NYU | Center for Data Science

# DS-GA 1007
# Programming for Data Science

## Lecture 6
## CLI II – Scripts and Make

## Reminders

- Labs
  - Due on Wednesday 11:59pm
- Homework
  - Due on Saturday 11:59pm
  - Submission
    - export notebook to html
    - print html to pdf
    - IPython.display.Image for embedded images
- Cluster
  - Network Administrator Group
  - Unavailable Monday October 14
- Midterm
  - Thursday October 24 in class

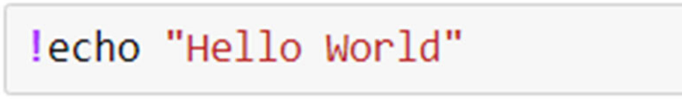
## Review

- Why use Shell?
  - Shell allows use to read, evaluate, print and loop through use of commands or other applications
  - Advantages
    - Automating tasks
    - Access network machines

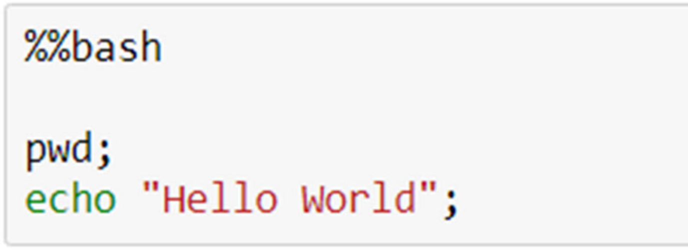
    https://wikis.nyu.edu/display/NYUHPC/Clusters
  - Disadvantages
    - Not a Graphical User Interface (GUI)

## Review

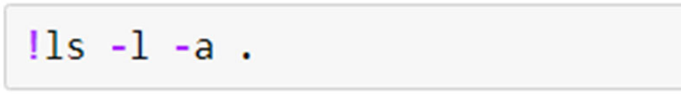
- Working with the Operating System
  - Shell
  - Command Line Interface
  - Terminal
- Commands
  - Options
  - Arguments
- Files and Directories
  - Paths
  - Glob-ing
    - Wildcards: * and ?
    - [Character Sets]



```
!echo "Hello World"
```



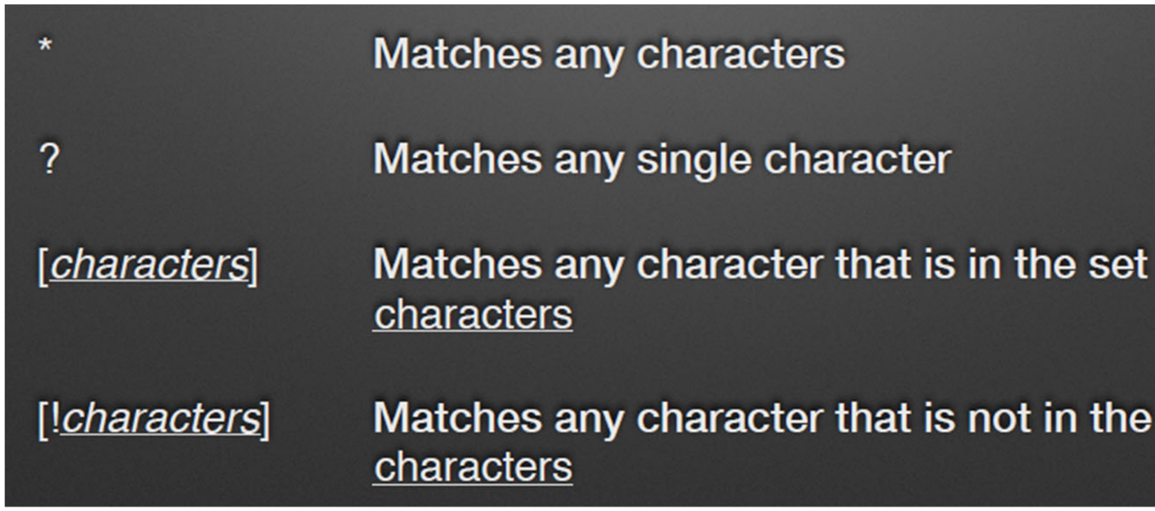
```
%%bash

pwd;
echo "Hello World";
```



```
!ls -l -a .
```

## Review

- Working with the Operating System
  - Shell
  - Command Line Interface
  - Terminal
- Commands
  - Options
  - Arguments
- Files and Directories
  - Paths
  - Glob-ing
    - Wildcards: * and ?
    - [Character Sets]



| | |
|---|---|
| * | Matches any characters |
| ? | Matches any single character |
| [*characters*] | Matches any character that is in the set characters |
| [!*characters*] | Matches any character that is not in the characters |

## Review

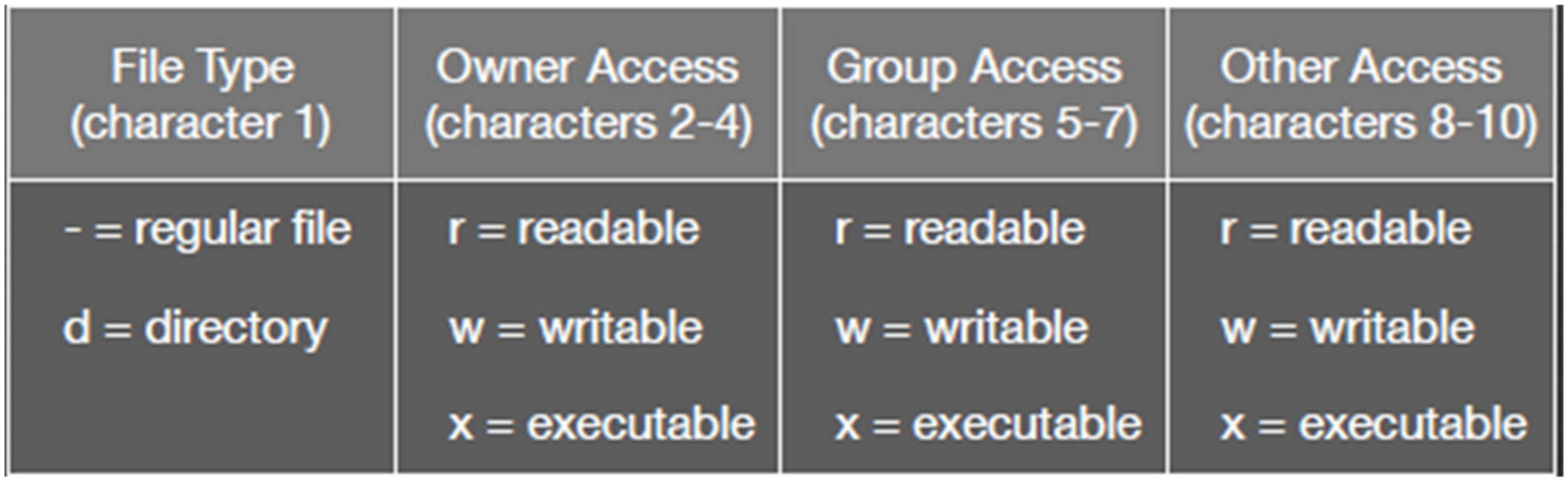
- Files and Directories
  - file
  - pwd
  - ls
  - cd
  - du
- Operations on Files
  - touch
  - mv
  - cp
  - rm
- Operations on Directories
  - rmdir
  - mkdir
- View Files
  - less
  - head
  - cat
- Keyboard Shortcuts
  - CTRL C
  - CTRL D
  - . and ..

## Objectives

- Set File Permissions for Owner, User, Group
- Use Redirection and Pipes to Handle Input and Output
- Set-up a bash script
- Pass arguments from the command line to script
- Implement a loop

## File Permissions

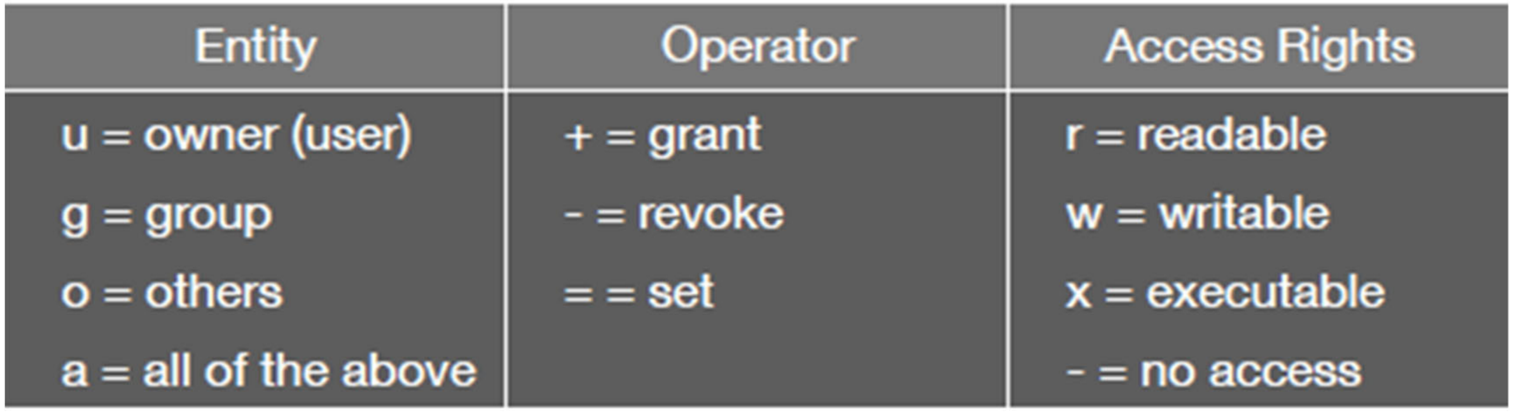
- Permissions used to control access to files and directories
- Use ls –l to show permissions of all files in current working directory
- Permissions are 10 character strings



| File Type (character 1) | Owner Access (characters 2-4) | Group Access (characters 5-7) | Other Access (characters 8-10) |
|---|---|---|---|
| - = regular file | r = readable | r = readable | r = readable |
| d = directory | w = writable | w = writable | w = writable |
|  | x = executable | x = executable | x = executable |

## File Permissions

- Command chmod used to change permission on file or directory
- For example chmod u+rw my_file.txt

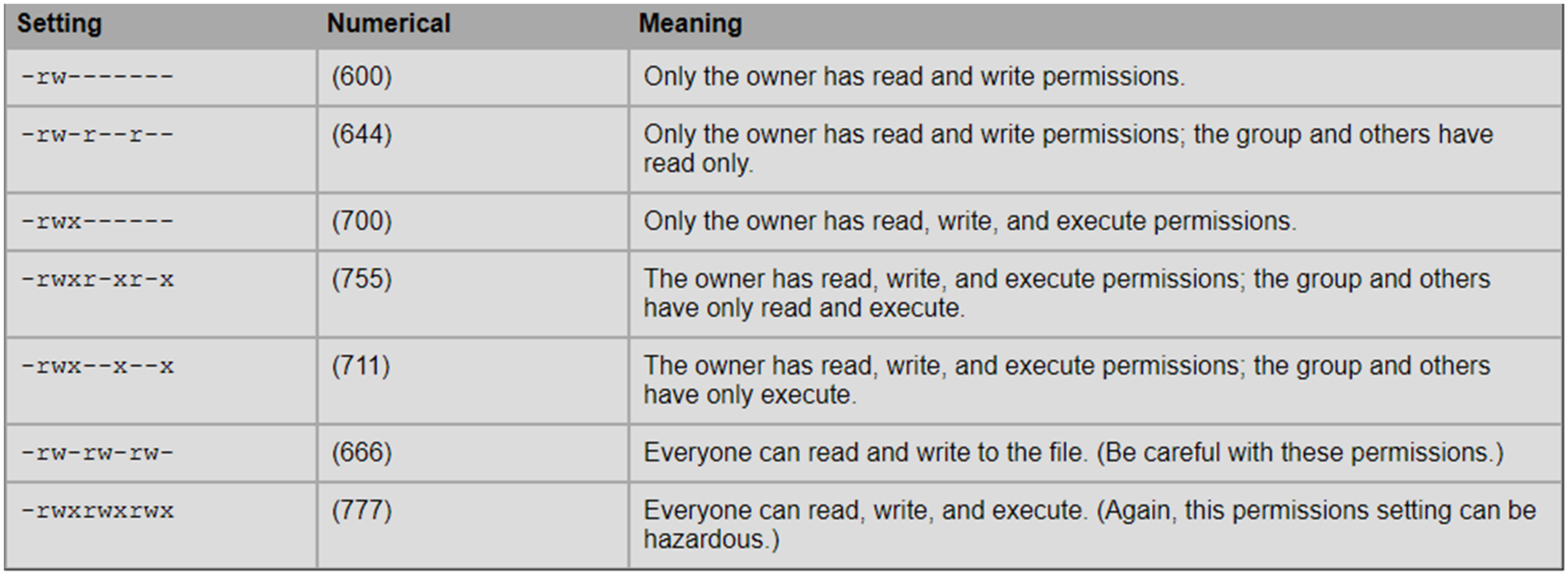


| Entity | Operator | Access Rights |
|---|---|---|
| u = owner (user) | + = grant | r = readable |
| g = group | - = revoke | w = writable |
| o = others | = = set | x = executable |
| a = all of the above | | - = no access |

## File Permissions

- Command chmod used to change permission on file or directory
- Three digit number for owner, user, group that sums
  - r = 4
  - w = 2
  - x = 1
  - - = 0
- For example, chmod 700 my_file.txt

## File Permissions



| Setting | Numerical | Meaning |
|---|---|---|
| `-rw-------` | (600) | Only the owner has read and write permissions. |
| `-rw-r--r--` | (644) | Only the owner has read and write permissions; the group and others have read only. |
| `-rwx------` | (700) | Only the owner has read, write, and execute permissions. |
| `-rwxr-xr-x` | (755) | The owner has read, write, and execute permissions; the group and others have only read and execute. |
| `-rwx--x--x` | (711) | The owner has read, write, and execute permissions; the group and others have only execute. |
| `-rw-rw-rw-` | (666) | Everyone can read and write to the file. (Be careful with these permissions.) |
| `-rwxrwxrwx` | (777) | Everyone can read, write, and execute. (Again, this permissions setting can be hazardous.) |

## File Permissions

| Setting | Numerical | Meaning |
|---|---|---|
| `drwx------` | (700) | Only the user can read, write in this directory. |
| `drwxr-xr-x` | (755) | Everyone can read the directory; users and groups have read and execute permissions. |

**Caution**

Remember that file permissions are a security feature. Whenever you allow anyone else to read, write to, and execute files, you are increasing the risk of files being tampered with, altered, or deleted. As a rule, you should only grant read and write permissions to those who truly need them.

## Links

- Hard Links
  - Linked file references same memory location as the original. Linked file has data of original
  - Both remain linked even if the original or linked files are moved throughout the file system.
  - If original file is removed then the link will still show the content of the file. Removing any link, just reduces the link count, but doesn’t affect other links.
  - $ ln [original filename] [link name]

## Links

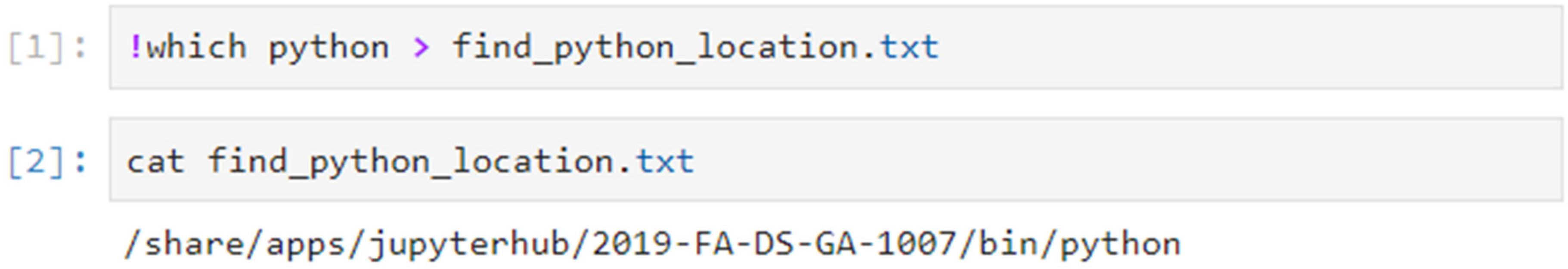
- Soft link
- Linked file contains the path for original file but not data.
- Command to create a Soft link is:$ ln -s [original filename] [link name]
- Removing soft link doesn’t affect anything. Removing link becomes “dangling” link which points to nonexistent file.

## Input and Output

- Standard Input
  - Usually come from keyboard
- Standard Output
  - Usually sent to screen
- Standard Error
  - Usually sent to screen
- Redirection
  - Changing where input comes from
  - Changing where output is going

## Input and Output

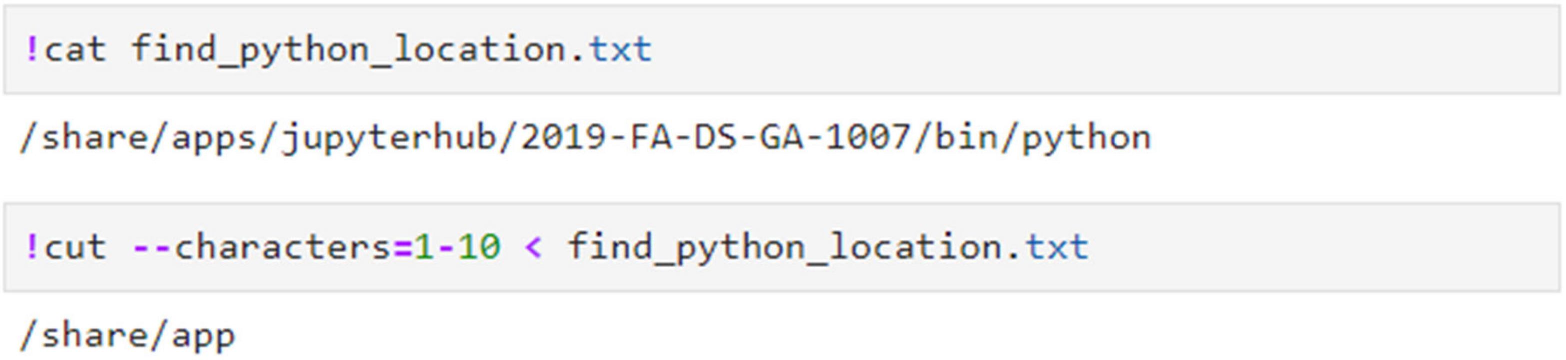
- Output redirection
  - Overwrite enabled with > character
  - Append enabled with >> characters
  - Redirection comes after command



```
[1]: !which python > find_python_location.txt

[2]: cat find_python_location.txt

     /share/apps/jupyterhub/2019-FA-DS-GA-1007/bin/python
```

## Input and Output

- Input redirection
  - Enabled with < character
  - Send content of file to command like from standard input



```
!cat find_python_location.txt
```

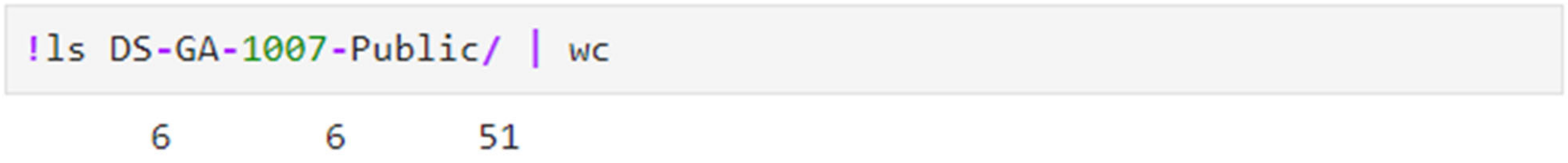
```
/share/apps/jupyterhub/2019-FA-DS-GA-1007/bin/python
```

```
!cut --characters=1-10 < find_python_location.txt
```

```
/share/app
```

## Input and Output

- Pipes
  - To redirect output of one command to input of another command
  - Enabled with | character
  - Send content of file to command like from standard input

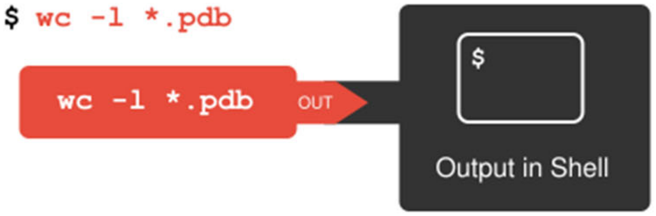


```
!ls DS-GA-1007-Public/ | wc
```

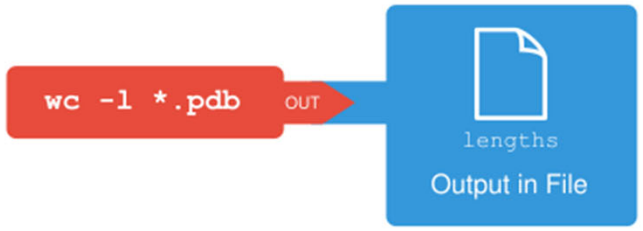
```
      6       6      51
```

## Input and Output

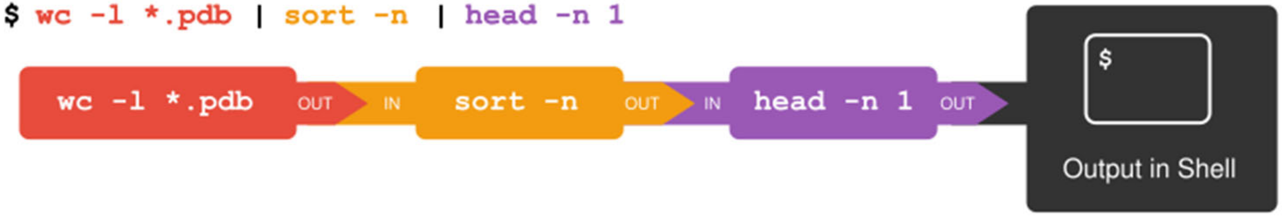
```
$ wc -l *.pdb
```



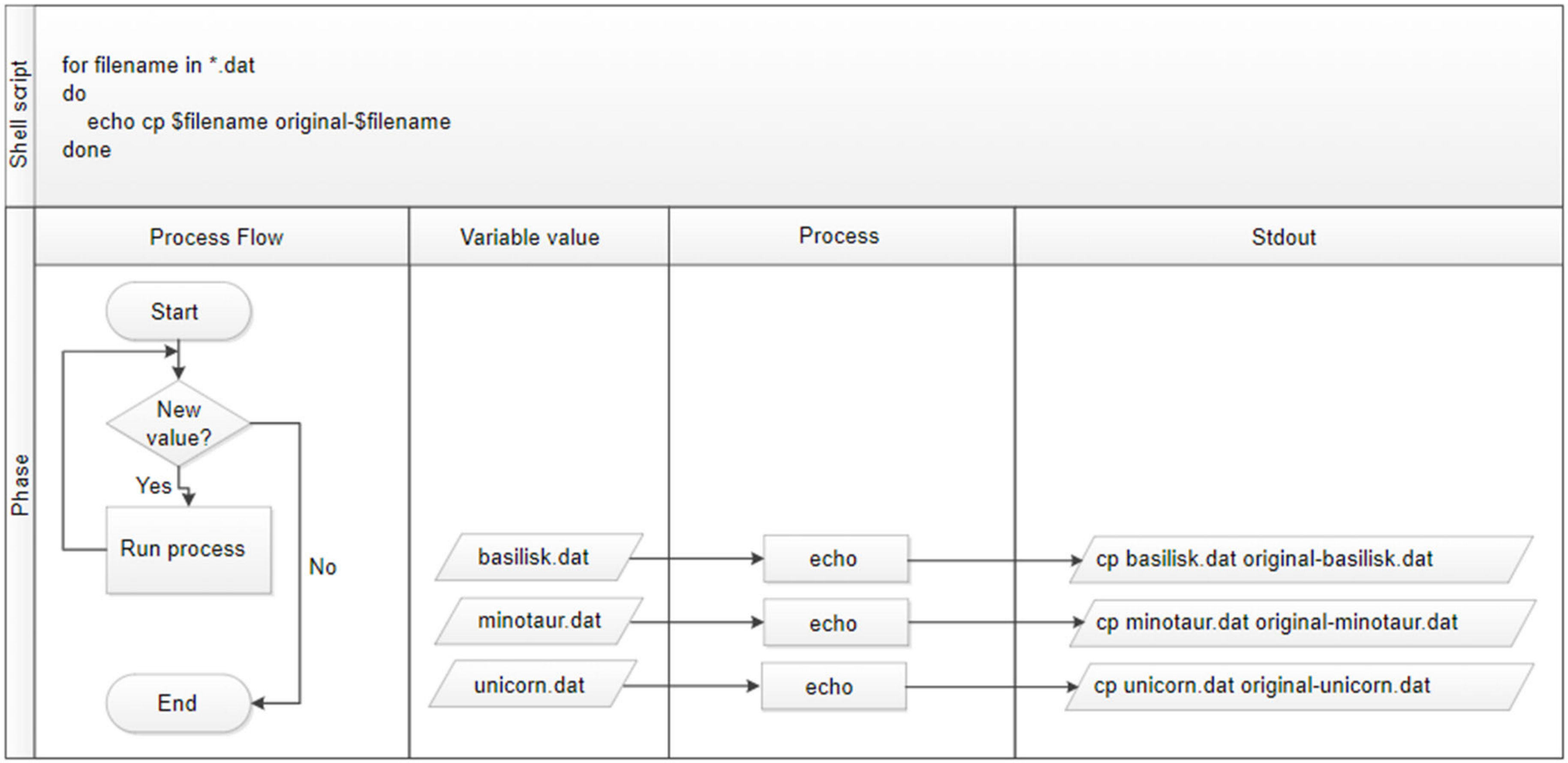
```
$ wc -l *.pdb > lengths
```



```
$ wc -l *.pdb | sort -n | head -n 1
```



## Loops



Shell script

```
for filename in *.dat
do
    echo cp $filename original-$filename
done
```

## Shell Scripts

- Scripts give us a way to save a collection of commands for reuse
- Script has three components
  - Indicate how to run script
  - Name of script
  - Arguments
- For example…
  - bash middle.sh octane.pdb 10 5
  - bash -x middle.sh octane.pdb 10 5

## Setting up Scripts

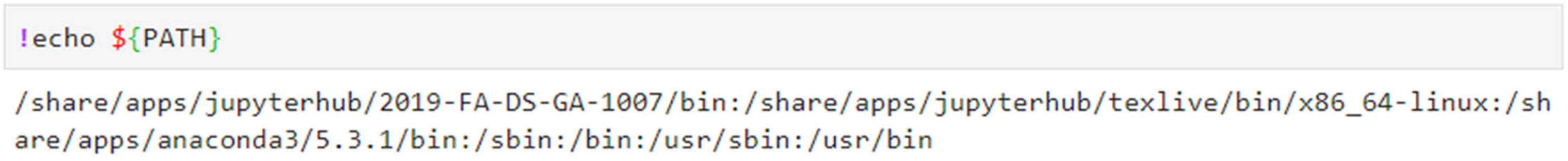
- File Extension
  - Can use .sh or .bash
- Executable
  - First Line
    - Set to #!/bin/bash. Try *which bash* to verify location
    - Note it’s not a comment despite #
  - Permissions
    - chmod +x my_script.sh
  - Run Script
    - absolute or relative path ./my_command

## Setting up Scripts

- Question
  - Will my_script.sh work?

## Setting up Scripts

- Question
  - Will my_script.sh work?
- Depends on environment variable PATH



```
!echo ${PATH}
```

```
/share/apps/jupyterhub/2019-FA-DS-GA-1007/bin:/share/apps/jupyterhub/texlive/bin/x86_64-linux:/sh
are/apps/anaconda3/5.3.1/bin:/sbin:/bin:/usr/sbin:/usr/bin
```

## Take-Aways

- File Permissions
- Redirection and Pipes
- Links
- Set-up a bash script
- Pass arguments from the command line to script
- Implement a loop and conditional