**Bash Script Orientation (3/30/2018)**

**General overview:**

These scripts run a sequence of terminal commands (e.g. make a new folder) and a series of tools via the command line (e.g. bagit) to automate the process of preparing a group of folders for ingest into the ARCHive. The commands are split into two scripts to allow staff to add metadata files after the AIP folders are in the right structure (aip-structure\_script) and before doing the rest of the processing (aip-finish\_script). The scripts work with Mac or Linux.

**Scripts we are using:**

* aip-structure\_script:
  + makes objects and metadata subfolders, moves content into objects subfolder
  + one argument: source directory (folder that contains the AIPs)
* aip-finish\_script:
  + creates master.xml from FITS, organizes metadata, bags, tar/zips, makes manifest
  + two arguments: source directory and department (ARCHive group-id)

**Common components of the scripts:**

**For loops**

The commands inside the loop are done to each of the folders or files selected by that loop. Usually this is the AIP folders, although it can be a specific document type such as FITS XML.

Loops may contain commands for running a tool (e.g. bagit.py) and/or terminal commands (e.g. mkdir). One loop may contain multiple commands which are run in sequence on the selected items. Loops may also contain if tests to only act on a portion of the selected items. General structure:

**for d in x; do** for each thing defined by x, which is given the variable name d, do the following:

**command** first this command

**command** then this command

**done** ends the loop

**If/Then/Else**

Tests for if a condition is true before applying a command. Can test for multiple conditions at once which can be combined using OR (when only one must be true) or AND (when all must be true). "If" is always used with "then" (what to do if the test is true) and can optionally include "else" (what to do if the test is not true). General structure:

**if [ test ]** test for a condition, e.g. \*\_master.xml for if the file name ends with \_master.xml

**then** if the test is true...

**command** … this is what the script should do

**else** if the test is not true… (optional - without else, it will do nothing if the test is false)

**command** ...this is what the script should do

**fi** end of the if test (if spelled backwards)

**Variables:**

Variables refer to some value by a name while running commands, such as the AIP folder or the output from a tool. The scripts define variables in two ways:

* In for loops, variables are set in the first line. The syntax is **for d in x; do** where d is the variable name and x is the variable value. Example: **for d in \*; do** sets the variable d to mean each thing in the current folder.
* For master.xml and bag validations, a variable is set to mean the output from the validation tools. The syntax is **valid=$(( tool command ) 2>&1)** where valid is the variable name and everything inside $() is the variable value. The variable's value is a combination of the tool's error output and the message it would normally display in the terminal after running.

We use the naming convention of d for a variable that is for directories and i for files (items). Variables are referred to in the commands as "$d" or "$i". The $ indicates it is a variable and the quotes make sure any spaces in the folder or file name will be treated properly in the terminal.

**Arguments:**

Arguments refer to information put into the terminal after the script name when the script is run. These are used to supply outside information into the script. Arguments are referred to in commands based on their order after the script name in the terminal: the first is "$1" and the second is "$2". In our scripts, "$1" is the source directory, used to navigate the file directory, and "$2" is the department name, used as a parameter value for the stylesheets that make the master.xml files.

**Most common commands in the scripts:**

**Standard bash commands**

* **#** indicates that row is a comment. Used to explain what each step does.
* **cd path/to/directory** change directory. It becomes the directory that relative paths in the script start from, although the current directory displayed in the terminal won't change.
* **cp what/to/copy where/to/copy/it/to** Copies file or folder from one place to another.
* **echo "text"** displays the text in the terminal. Used to display the script's status.
* **echo ""** displays a blank line in the terminal. Puts whitespace between notes for readability.
* **mkdir directory-name** makes a folder inside the current directory with that name
* **mkdir "$d"/directory-name** makes a folder inside each "$d", with "$d" defined by the for loop
* **mv what/to/move where/it/goes** move something to a new location; also used to rename
* **rm path/to/file** deletes the file
* **rmdir path/to/folder** deletes the folder - must be empty

**Checking arguments included in the terminal when run the script:**

These test for if the correct arguments needed for the script to run are NOT present. If any of the tests are true, a message displays in the terminal explaining the required arguments and the script quits.

* if [ "$#" -ne "2" ] tests if there are the wrong number of arguments (in this case if not 2)
* if [ ! -d "$1" ] tests if the 1st argument is not a directory
* if [ "$2" != "hargrett" ] && [ "$2" != "russell" ] tests if the 2nd argument is not hargrett and not russell

**Checking "$d" is a AIP folder before doing the command:**

Tests if something is a AIP folder by seeing if its name is an aip-id, meaning it starts with either harg or rbrl. This is so other folders in the source directory, e.g. master-xml, are skipped by that step.

**for d in \*; do**

**if [[ "$d" = "harg"\* ]] || [[ "$d" = "rbrl"\* ]]** if the folder name starts with harg or rbrl (the AIP id)

**then**

**command** does this only when it is a AIP folder

**fi**

**done**

**Naming:**

When an output file, such as the master.xml file, is made as part of the script, the value of the variable from the for loop can be used as part of the filename with the syntax ${d} or ${i}. This is most commonly done to include the aip-id as part of the filename. Just a portion of the variable value can be used and new things can also be added. Examples:

|  |  |  |
| --- | --- | --- |
| Command | Value of variable | Resulting filename |
| ${d}\_combined-fits.xml | aip-id | aip-id\_combined-fits.xml |
| ${i%\_cleaned-fits.xml}\_master.xml | aip-id\_cleaned-fits.xml | aip-id\_master.xml |

**Renaming:**

The mv command is also used for renaming by "moving" a file to the same location but with a different name. This can be be done with find/replace syntax to change or remove a portion of the existing name and content can also be added to the existing name. Examples:

|  |  |  |
| --- | --- | --- |
| Command | Value of variable | Resulting filename |
| mv "$d" "${d}\_bag" | aip-id | aip-id\_bag |
| mv "$d" "${d//\_\*/}" | aip-id\_AIP Title | aip-id |
| mv "$i" "${i//.fits/\_fits}" | filename.fits.xml | filename\_fits.xml |