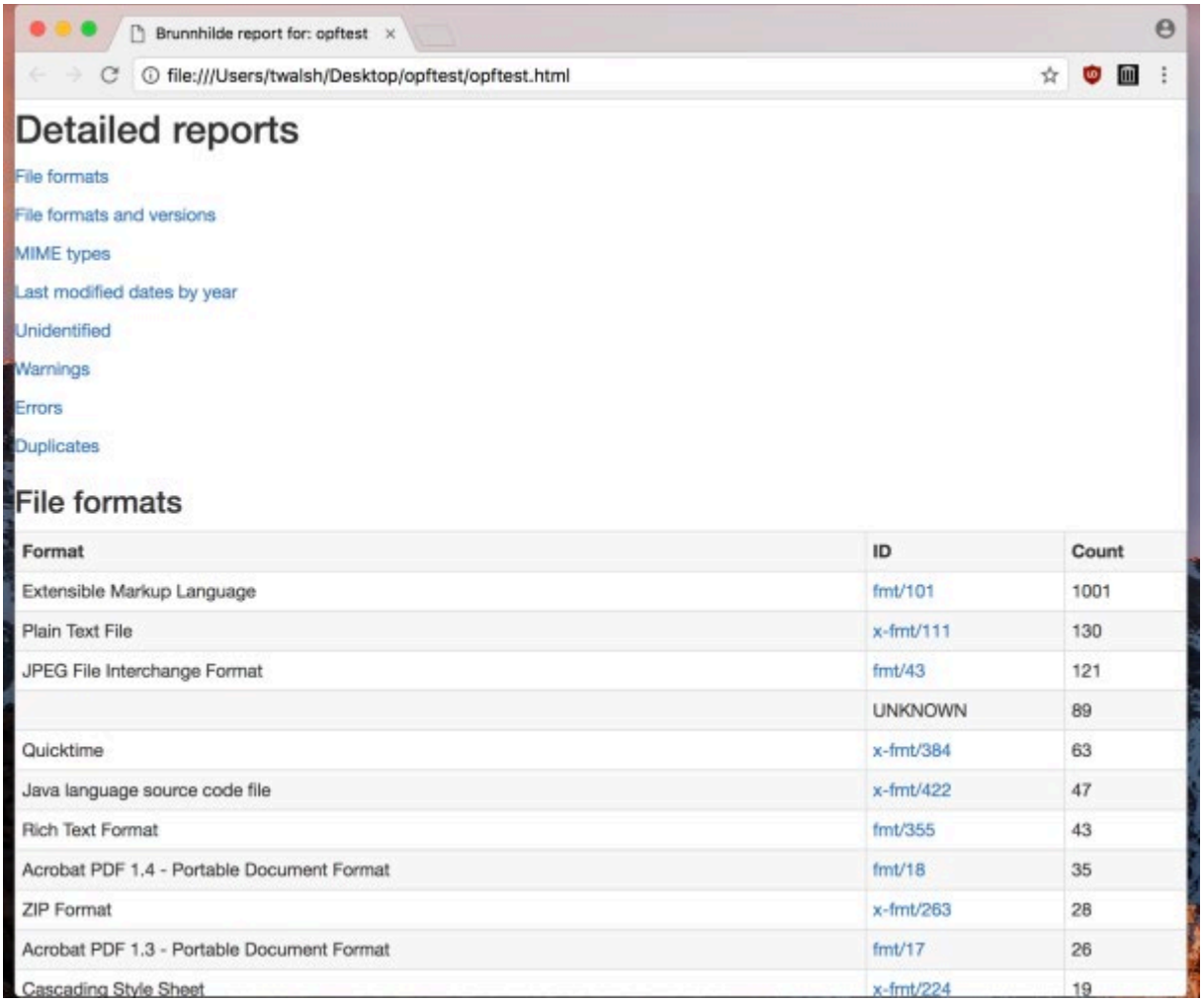

Figure 5: HTML report – Detailed reports

These reports give detailed information about file formats, file format versions, MIME types, and the years files were most commonly last modified; list all unidentified files, duplicate files, and files that triggered Siegfried warnings and errors; and, optionally, the results of a Bulk Extractor scan for personally identifiable information (PII). PUIDs (PRONOM unique identifiers for file format types) in these reports are hyperlinked so that the end user can click on, for example, “fmt/43” and be taken to the PRONOM page on the UK National Archives’ website that describes the format in question.

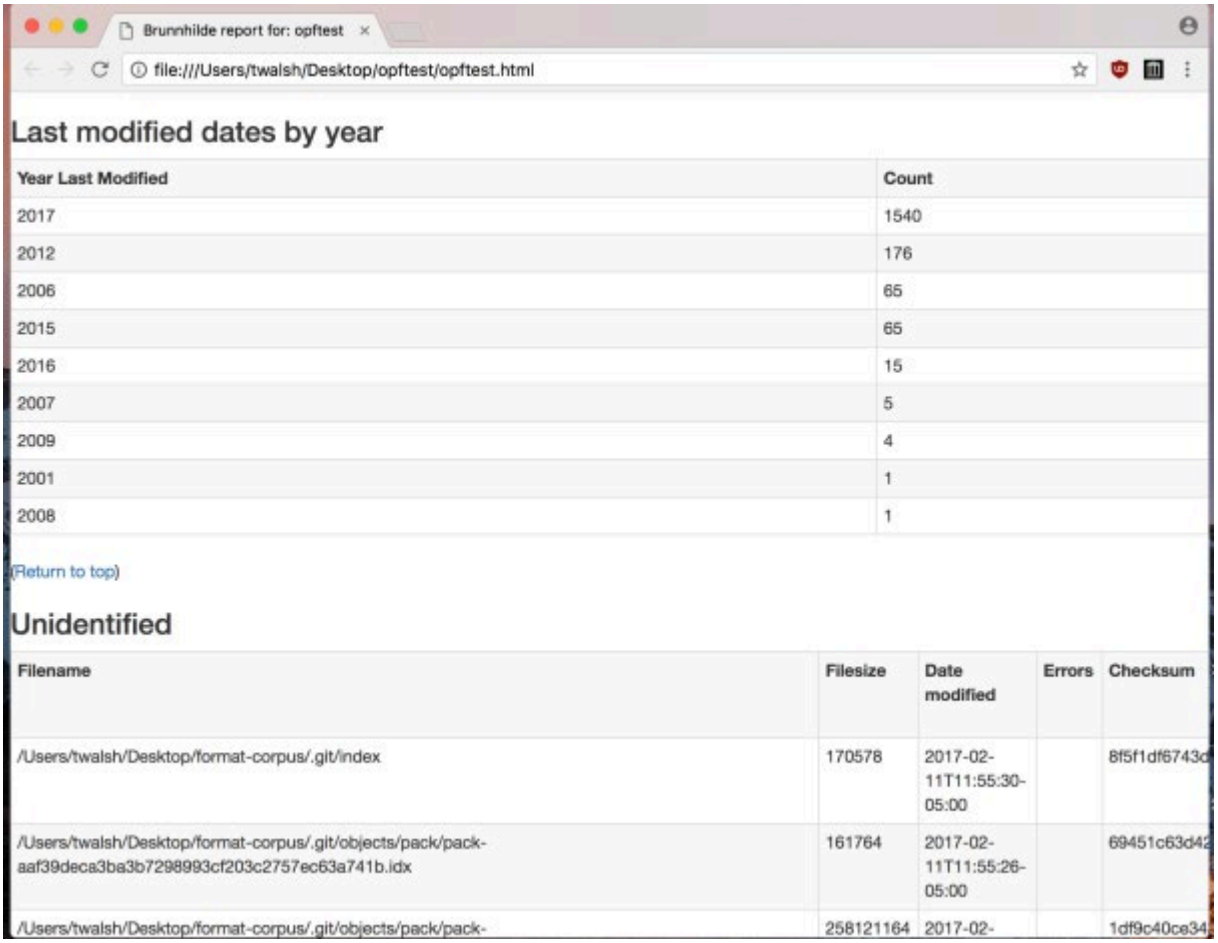

Figure 6: HTML report – Detailed reports (continued)

Use Cases

To illustrate how Brunnhilde can help archivists make informed processing decisions for born-digital materials, it may be helpful to consider two possible use cases.

Use Case #1: Removable media in a personal papers collection

Repository A has the Jane Doe papers, a personal paper collection that contains, among papers and records in other formats, a Zip disk found within a folder containing correspondence between Doe and her acquaintances. Based on the disk's label and context alone, archivists at Repository A are unable to determine the disk's contents or whether the disk contains any sensitive information that should be redacted prior to end user access online or in the reading room.

As part of a larger digital preservation effort, the repository has already created a disk image of the Zip disk using a tool such as Guymager[19] (in BitCurator) or FTK Imager[20] and is now considering how to move on to the next step of appraising, arranging, and describing the disk within the context of its containing folder and

the collection as a whole. A copy of the disk image exists on the desktop of a BitCurator machine at the path /home/bcadmin/Desktop/JaneDoe/DoeZip001.E01.

Archivists at repository A could run Brunnhilde against the disk image using the simple-to-operate GUI, making sure to select “Run bulk extractor” from the available options:

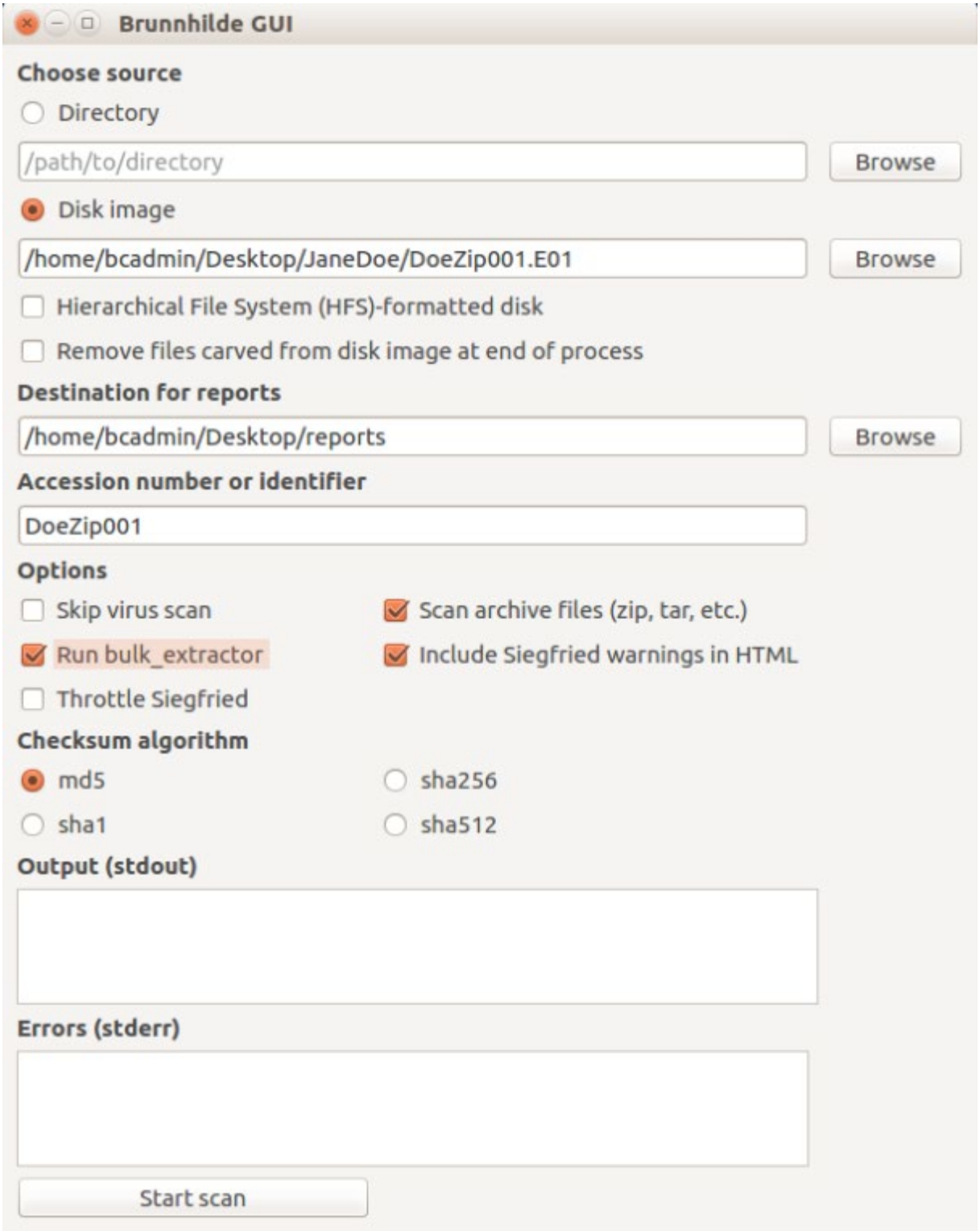

Figure 7: Brunnhilde GUI

Alternatively, the archivists could use the Brunnhilde command-line utility, entering the following command into a terminal:

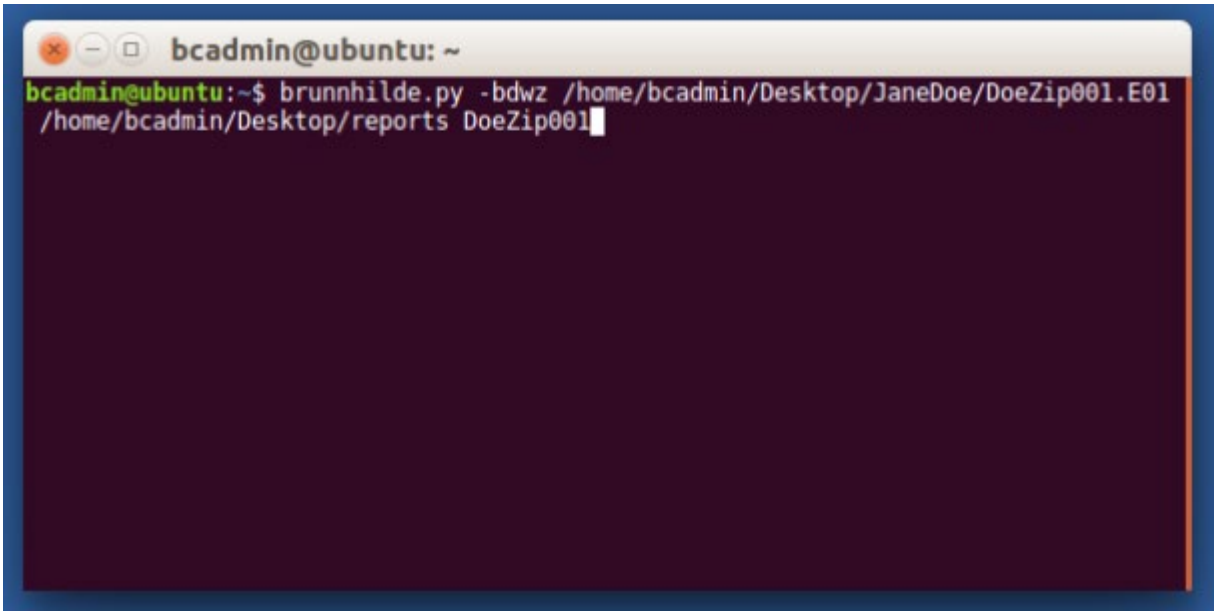

Figure 8: Brunnhilde CLI terminal command

This command invokes four optional flags:

- -b: Tells Brunnhilde to run Bulk Extractor against exported files
- -d: Tells Brunnhilde the source is a disk image
- -w: Tells Brunnhilde to include detailed information about Siegfried warnings in the HTML report
- -z: Tells Brunnhilde to have Siegfried decompress and analyze any archive files (e.g., .zip, .tar) it may find on the disk

Along with a copy of the exported files, the archivists will now have an HTML report that summarizes basic information about the contents of the disk such as the number of files and size of the contents on the disk, the years files were last modified, file formats and versions, and the results of a virus scan; provides a list of files that may require additional work such as adding or fixing file extensions; and lists potential matches for social security numbers and other potential personally identifiable information on the disk.

For a more detailed look into any potential personally identifiable information (PII), archivists will only need to consult the “bulkextractor” reports directory. The logs in the “bulkextractor” directory can also be fed as input into Bulk Extractor Viewer (BEViewer)[21] in BitCurator, a tool which enables easier browsing through results.

Based on this information, the archivists can choose to either leave the files as-is and give them a file-level description using the date and extent information generated by Brunnhilde or create an informed plan of action for more involved processing.

Use Case #2: Large accession of files from external drive/network transfer

Repository B has received a large file transfer containing a number of top-level directories, each with its own organizational structure as the records were kept by the donor, Non-Profit Org. Along with the file transfer, the archivists received some basic accession-level information about the transfer, but do not yet have much information about the various directories that together make up the accession.

In this case, Brunnhilde can be used in two stages: in the first, to supplement the accession record with more detailed information about the total contents of the transfer; and in the second, to gather information about each of the respective top-level directories for use in arrangement and description.

The first step looks just like Use Case #1. Using the GUI or command-line utility, the archivists at Repository B can run Brunnhilde against the transfer as a whole to gather high-level information about the transfer.

For the second step, archivists could use the Brunnhilde GUI to manually run Brunnhilde against each individual directory of interest. Depending on the number of directories included in the transfer, this could be a time-consuming task.

More efficiently, the archivists could use a simple bash script to call the Brunnhilde command-line utility against each directory at a given level in the transfer, saving the results for each directory as a new folder in a “reports” directory on the desktop for consultation in the triage and processing planning phase of processing. Such a bash script (written to be used with Brunnhilde 1.5.0) might look like this:

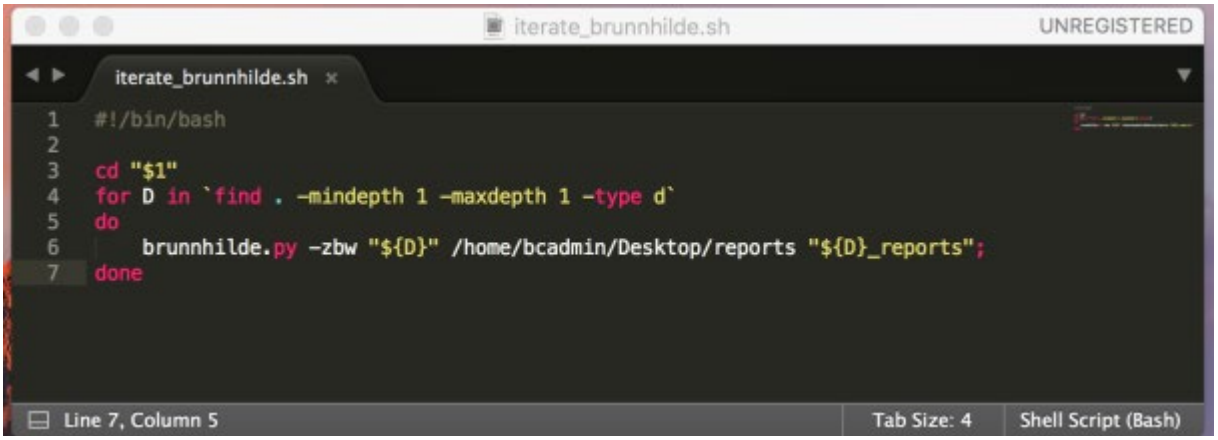

Figure 9: Example Bash script "iterate_brunnhilde.sh"

Taking this one step further, if Repository B had already decided based on existing policies and procedures to give each top-level directory a file level description in a finding aid, archivists could call the Brunnhilde command-line utility from another tool during processing itself.

For example, the Folder Processor tool developed for use in processing at CCA allows users to select any number of directories as input and creates a ready-to-ingest SIP packaged according to Archivematica's specifications for each supplied input directory.

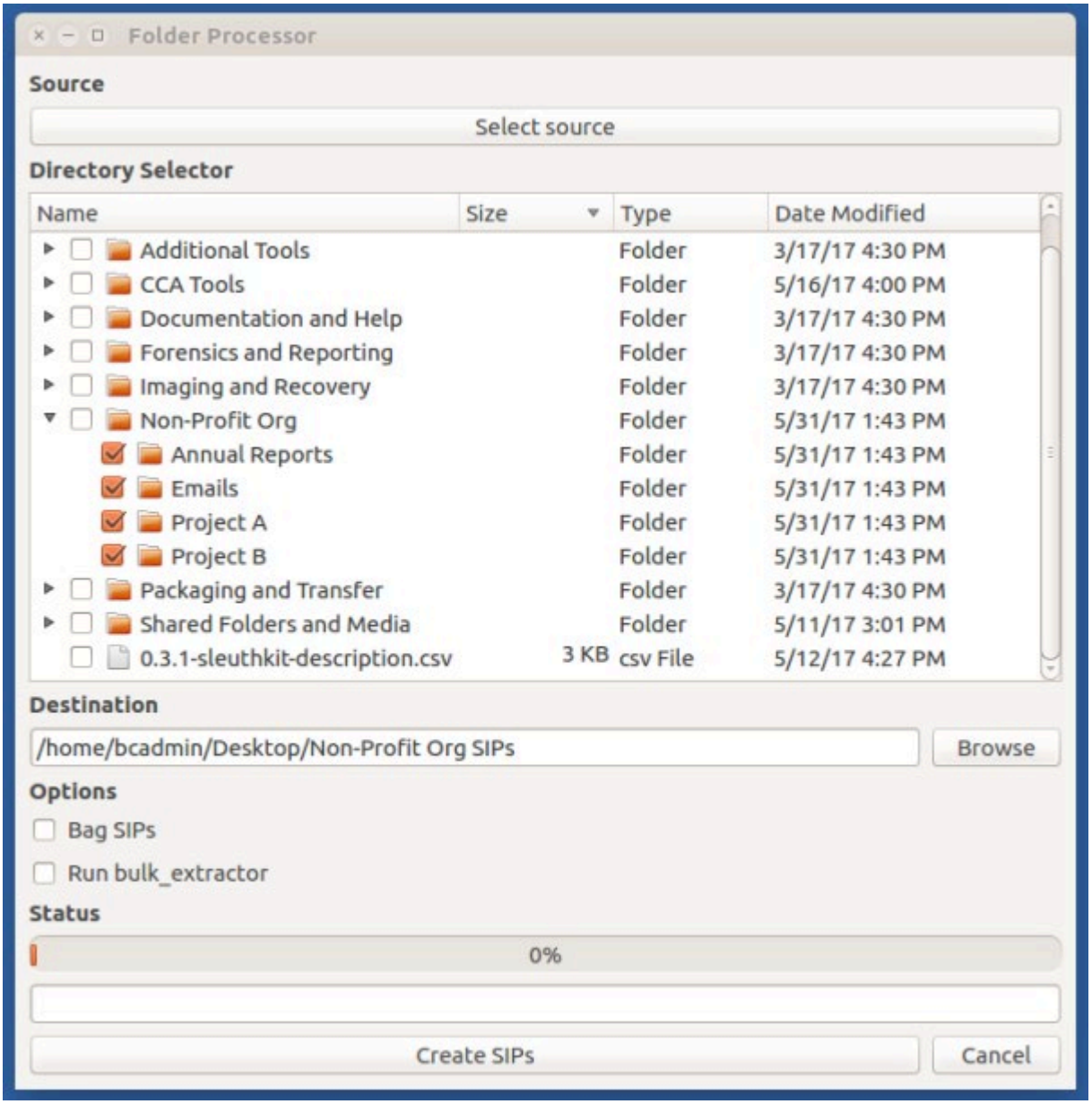

Figure 10: CCA Folder Processor

Using the Folder Processor with the settings as in the picture above, archivists would have a ready-to-ingest SIP for each top-level directory in the file transfer, a corresponding Brunnhilde report and DFXML file for each SIP[22] an md5 checksum manifest (the user can choose for SIPs to be bagged instead), and an auto-populated description spreadsheet with information on each target directory.

In addition to being included in the "submissionDocumentation" directory of each SIP, the DFXML and Brunnhilde outputs are used internally within the Folder Processor tool to populate the description CSV file with information such as start and end date, extent, and common file formats.

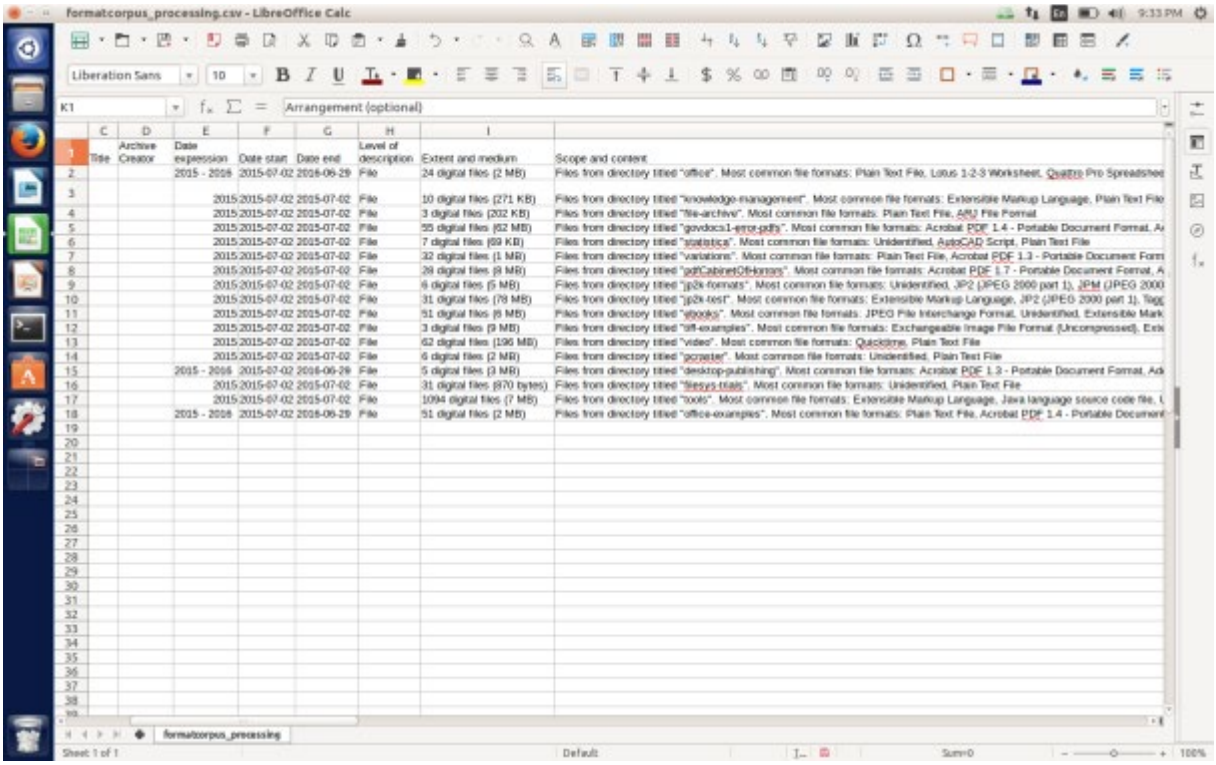

Figure 11: Description CSV generated by Folder Processor

The Folder Processor approach is just one way in which archivists can use Brunnhilde to support and automate aspects of processing born-digital archives. This type of automation and level of information allows archivists to spend more time focusing on the unique skills that they bring to the table – contextualization, arrangement, description, and end-user services – and less time browsing through files in an accession and manually entering factual information such as extent statements.

Conclusion

Brunnhilde was developed to meet a perceived need in the tools available for archivists developing and refining their workflows for processing born-digital records. It is the author's sincere hope that this tool will prove as useful at other repositories as it has at CCA, and inspire other archivists to similarly create the tools that are missing in the current born-digital archives landscape.

As free and open source software, Brunnhilde is yours to play with, implement, fork, and hack on as much as you want. Have fun, good luck, and please send any suggestions and other feedback to the author by email or by opening an issue in the Brunnhilde[23]or Brunnhilde GUI[24] Github repositories.

About the author

Tim Walsh is the Digital Archivist at the Canadian Centre for Architecture (CCA) in Montreal, Quebec. Among other responsibilities, Tim is responsible for developing and supervising CCA's processing workflows for born-digital archives and managing CCA's Archivematica-based digital repository. He holds an MS in Library and Information Science from Simmons College and a BA in English from the University of Florida. Tim is a firm believer in free and open source software and in the benefits of sharing workflows and tools.

Notes:

[1] "File profiling tool (DROID)," The National Archives, accessed September 29, 2016, http://www.nationalarchives.gov.uk/information-management/manage-information/policy-process/digital-continuity/file-profiling-tool-droid/.

[2] "Siegfried," Github, accessed September 29, 2016, https://github.com/richardlehane/siegfried.

[3] "Apache Tika," The Apache Software Foundation, accessed September 29, 2016, https://tika.apache.org/.

[4] "The technical registry – PRONOM," The National Archives, accessed September 29, 2016, http://www.nationalarchives.gov.uk/PRONOM/Default.aspx.

[5] "About," Canadian Centre for Architecture, accessed September 29, 2016, http://www.cca.qc.ca/en/about.

[6] BitCurator, accessed September 29, 2016, http://www.BitCurator.net/.

[7] "Category: Digital Forensics XML," Forensics Wiki, accessed September 29, 2016, http://www.forensicswiki.org/wiki/Category:Digital_Forensics_XML.

[8] "Bulk extractor," Forensics Wiki, accessed September 29, 2016, http://www.forensicswiki.org/wiki/Bulk_extractor.

[9] Sleuth Kit, accessed September 29, 2016, http://sleuthkit.org/.

[10] "droid-siegfried-sqlite-analysis-engine," Github, accessed September 29, 2016, https://github.com/exponential-decay/droid-siegfried-sqlite-analysis-engine.

[11] It should be noted that since the author began working on Brunnhilde, Spencer's project has expanded to use Siegfried in addition to DROID.

[12] "tsk_recover man page," accessed September 29, 2016, http://www.sleuthkit.org/sleuthkit/man/tsk_recover.html.

[13] "HFSExplorer," Catacombae, accessed September 29, 2016, http://www.catacombae.org/hfsexplorer/.

[14] ClamAV, accessed September 29, 2016, http://www.clamav.net/.

[15] SQLite, accessed September 29, 2016, https://www.sqlite.org/.

[16] "tree(1) – Linux man page," accessed September 29, 2016, https://linux.die.net/man/1/tree.

[17] "CCA Disk Image Processor," Github, accessed May 31, 2017, https://github.com/timothyryanwalsh/cca-diskimageprocessor.

[18] "CCA Folder Processor," Github, accessed May 31, 2017, https://github.com/timothyryanwalsh/cca-folderprocessor.

[19] "Guymager," Sourceforge, accessed February 11, 2017, http://guymager.sourceforge.net/.

[20] "FTK Imager 3.4.2," AccessData, accessed September 29, 2016, http://marketing.accessdata.com/ftkimager3.4.2.

[21] "Bulk Extractor Viewer," Forensics Wiki, accessed June 2, 2017, http://www.forensicswiki.org/wiki/Bulk_Extractor_Viewer.

[22] Examples here use the Open Preserve Format Corpus. "format-corpus," Github, accessed September 29, 2016, https://github.com/openpreserve/format-corpus.

[23] "Brunnhilde," Github, accessed February 11, 2017, https://github.com/timothyryanwalsh/brunnhilde.

[24] "Brunnhilde GUI," Github, accessed February 11, 2017, https://github.com/timothyryanwalsh/brunnhilde-gui.