Renaming files with Bridge, the command prompt, or Python

When you come across a directory of files that need to be renamed in some uniform way, it can be tedious and time-consuming to rename each file individually. Manual renaming is error prone, and in the case of medium to large directories it's much slower than scripting the process. This guide explains the principles of renaming batches of files using Adobe Bridge, the Windows command prompt, and Python. Each method has strengths and weaknesses, and the right tool for a given job is typically whichever one you're most comfortable with. The goal with this kind of work is to reduce stress, not to increase it!

Adobe Bridge

Bridge is a free asset management tool released by Adobe that plugs in very nicely to various Adobe creative suite products. It has some useful tools that don't rely on any other software, too, such as the Batch Rename tool. On a surface level, Bridge works somewhat like Windows Explorer or Finder on OS X. You can navigate the same drives and folders you ordinarily see on your computer.

First, navigate to the folder containing the files you need to rename, and select the files in question. If you only need to rename a subset of files in a directory, and those files have some filename element or other attribute that sets them apart, you can run a search within Bridge to select only these files, excluding the others. Note that you will still need to *select* any files in the results of your search in order to use the batch rename tool.

When you've selected your files, click "Tools Batch rename...".

In the window that opens, you will be presented with a few options:

1. Destination Folder: Whether you'd like to rename files in place (overwriting the files you've selected), move them to a new folder with the new names you give them, or copy them to a new folder with the new names (leaving the originals in place).
2. New Filenames: The individual filename elements you'd like to use for the new filenames
3. Options: Whether you'd like to preserve the current filename in XMP metadata, for image files with an XMP sidecar

The most important of these is #2, because it's where you'll actually build and apply the new filenames. Bridge gives you several different ways of doing this, which you can combine to make your new filenames exactly the way you need them. You can select a specific naming method using the drop down on the left of the New Filenames section, and add or remove them using the + and - buttons on the right. The naming methods you select from top to bottom will build the filenames from left to write. Here are the methods:

- Text: Input simple text that will appear in the filename. For example, if every file in a directory needs to contain the collection name, type that name in here. The text field is also how you add underscores to separate filename elements
- New extension: If you need to change file extensions. This doesn't convert files (i.e. it won't create JPG derivatives from TIFFs), but it will change .txt files to .xml files, or .jpeg files to .jpg files.
- Current filename: Useful if you just want to add to the existing filename, or make it all lowercase, uppercase, etc. This method will bring in the existing filename for each file you've selected
- Preserved filename: Loads in a previous filename found in XMP metadata (see #3 above), otherwise does the same as the "Current filename" method
- Sequence number: Adds a number to each selected file, in the order the files appear in Bridge (so be sure you've selected them in the order you want). You can start at any number and use up to 6 digits, to match whatever naming convention you'd like.
- Sequence letter: The same principle as Sequence number, but using letters.
- Date time: Add the current date or time, or the time of creation or modification, in a variety of formats
- Metadata: Draw on EXIF metadata found in Raw or TIFF files. This one is probably not the most useful for archival purposes.
- Folder name: If the name of the folder containing the images is missing from the filenames, you can use this to insert it.
- String substitution: Search for a string of text found in the current filenames and replace that string with something else. This tool is particularly useful for files that have a typo or other error in their names but which don't necessarily need to be totally renamed from the ground up (changing genaro_garcia_p001, p002, p003 to gg_p001, p002, p003, e.g.)

As you add filename elements, the preview at the bottom of the window will update to give you an idea of what you're building. Remember to add text underscores in between elements.

Before you click "Rename", it's important to click "Preview" and look over the list of changes that will be made, to make sure file sequences are correctly preserved in the new naming template. ***If the order of files changes as a result of bulk renaming, it is exceptionally difficult to undo the change and fix the order.*** It's important to be sure before committing the change. It's also a very good idea to select the "Export to CSV" button in the Preview window, to save a record of the change. That CSV can be quickly manipulated into a txt file and saved as a readme.

Windows command prompt

For relatively quick and straightforward renaming tasks, the command prompt can be handy. The command prompt is less suited to inserting filename elements in the middle of a filename, or to adding sequence numbers, but if you need to add an element to the beginning or end of a set of files, it can do the job very quickly.

Open the command prompt and navigate to the directory containing the files. Remember that to change drives, you just type the drive letter followed by a colon (i.e. C: to navigate to your local hard drive). Use cd to change directories within a drive.

Once you have navigated to the directory in question, you will be using the dir command to produce a list of the files within that directory. Each filename in that list will be temporarily stored as a "token", which can be added to using other commands. In this case we'll be pairing dir with the ren (rename) command.

Let's imagine you have a set of TIFFs in the following directory:

C:\digitized_collections\bliss_david_papers

And you need to add the prefix "correspondence_" to each file. Navigating to the directory and running the following command will print a simple list of all the files in the directory:

```
dir /b
```

You can also run this command from another directory if you prefer (say, if you find the cd command tedious), if you add the full path to the directory:

```
dir /b C:\digitized_collections\bliss_david_papers
```

In these examples, you add the /b parameter to tell Windows to print *bare* output, stripped of timestamps, full directory paths, and byte counts that you don't need. Since you're working with filenames, you just need a simple list.

If the directory contained other files that you didn't want to include in the list, such as a readme file, you could specify a particular file format to be included:

```
dir /b *.tif C:\digitized_collections\bliss_david_papers
```

If there were a single file in that folder that you needed to rename, you could use the ren command to do this:

```
ren C:\digitized_collections\bliss_david_papers\file001.tif correspondence_file001.tif
```

The ren command expects to be followed by two items, separated by a space. The first is a file to be renamed, written as a full path. In this case, that's file001.tif. The second item is a new name for the file, which doesn't need a full path, which in this case is correspondence_file001.tif.

You can combine these two commands to rename every file in a given directory, but first you need to indicate that you want to do something repeatedly, once per item in a list. This is done using a "for loop", as in "for each item in a list, do X". You also have to specify a token, denoted using a % symbol, to be used as a stand-in for each item in the list, which in this case will be a filename. We will use this for loop and a token to add the "correspondence_" to each file name.

```
for /f %a in ('dir /b *.tif C:\digitized_collections\bliss_david_papers') do ren C:
\digitized_collections\bliss_david_papers\%a correspondence_%a
```

Here's a rough breakdown of what this command is doing:

1. The section before the parentheses is you telling the command line that you will be producing a list and asking it to do something once per item in that list. "for /f" tells the computer that there will be an input enclosed by parentheses. The %a that follows is the token, which can be another letter if you prefer, but which must start with a % symbol. You can see the same token used twice later in the command.
2. The section between the parentheses reads the directory and produces a list of TIFFs, with bare output. Instead of showing the list on screen, it passes each output from the dir command to the next command. Because we already specified %a as the token, the computer will read %a as referring to whatever file it has been pointed to at that moment. It does this once per file, very quickly.
3. The section after the parentheses uses the output from the dir command to do something, in this case renaming files. %a is used as a stand in for the file name in both parts: you're telling the computer to look for a given filename (which was found in the directory in the previous part) and to add "correspondence_" to it.

In this example the original file name is *added to* but not manipulated in any more advanced way. To do more advanced transformations, you're better off using Bridge or Python.

Python

For more complicated jobs, you can write a simple script in Python that will parse a file list and manipulate each item. You will need to install a Python 3 code editor (I use Spyder, installed as part of Anaconda) to write and execute your script.

There are many different ways of editing filenames using Python, and the method you use will depend on what your filenames are and what you need them to be. Replacing hyphens with underscores is a different process than adding a sequence number, for example.

In any case, there are a few tools you'll probably want to use no matter what your filenames look like:

- os library - https://docs.python.org/3/library/os.html
    - Import this at the beginning of your script with import os
- os.listdir
    - This is sort of like the dir command in the Windows command prompt. It produces a list of the contents of a directory.
- os.rename
    - This is sort of like the ren command in the Windows command prompt. You tell it which file you want to rename, then give it a new name.
- split - https://www.w3schools.com/python/ref_string_split.asp

    - This breaks a string up into a list of elements, split along a given character found in the original string. For example, split "david-bliss-p-100.tif" along the hyphen to make a list made up of "david", "bliss", "p", and "100.tif", which can then be manipulated or recombined.
- replace - https://www.w3schools.com/python/ref_string_replace.asp
    - This replaces all the instances of a character in a string with a different string. For example, replace the hyphens in "david-bliss-p-100.tif" with underscores to change the string to david_bliss_p_100.tif

While there are many ways of changing filenames, generally speaking most of them involve the following steps:

1. Reading a directory of files and storing their names as a list (using os.listdir)
2. Opening a for loop, to run through each item in the list
3. Manipulating each item in the list in the same way and saving the new name as a variable
4. Replacing the old filename with the new filename that was stored as a variable

For example, if a directory had TIFF files whose elements were separated by hyphens rather than underscores, and you needed to change all the files in that directory at once, you could do so very quickly with the following script:

```
import os

fileList = os.listdir('C:/digitized_collections/bliss_david_papers')
for fileName in fileList:
    newName = fileName.replace('-', '_')
    os.rename('C:/digitized_collections/bliss_david_papers/' + fileName, 'C:/digitized_collections
/bliss_david_papers/' + newName)
    print(fileName + ' changed to: ' + newName)
print('All done!')
```

This script produces a list of files at the given directory (stored as fileList), then for each existing file (fileName) in the list stores a new variable (newName) that replaces hyphens with underscores. Then, it renames the original file, replacing it with the new file name. Finally, it displays (prints) a short statement indicating what the filename was changed to. When every item in the list has been changed, the script prints "All done!" a single time.

Library Carpentry has a great introduction to Python course, with applied examples like this one that can help reinforce the content of the lessons: https://librarycarpentry.org/lc-python-intro/

When writing Python scripts, even simple processes like the one above can sometimes fail for reasons that are difficult to discern. A single missing or misplaced character can cause an otherwise functioning script to break completely, so when you're first getting started, it's often a good idea to collaborate on Python scripts and/or copy from scripts that you know work.