REPLICATING DIGITAL AUDIOVISUAL ARCHIVES IN THE CLOUD: THE UNINTENDED CONSEQUENCES OF NON-ALPHANUMERIC CHARACTERS IN FILENAMES AND FILEPATHS

Joe Stolarick, Digital Systems Lead, New Orleans Jazz & Heritage Foundation Archive, New Orleans, USA

Abstract

Non-alphanumeric characters in filenames and file paths can lead to unintended consequences in digital preservation. The New Orleans Jazz & Heritage Foundation Archive conducted simple tests to investigate peculiarities related to this issue that surfaced when preserving their audiovisual assets. This case study resulted in a list of problematic, albeit commonly used, non-alphanumeric characters. It also uncovered significant implications for using proprietary cloud replication software as part of digital preservation workflows.

KEYWORDS: digital preservation, character encoding, cloud storage, case study

Introduction

Character encoding and its impact on digital preservation is a complex issue that has resulted in a small, albeit informative body of literature, primarily in the form of journal publications and blog posts. Most of the writings I reviewed in preparation for this report focus on the basics: introducing key standards (e.g., ASCII, Unicode) and the complications of identifying and converting plain text formats. *Plain Text & Character Encoding: A Primer for Data Curators by Seth Erickson* (2021) provides an excellent introduction to character encoding as it relates to digital preservation. Dig a little deeper and one will inevitably find more ethically and culturally-grounded conversations surrounding the implications of diacritic handling, examining the "racial and Western-centric implications of considering certain characters to be 'illegal'" (Blewer, 2019).

Rather than spend time retreading this territory, I designed a simple case study addressing observations made during my personal digital preservation workflow. Recently, I discovered that files being copied from file system to file system (e.g., Macbook to Synology NAS, Synology NAS to Backblaze B2 cloud storage), were quietly failing to replicate correctly: or not replicating at all. The common thread among files that fail is non-alphanumeric characters in the filename. Additionally, the failures were not always consistent; for example, a file that reached one destination was not guaranteed to reach another. To keep this report grounded in practical application, I only address issues observed within specific systems that are part of my personal digital preservation workflow. It is my hope that these findings will be of use to other non-profits and repositories that may be establishing and managing their own digital preservation programs. For those interested in a much deeper dive, the resources are out there. For instance, *The Big List of Naughty Strings* (Woolf, n.d.) is intended as a quality assurance tool for programmers.

Before we discuss my specific workflow, here is a bit of context. In August 2020, I was hired as the Digital Systems Lead for the New Orleans Jazz & Heritage Foundation with the generous support of The Helis Foundation. I am tasked with implementing a digital preservation program for the New Orleans Jazz & Heritage Foundation Archive (referenced from this point forward as the "Jazz & Heritage Archive'' or "Archive"). The Archive, located in a small Creole cottage in the French Quarter, houses records and materials relevant to the Foundation's history as well as its assets, most notably the New Orleans Jazz & Heritage Festival. I am one of two full-time employees in the Archive. In addition to our primary location, we have six off-site storage locations throughout the

country. Our digital preservation program is modeled on assessments and recommendations made by AVP, a digital preservation consulting firm. The Digital Systems Lead position was created in response to AVP's *Preservation Assessment & Planning Project Report* (or the "AVP Plan"), conducted in 2019.

Digital preservation workflow

My first task outlined in the AVP Plan was to "migrate data from external sources into managed storage." To do so, a custom bash script was used to create "snapshot" csv files, recording the file path, filename, and file size of each digital object stored on an external carrier. Md5 checksum values were generated for each file and all metadata imported to a MySQL database. The drive contents were then copied to our network attached storage device (NAS). Once migrated, the data was evaluated for fixity and attendance using the original checksums and newly generated checksums for files as stored on the NAS. Following checksum verification, the data was copied to two separate cloud storage vendors using NAS cloud replication software to have "more than two copies... of digital content stored in separate geographic locations," another AVP Plan recommendation.

To accomplish this workflow, hard drives were mounted via a TOTU Type C Hub to a MacBook Pro laptop running macOS 10.15.7. Occasionally, non-alphanumeric characters in file names would cause issues with delimiting or script execution (both snapshot and checksum scripts), at which point I would be required to amend file names. If this was done on macOS, it was performed manually or using the open-source program detox[1]. If the drive was formatted NTFS or otherwise restricted on my MacBook Pro, I mounted it on a Dell desktop running Windows 10 Pro (10.0.19043 Build 19043) to make changes manually or using a program called NB Renamer[2]. NB Renamer behaves similarly to the "replace" function used to rename files in macOS. Once all changes were complete, the drive was returned to the MacBook Pro to complete the workflow.

Despite my best efforts to remove potentially problematic characters, some files were copied to our Synology NAS (DSM 6.2.4-25556 Update 2) without sufficient changes. If no red flags were raised by the checksum verification process, the NAS's Cloud Sync software would attempt to replicate these files with problematic characters in their names or paths to their respective buckets in cloud storage. For cloud storage, we have accounts with both Wasabi and Backblaze B2. It was not until I began scrutinizing our data from within these accounts that I noticed some peculiarities.

Currently, Synology Cloud Sync replicates one in five shares on a staggered, bi-weekly schedule (i.e., each share updates twice per week). In total, our NAS is storing approximately 65 TB of data. To monitor our cloud storage data, I use rclone[3], a command line tool for monitoring and managing files in cloud storage. Rclone prints csv reports that I can compare with up-to-date versions of our hard drive snapshot data stored in our MySQL database. Comparisons showed that by and large, the files not being replicated contained non-alphanumeric characters in file names. However, the failures were not uniform among cloud providers (i.e., some files that were replicated to B2 were missing from Wasabi).

1 detox available at: https://linux.die.net/man/1/detox
2 NB Renamer available at: https://hermit99.github.io/nbrenamer/
3 Rclone available at: https://rclone.org/

My first step in investigating this issue found me at both the Wasabi and Backblaze websites in search of any literature regarding known issues regarding file handling and non-alphanumeric characters. On Wasabi's FAQ page, it states that Wasabi does not support non-ASCII characters that are 4-byte UTF8 characters:

> "Certain files may have non-ASCII characters that are 4 byte UTF8 characters (such as emojis) in the filename. Wasabi does not support these characters and will return a 400 error message to an application that tries to write these files with 4 byte UTF characters in the file name. We recommend renaming the affected files if possible" (Wasabi, n.d.).

Similarly, Backblaze lists the following guidelines for file names on their website:

> "Names can be pretty much any UTF-8 string up to 1024 bytes long. There are a few picky rules:
> - No character codes below 32 are allowed.
> - DEL characters (127) are not allowed"
> (Backblaze, n.d.)."

For someone not entirely fluent in the language of character encoding, these policies were not helpful for predicting which non-alphanumeric characters might cause problems with data replication. To better understand the limitations within my own personal workflow, I decided to design my own study.

Case study

To begin, I first compiled a dataset consisting of thirty-one text (.txt) files created in TextEdit (Version 1.15). Each file name had a unique, singular, non-alphanumeric character inserted between the "e" and "s" in the word "test" (e.g. "te!st.txt", "te/st.txt", etc.). The non-alphanumeric characters selected were based on their inclusion on our MacBook Pro's keyboard. I intentionally excluded diacritics from this study simply because they were absent from the filenames of files that failed to replicate. Once the thirty-one text files were created, I moved them to a folder titled "cloud-filename-tests" on the MacBook Pro's desktop.

Before I even finished creating my dataset, I ran into an issue. MacOS will not allow colons (:) in file names, automatically changing any instance of a colon to a hyphen (-). This is because the macOS Extended File System (HFS+) uses colons (:), not slashes (/), as path separators (Sánchez, 2000). Therefore, "te:st" was dropped from my dataset, due to the inability to create it (though it comes back to haunt me later).

The next step in the workflow was getting my files to our NAS device. While attempting a drag-and-drop copy of the dataset to the NAS device, I received an error message from Synology stating:

"Failed to upload 'cloud-filename-tests'. File and folder names cannot contain colons (:) and slashes (/), start with ._ (e.g., ._name), or use any combination of characters that are reserved for system use (e.g., . or ..). Please enter another name."

As a result, the file containing the forward slash (/) was not copied and dropped from my dataset. Similarly, the test file with double quote (te”st.txt) was copied, but its name was changed to “te%22st.txt”. It seems likely that there is a discrepancy inherent in the file systems’ data transformation logic that is misinterpreting the double quotes. Unlike the colon-turned-hyphen, I decided to leave this file as “te%22st.txt” for the remainder of the study to achieve a more linear result.

Now that the files were on the NAS, I needed to create two replication jobs in Cloud Sync. Each job would replicate the “cloud-filename-tests” folder to test buckets in both Wasabi and Backblaze B2. Once the proper connections were made to their destination folders within Cloud Sync, I ran both jobs, which completed with no *reported* errors (Fig. 1).

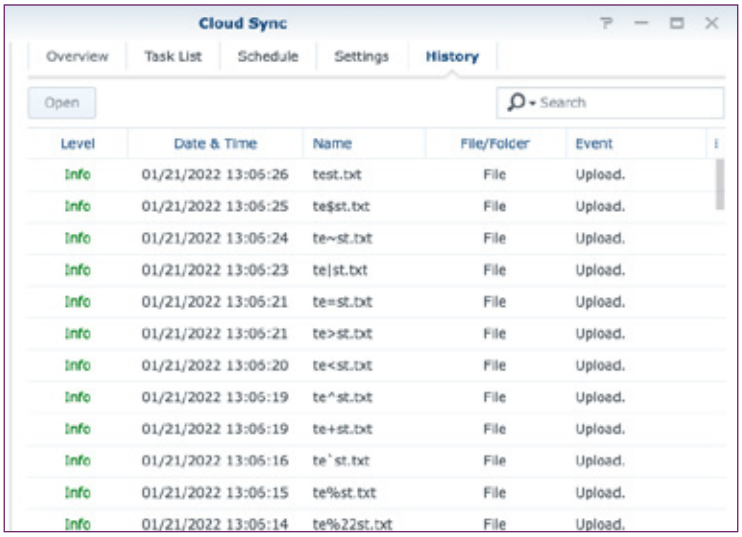

Fig. 1. Screenshot of Synology Cloud Sync

In fact, there were quite a few problems, particularly with the replications to Wasabi cloud storage. Only 15 of 29 files were copied. The files that failed to replicate had the following non-alphanumeric characters in the file name: close bracket (]), double quote (“), close brace (}), open bracket ([), open brace ({), acute (`), pipe (|), caret (^), hash (#), tilde (~), backslash (\), percent (%), greater than (>), and less than (<).

Replications to Backblaze B2, on the other hand, had a much higher success rate with only one file not copying (“te\st.txt”). I also recorded that “te%22st.txt” copied, but noted that it maintained its altered file name (Table 1).

Character Name	Char	Wasabi	B2
plus	+	COPIED	COPIED
close parenthesis	)	COPIED	COPIED
open parenthesis	(	COPIED	COPIED
asterisk	*	COPIED	COPIED
close bracket	]	FAILED	COPIED
semi colon	;	COPIED	COPIED
double quote	"	FAILED	COPIED AS "te%22st.txt"
close brace	}	FAILED	COPIED
equal	=	COPIED	COPIED
ampersand	&	COPIED	COPIED
ampersat	@	COPIED	COPIED
open bracket	[	FAILED	COPIED
open brace	{	FAILED	COPIED
acute	`	FAILED	COPIED
question mark	?	COPIED	COPIED
pipe	|	FAILED	COPIED
caret	^	FAILED	COPIED
hash	#	FAILED	COPIED
colon	:	NOT IN DATASET	NOT IN DATASET
exclamation	!	COPIED	COPIED
tilde	~	FAILED	COPIED
backslash	\	FAILED	FAILED
percent	%	FAILED	COPIED
greater than	>	FAILED	COPIED
apostrophe	'	COPIED	COPIED
less than	<	FAILED	COPIED
dollar sign	$	COPIED	COPIED
hyphen	-	COPIED	COPIED
underscore	_	COPIED	COPIED
forward slash	/	NOT IN DATASET	NOT IN DATASET
comma	,	COPIED	COPIED

Table 1. Synology NAS to cloud storage test results.

Having tracked each point of the workflow, I decided to attempt this same experiment on my personal NAS, a QNAP TS-251B (Firmware: 5.0.0.1891), primarily to get a better sense of whether Synology and its CloudSync software could be having any effect on the file transfers. Almost immediately, I saw a difference in that the QNAP allowed me to copy all 30 files with only one filename change; the file "te/st.txt" was changed to "te:st.txt". As with the Macbook Pro, it is likely that QNAP's file system uses the forward slash as a path separator. Regardless, this change marked the triumphant return of the colon (:), which resulted in some momentary confusion.

As with Cloud Sync, I next created backup jobs in QNAP's Hyper Backup Sync (HBS 3) and established the proper connections with new test buckets in both Wasabi and Backblaze B2. After running both jobs without errors, I checked both QNAP test buckets and found that the replications done via QNAP were far more successful within Wasabi. In fact, every file copied successfully. Backblaze B2, while largely successful, did appear to have a character encoding issue with backslash (\), which was copied as "teXA==st.txt" (Table 2).

Character Name	Char	Wasabi	B2
plus	+	COPIED	COPIED
close parenthesis	)	COPIED	COPIED
open parenthesis	(	COPIED	COPIED
asterisk	*	COPIED	COPIED
close bracket	]	COPIED	COPIED
semi colon	;	COPIED	COPIED
quote	"	COPIED	COPIED
close brace	}	COPIED	COPIED
equal	=	COPIED	COPIED
ampersand	&	COPIED	COPIED
ampersat	@	COPIED	COPIED
open bracket	[	COPIED	COPIED
open brace	{	COPIED	COPIED
acute	`	COPIED	COPIED
question mark	?	COPIED	COPIED
pipe	|	COPIED	COPIED
caret	^	COPIED	COPIED
hash	#	COPIED	COPIED
colon	:	NOT IN DATASET	NOT IN DATASET
exclamation	!	COPIED	COPIED
tilde	~	COPIED	COPIED
backslash	\	COPIED	COPIED AS "teXA==st.txt"
percent	%	COPIED	COPIED
greater than	>	COPIED	COPIED
apostrophe	'	COPIED	COPIED
less than	<	COPIED	COPIED
dollar sign	$	COPIED	COPIED
hyphen	-	COPIED	COPIED
underscore	_	COPIED	COPIED
forward slash	/	COPIED AS (:)	COPIED AS (:)
comma	,	COPIED	COPIED

Table 2. QNAP NAS to cloud storage test results.

Conclusion

At the beginning of this study, I anticipated running into issues when moving files from system to system (e.g., from MacBook to NAS, NAS to cloud, etc.). However, I did not anticipate the high degree to which proprietary software used to replicate the data (i.e., Cloud Sync and HBS 3) could hinder my digital preservation workflow. While files were created and copied with relative ease in the MacBook and Synology/QNAP file systems, there were significant errors when replicated to cloud storage in the form of dropped files and unexpected file name changes. Most of these errors occurred when files were copied via Synology's Cloud Sync from the NAS to Wasabi cloud storage. In contrast, files on the QNAP NAS were copied to Wasabi with virtually no errors.

To conclude, this study is intended to investigate and address a specific problem in the Jazz & Heritage Archive's digital preservation workflow, not to examine or analyze the behavior of character encoding and data transformation at a technical level. The information presented here is certainly worth considering when planning a data migration to network attached storage and/or cloud storage. These results indicate the benefits of using a QNAP NAS, whose software performed far better with data transfers than that of our Synology NAS. However, the most salient point resulting from this case study is that the use of non-alphanumeric characters and diacritical markings should be avoided in file and/or folder names. The author understands that it is not always possible or desirable to change file/folder names in some archival circumstances. But when an organization has the option to determine file and folder names, it is strongly recommended that only letters (A-Z), numbers (0-9), hyphens (-) and underscores (_) are used. Trial and error have been ever present in my work building a digital preservation system. It is my hope that, in reading this, others might be able to bypass at least one of those errors.

References

Backblaze, (n.d.). Files. Available at: https://www.backblaze.com/b2/docs/files.html [Accessed 24 April 2022].

Blewer, A., 2019. Artist_Exhibition-copy (FINAL)(2).mov: Preserving diacritics in filenames as significant properties in media conservation. [Blog] Available at: https://bits.ashleyblewer.com/blog/2019/06/17/artist-exibition-copy-final-2-preserving-diacritics-in-filenames-as-significant-properties-in-media-conservation/ [Accessed 24 April 2022].

Erickson S., 2021. Plain Text & Character Encoding: A Primer for Data Curators. *Journal of eScience Librarianship* 2021; 10(3): e1211. https://doi.org/10.7191/jeslib.2021.1211. Available at: https://escholarship.umassmed.edu/cgi/viewcontent.cgi?article=1211&context=jeslib [Accessed 24 April 2022].

Sánchez, W., 2000. The Challenges of Integrating the Unix and Mac OS Environments. In: *USENIX Annual Technical Conference*, Invited Talks Track. Available at: https://www.usenix.org/legacy/event/usenix2000/invitedtalks/sanchez_html/ [Accessed 24 April 2022].

Wasabi, (n.d.). Does Wasabi support 4 byte UTF8 characters? Available at: https://wasabi-support.zendesk.com/hc/en-us/articles/360002101712-Does-Wasabi-support-4-byte-UTF8-characters- [Accessed 24 April 2022].

Woolf, M., (n.d.) The Big List of Naughty Strings. GitHub. Available at: https://github.com/minimaxir/big-list-of-naughty-strings [Accessed 24 April 2022].