

Clean Incoming Folder Script

This program was created to automate several tasks TAs complete after downloading a new package sent to ILM by our Clients. Tasks include; reformatting package names, removing duplicate subfolders, running file_cleaner, unzipping files, and updating file permissions.

Setting Up

You can run the script directly from its location:

```
/lucas/ilm/dept/ra/mhartney/bin/scripts/python/ta_clean_folder/clean-incoming-folder
```

For ease, I would add to your aliases file under 'clean_folder'. Usage examples will show me running the program with this alias.

Copy below to .aliases file under your User:

```
a clean_folder "/dept/ra/mhartney/bin/scripts/python/ta_clean_folder/clean-incoming-folder"
```

Note: when adding a new alias, remember to run 'source .aliases' or open a new shell before running the program.

Usage Instructions

The syntax for running the program on the command line is as follows:

```
clean_folder <option> <folder name or path>
```

The script takes input of a folder name / directory, or a directory path. For passing a directory you will use '-d' and for path '-p'.

```
clean_folder -d <folder name>
clean_folder -p <folder path>
```

Arguments / options for program:

```
-h, --help          show this help message and exit
--dir DIR, -d DIR    Pass folder / directory to the program on command line. You MUST change
directory to parent folder of target folder...
--path PATH, -p PATH Passing full path to program to avoid changing directory to folder...
```

```
[lon047:: ] /home/mhartney/example_workspace
> clean_folder -d 'PKG - example folder/'
Checking folder name 'PKG - example folder' ...
Rename to 'example_folder' ? (y / n) █
```

```
[lon047:: ] /home/mhartney
> clean_folder -p /home/mhartney/example_workspace/'PKG - example folder'
Checking folder name 'PKG - example folder' ...
Rename to 'example_folder' ? (y / n) █
```

Another way to input folders is to use the tab method on the command line which will automatically add escape characters to the package name:

```
[lon047:: ] /home/mhartney
> clean_folder -p /home/mhartney/example_workspace/PKG\ -\ example\ folder
Checking folder name 'PKG - example folder' ...
Rename to 'example_folder' ? (y / n) █
```

In the above example, I typed in the path up until '/home/mhartney/example_workspace/', then I pressed TAB to get the rest of the package name.

Examples

Here's an example of how the program will typically look when you run it:

```
[lon047:: ] /home/mhartney/example_workspace
> clean_folder -d 'PKG - example folder/'
Checking folder name 'PKG - example folder' ...
Rename to 'example_folder' ? (y / n) y
Found subfolder '/lucas/ilm/home/mhartney/example_workspace/PKG - example folder/example_fold
Checking folder name 'example_folder' ...
Removing duplicate folder ...
/lucas/ilm/home/mhartney/example_workspace/example_folder
```

Here you can see the program has replaced all space with '_' and removed the 'PKG' from the name. Before renaming, the script checks if there is a duplicate folder inside that has the same name as the parent. Falcon often saves the folder inside a duplicate named folder, and TAs will usually need to remove this, but here the program will do it for you. Printed in green is the new output location.

```
/lucas/ilm/home/mhartney/example_workspace/example_folder
Run file_cleaner on above path? (y / n) n
```

Next you will be prompted to run file cleaner. This will run the below command on your input folder path:

```
/sww/sand/bin/file_cleaner -ar <input_path>
```

After this has run you have an option to check for zip files and unzip:

```
Unzip any archive files in path? (y / n) y
-----
ZIP file found: a_archive.zip

Output Folder: /lucas/ilm/home/mhartney/example_workspace/example_folder/a_archive_unzipped
Extracting 'file1.txt'
Extracting 'file3.txt'
Extracting 'file.txt'
Extracting 'folder1/'
Extraction complete.
Updated Permissions: /lucas/ilm/home/mhartney/example_workspace/example_folder/a_archive_unzipped/folder1
Printed 'a_archive.zip' contents to '/lucas/ilm/home/mhartney/example_workspace/example_folder/a_archive_zip_contents.t
```

The program will find the zip file, and unzip into a folder based on it's name. For example, 'example.zip' will be unzipped into 'example_unzipped/'.

Once extraction is complete, permissions for each file and folder will be updated so they are open. Normally when you unzip a file sent by client, the folders are locked.

The program will then print the contents of the zip file to a text file, so there is a record of the ZIP on disk.

```
Zip files have all been extracted. The program can check files have been
unzipped successfully, and then delete or you can choose to be prompted
before each file is deleted.
```

```
Do you want to be prompted before deleting? (y / n) y
```

```
/lucas/ilm/home/mhartney/example_workspace/example_folder/a_archive.zip
```

```
File number on disk '4' matches ZIP '4'.
```

```
Delete ZIP file? (y / n) y
```

```
ZIP file has now been deleted.
```

```
Zip Record 1: /lucas/ilm/home/mhartney/example_workspace/example_folder/a_archive_zip_contents.tx
```

You can either choose to let the script check if the unzip has been successful and delete or be prompted for each file before it deletes.

Note: The script will delete the original ZIP file **only after it has checked if it's been successfully unzipped**. The check is done by counting the files / folders listed on the first level of the ZIP file, and then comparing with the number of items on disk. If they match, it will delete the ZIP file. If the file count does not match, this is a sign it has not unzipped successfully, and it **won't delete the ZIP file**.

```
Copied text to clipboard: Client package 'example_folder' has finished downloading!
/lucas/ilm/home/mhartney/example_workspace/example_folder
```

Finally the script will copy the new download package path to the clipboard, along with the message 'Client package X has finished downloading!'. You can then directly paste this into the Download Jira ticket for Production like below:

▼ Activity

All


Comments

Work Log

History

Activity

▼

 Max Hartney added a comment - 09/Jan/25 9:38 AM

Client package 'TO_ILM_250108D_ILM_LON_Review_Recording' has finished downloading!
/lucas/ilm/show/saga/staging/incoming/from_client/2025/2025.01/2025.01.08/TO_ILM_250108D_ILM_LON_Review_Recording

Folder Name Processing

The program will edit the folder name, and will also clean the subfolder name as well. This is the order in which it will clean a folder name, so you know what is being edited when you run it:

1. **Removes extensions** by iterating over a list (titled "extensions_list.json" in the clean_resources folder) and removing extension if found in basename.
2. **Removes 'PKG'** from folder name. This is added by Falcon to most folders, so should be removed, so the name on disk matches what we have on Aspera.
3. **Replaces any empty space characters, also '-' and '.' with '_'**. I wanted to replace these characters with an underscore and not just remove because they will often be used as a separator in text, and the folder name won't make sense or become confusing if you remove.
4. **Replaces '&' and '+' characters with '_and_'**. Again trying to replace special characters but keep the original meaning / intention in the folder name.
5. **Remove all special characters** that aren't letters or numbers.
6. **Ensure correct spacing** by not allowing more than 1 underscore to appear consecutively. For example, 'my__package_for_ilm' will change to 'my_package_for_ilm'.
7. Ensures that a package name **will not start or finish with an underscore**.

The above edits are all made in the above sequence.

```
> clean_folder -d PKG\ -\ my\ example\$\%^ +\ testf\*(\ folder\ \&\ other\ files____.t
Checking folder name 'PKG - my example$%^ + testf*( folder & other files____.txt' ...
Rename to 'my_example_and_test_folder_and_other_files' ? (y / n) █
```

