Introduction to Bash Scripting

A River Crossing Puzzle
This tutorial will use a classical problem in which a farmer wishes to cross a fox, chicken, and bag of grain from the left bank of the river to the right bank of the river. On each single river crossing the farmer may cross with at most one of the fox, chicken and bag of grain. Unfortunately, if the farmer leaves the chicken alone with the fox, the fox will eat the chicken; the chicken will eat the grain, if the two are ever left alone on the bank. The objective is to get all three items across the river safely.

What is a Bash Script?

Bash (Bourne Again SHell) is a Unix shell and command language widely used on Linux, macOS, and other Unix-like systems. It allows users to interact with their system by typing commands in the terminal. With Bash, you can also write **scripts** to automate tasks and perform system administration, file manipulation, and more.

A **Bash script** is a text file containing a sequence of commands that you could manually type in the terminal. These scripts are useful for automating repetitive tasks, setting up environments, and simplifying complex workflows.

Bash Scripts are used for:

- **Automation**: Execute multiple commands in a single script without typing them manually each time.
- **Efficiency**: Automate tasks that are repetitive or time-consuming.
- **Consistency**: Ensure that processes run the same way every time by using the same script.
- **Customization**: You can build scripts tailored to your specific needs.

Creating Directories and Files

Open a linux terminal and create two directories:
`mkdir left_bank right_bank`

Create four files in the `left_bank` directory to represent the farmer, chicken, fox, and grain:
`touch left_bank/farmer left_bank/fox left_bank/chicken left_bank/grain`

You can later use these directories to represent the two sides of the river and files to represent the fox, chicken, and grain.

Viewing the Present Working Directory

View the present working directory using the `pwd` command:

```
pwd
```

Changing Directory and Listing Files

Change the directory using the `cd` command:

```
cd ./left_bank
```

You can list the files in a directory using the `ls` command:

```
ls
```

You should see all the items (i.e., files) listed:

```
farmer
chicken
fox
grain
```

Change the directory to `right_bank` and list the files in that directory:

```
cd ../right_bank
ls
```

You should see no items (i.e., files) listed.

Saving Information in Variables

You can store information in variables and use them later.

First let's return to the parent directory of `right_bank` (and `left_bank`):

```
cd ..
```

Then run each of the following commands in turn:

```
item="chicken"
from="left_bank"
to="right_bank"
echo "About to move $item from $from to $to"
```

Copying, Moving and Deleting Files

Move the chicken from `left_bank` to `right_bank` and list the files in the `left_bank`:
```
mv "$from/$item" "$to/"
ls left_bank
```

You should see 3 items excluding the `chicken`.
List the files in the `right_bank`:
```
ls right_bank
```

You should see just the `chicken` listed.

Of course this means the chicken would eat the grain and/or the fox would eat the chicken, so let's urgently move the farmer back:

```
mv "$to/$item" "$from/"
```

List the files in the `left_bank`:
```
ls left_bank
```

You should see all 4 items listed.

List the files in the `right_bank`:
```
ls right_bank
```

You should see no items listed.

Writing A First Bash Script

Create a new file using a text editor like `nano`:
```
nano river_crossing.sh
```

Inside the file, type the following:
```
#!/bin/bash
echo "Foxes eat chickens, chickens eat grain"
```

The first line tells the system to run the script using the Bash shell, the second line prints the quoted text to the terminal.

Save the file and exit nano (`CTRL + X`, then `Y`, then `ENTER`).

Make the script executable:

```
chmod +x river_crossing.sh
```

Run the script:
```
./river_crossing.sh
```

You should see this output:
```
  Foxes eat chickens, chickens eat grain
```

Extending the Script with the Commands You Have Seen Already

Extend `river_crossing.sh` in the following way and save and run the file.

```
#!/bin/bash
echo "Foxes eat chickens, chickens eat grain"
item1="farmer"
item2="chicken"
from="left_bank"
to="right_bank"
mv "$from/$item1" "$to/"
mv "$from/$item2" "$to/"
echo "Moved $item1 and $item2 from $from to $to"
```

Running this script will output:
```
Foxes eat chickens, chickens eat grain
Moved farmer and chicken from left_bank to right_bank
```

List the files in the left_bank directory
```
ls ./left_bank
```

Only the fox and the grain should remain; that's fine as foxes don't eat grain in this puzzle.

List the items in the left_bank
```
ls ./right_bank
```

The farmer and the chicken should now be listed.

Passing Arguments When Calling a Script

In a bash script `$1` is the first argument, `$2` is the second argument, and so on.

Create a new script called `move_item.sh` containing the following lines:

```
#!/bin/bash
echo "About to move the $1"
```

Running the script to move the fox:
```
./move_item.sh fox
```

The console should display:
```
About to move the fox
```

Using if Statements

Revise `move_item.sh` in the following way
```
#!/bin/bash
farmer="farmer"
if [ "$1" == "$farmer" ]
then
  echo "About to move the farmer"
else
  echo "About to move the farmer and the $1"
fi
```

Run the script to move the farmer:
```
./move_item.sh farmer
```

The following should be displayed:
```
About to move the farmer
```

Whereas, running the script to move the chicken should display:
```
About to move the farmer and the chicken
```

Writing Functions

```
#!/bin/bash
farmer="farmer"
local item=$1
local from=$2
local to=$3
if [ $item == farmer ]
then
  mv "$from/$item" "$to/"
```

```
  echo "Moved $item from $from to $to"
else
  mv "$from/farmer" "$to/"
  mv "$from/$item" "$to/"
  echo "Moved farmer and $item from $from to $to"
fi
echo "Left Bank:"
echo "$(ls -1 left_bank)"
echo "Right Bank:"
echo "$(ls -1 right_bank)"
echo "-----------------------------"
```

GOAL

Use the script to safely move the farmer, fox, chicken and grain files from the left_bank directory to the right_bank directory.

Which of the following can you achieve as stretch goals:

- Write a script that performs each of the commands used to accomplish the GOAL above.
- Tidy up your scripts with the use of functions.
- Make the code more robust by adding checks to establish whether a file is present in a directory before attempting to move it.
- Write helpful bash scripts that move files in a similar manner for the Towers of Hanoi puzzle, rather than the River Crossing puzzle.
- Come up with your own similar puzzle that exercises your ability to write bash script.

Conclusion

In this tutorial, you learned:
- The basics of what a Bash script is and how to write one.
- How to perform file operations such as moving files between directories.
- How to use basic Bash features like variables, input parameters and conditionals.