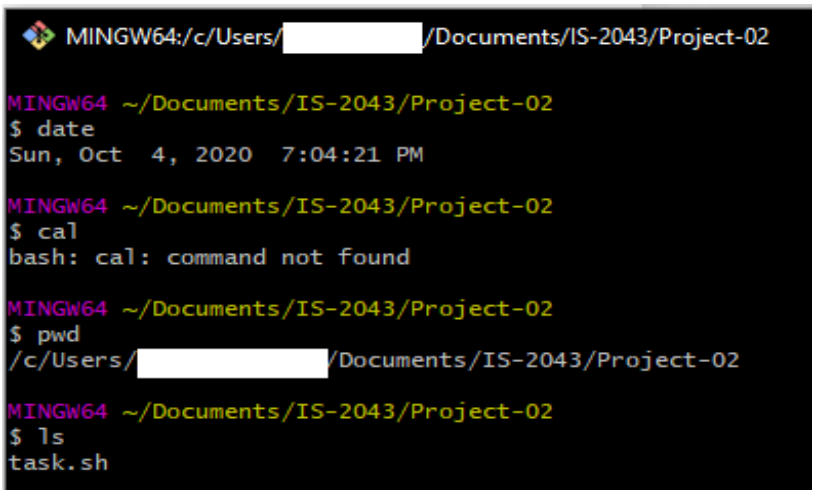
Bash Shell Scripting Definition

To begin understanding bash shell scripting, I had to read and understand the definitions of it. From my understanding, Bash is a command language interpreter that is an abbreviation of Bourne-Again SHell. Shell is a macro processor that accepts command executions. Scripting is a way to run commands automatically.

What is Shell

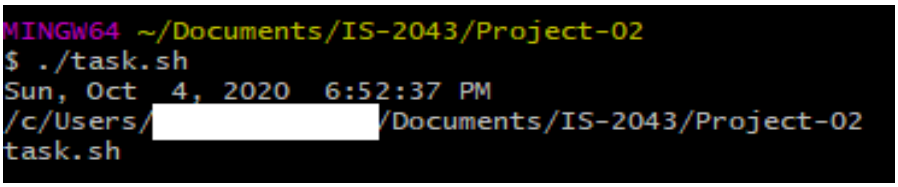
The basics of shell consisted of running the commands, *date*, *cal*, *pwd*, and *ls* in Git BASH.



Commands.

What is Scripting

To begin scripting, I created a directory called Project-02 using the command *mkdir Project-02*. The first filed I created was task.sh by using *nano task.sh*. In the file I put the commands *date*, *cal*, *pwd*, *ls*. I saved the file and then made the file an executable with the command *chmod +x tash.sh*. I ran the file with the command *./task.sh*. The error of *bash: CAL: command not found* was returned and I could not find a way to add *cal* to Git BASH. I commented it out the *cal* commend and ran the file again.



./task.sh

What is Bash

Bash is the default interpreter, so I have been already using it. To see my default interpreter, I ran the command *echo $SHELL*.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02

MINGW64 ~/Documents/IS-2043/Project-02
$ echo $SHELL
/usr/bin/bash
```

Default Interpreter.

To define my scripts interpreter, I ran the command *which bash* to get the full path to the executable binary. I then inserted *#!/usr/bin/bash* at the top of task.sh.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02

MINGW64 ~/Documents/IS-2043/Project-02
$ which bash
/usr/bin/bash
```

which command.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02
  GNU nano 4.9.3                                  task.sh
#!/usr/bin/bash

date
#cal
pwd
ls
```

#!/usr/bin/bash.

File Names and Permissions

To understand File Names and Permissions, I had to create a file called hello-world.sh that contained the text *Test*. To identify its file type, I ran the command *file hello-world.sh*. I did not receive the same results as the tutorial; Bourne-Again shell script and text executable were missing.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02

MINGW64 ~/Documents/IS-2043/Project-02
$ chmod +x hello-world.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ file hello-world.sh
hello-world.sh: ASCII text
```

file hello-world.sh.

Next, I copied hello-world.sh to a file called 0_xvz that had no extension. Running *file 0_xvz* displayed the same results as file *hello-world.sh*.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02

MINGW64 ~/Documents/IS-2043/Project-02
$ cp hello-world.sh 0_xvz

MINGW64 ~/Documents/IS-2043/Project-02
$ file 0_xvz
0_xvz: ASCII text
```

file 0_xvz.

Script Execution

To execute a script, I created a date script using the command *echo date > date.sh*. The command *cat date.sh* displayed the contents of the file and the commands *./date.sh* and *bash date.sh* ran the file. I also put *#!/usr/bin/bash* at the top of the date.sh file. Though I didn't run into a bash error like the tutorial, I did run the command *chmod +x date.sh*.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02

MINGW64 ~/Documents/IS-2043/Project-02
$ echo date > date.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ cat date.sh
date

MINGW64 ~/Documents/IS-2043/Project-02
$ ./date.sh
Sun, Oct  4, 2020  7:42:55 PM

MINGW64 ~/Documents/IS-2043/Project-02
$ bash date.sh
Sun, Oct  4, 2020  7:43:08 PM

MINGW64 ~/Documents/IS-2043/Project-02
$ nano date.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ chmod +x date.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ ./date.sh
Sun, Oct  4, 2020  7:50:16 PM
```

date script execution.

Relative vs Absolute Path

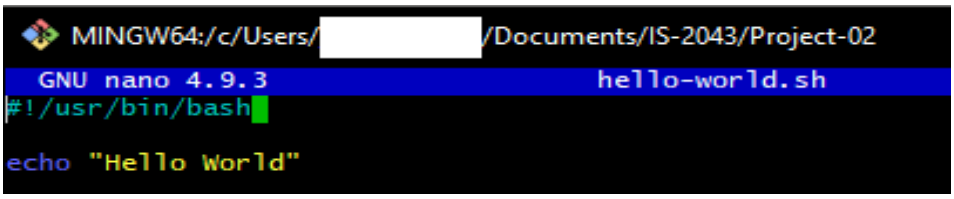
To learn Relative and Absolute path I had to navigate directories using *pwd*, *cd /*, *cd…*, and *cd -*. I did not have the same directories as the tutorial and had to note the directory I started with to get back to it.

```
MINGW64:/c/Users/[redacted]Documents/IS-2043/Project-02

MINGW64 ~/Documents/IS-2043/Project-02
$ pwd
/c/Users/[redacted]/Documents/IS-2043/Project-02

MINGW64 ~/Documents/IS-2043/Project-02
$ cd /

MINGW64 /
$ pwd
/

MINGW64 /
$ cd home/
bash: cd: home/: No such file or directory

MINGW64 /
$ ls
bin/  etc/           LICENSE.txt  ReleaseNotes.html  unins000.exe*
cmd/  git-bash.exe*  mingw64/     tmp/               unins000.msg
dev/  git-cmd.exe*   proc/        unins000.dat       usr/

MINGW64 /
$ cd ..

MINGW64 /
$ pwd
/

MINGW64 /
$ cd usr/

MINGW64 /usr
$ ls
bin/  etc/  lib/  libexec/  share/  ssl/

MINGW64 /usr
$ cd /etc/

MINGW64 /etc
$ ls
bash.bash_logout  hosts                nsswitch.conf         services
bash.bashrc       inputrc              package-versions.txt  ssh/
DIR_COLORS        install-options.txt  pkcs11/               tigrc
docx2txt.config   msystem              pki/                  vimrc
fstab             mtab@                profile
gitattributes     nanorc               profile.d/
gitconfig         networks             protocols

MINGW64 /etc
$ cd ../~//Documents/IS-2043/Project-02
bash: cd: ../~//Documents/IS-2043/Project-02: No such file or directory

MINGW64 /etc
$ cd ../~/Documents/IS-2043/Project-02
bash: cd: ../~/Documents/IS-2043/Project-02: No such file or directory

MINGW64 /etc
$ cd ../Documents/IS-2043/Project-02
bash: cd: ../Documents/IS-2043/Project-02: No such file or directory

MINGW64 /etc
$ cd /

MINGW64 /
$ ls
bin/  etc/           LICENSE.txt  ReleaseNotes.html  unins000.exe*
cmd/  git-bash.exe*  mingw64/     tmp/               unins000.msg
dev/  git-cmd.exe*   proc/        unins000.dat       usr/

MINGW64 /
$ cd /~
bash: cd: /~: No such file or directory

MINGW64 /
$ cd ~
```

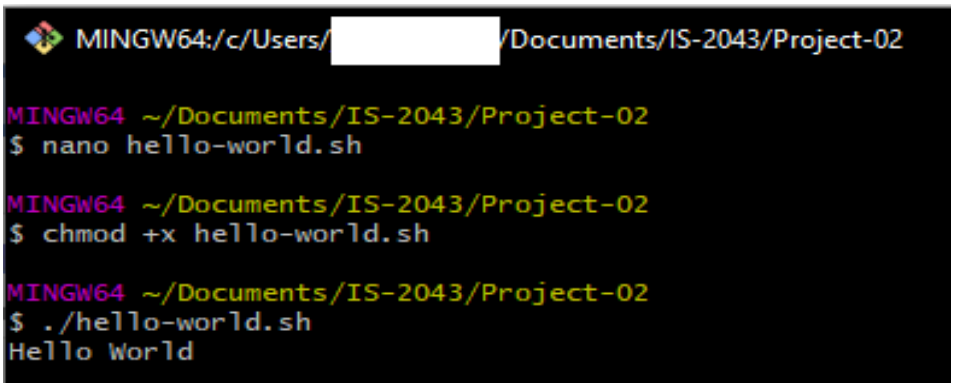
Navigating directories.

Hello World Bash Shell Script

To create a Hello World Bash Shell Script, I edited the previously created hello-world.sh file to include *#!/usr/bin/bash* and *echo "Hello World"*. I then make the file an executable and run it. It prints Hello World.



hello-world.sh



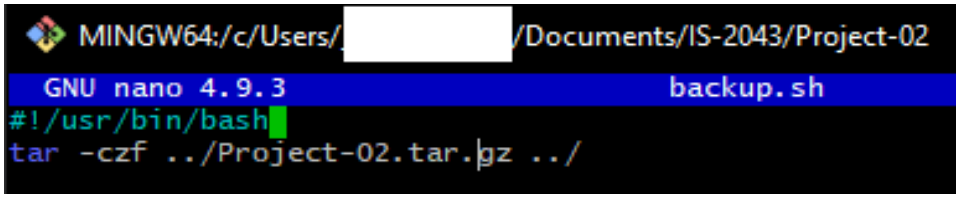
hello-world.sh executed.

Simple Backup Bash Shell Script

I could not get the man command to work with Git BASH.

To backup a directory I created a file called backup.sh and included *#!/usr/bin/bash* and

tar -czf ../Project-02.tar.gz ../. I then made the file an executable and ran it. It created the Project-02.tar.gz in the directory above the current directory.



backup.sh.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043
MINGW64 ~/Documents/IS-2043/Project-02
$ chmod +x backup.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ ./backup.sh
tar: Removing leading `..' from member names
tar: Removing leading `../' from member names
tar: ../Project-02.tar.gz: file changed as we read it

MINGW64 ~/Documents/IS-2043/Project-02
$ cd ../

MINGW64 ~/Documents/IS-2043
$ ls
'~$adMeofP2.docx'       Greenup-Shadice_vbx237-p01.pdf        Project-02.tar.gz
'~WRL1299.tmp'          Greenup-Shadice_vbx237-p01.pdf.docx   readmeforP02.txt
 1.5/                   Java/                                 ReadMeofP2.docx
 Deitel_code_ch08/      Java.rar                              voting.PNG
 Deitel_code_ch08.zip   Project-02/
```

Project-02.tar.gz file.

Variables

To work with variables, I created a file called welcome.sh and included the code that the tutorial stated to include. The code defined three variables, greeting, user, and day and assigned values to the variables using commands and text. I made the file an executable and ran it. The file displayed the three defined variables and a variable that displayed the bash version.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02
  GNU nano 4.9.3                    welcome.sh
#!/usr/bin/bash

greeting="Welcome"
user=$(whoami)
day=$(date +%A)

echo "$greeting back $user! Today is $day, which is the best day of the entire
echo "Your Bash shell version is: $BASH_VERSION. Enjoy!"
```

welcome.sh

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02

MINGW64 ~/Documents/IS-2043/Project-02
$ nano welcome.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ chmod +x welcome.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ ./welcome
bash: ./welcome: No such file or directory

MINGW64 ~/Documents/IS-2043/Project-02
$ ./welcome.sh
Welcome back [redacted]! Today is Sunday, which is the best day of the entire
 week!
Your Bash shell version is: 4.4.23(1)-release. Enjoy!
```

welcome.sh execution.

The tutorial requested a change to the backup.sh to backup a user's home directory without being bind to a specific user. Executing the script did not work for me because I do not have the same directories as the tutorial.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02
  GNU nano 4.9.3                    backup.sh                    Modif
#!/usr/bin/bash

# This bash script is used to backup a user's home directory to /tmp/.

user=$(whoami)
input=/home/$user
output=/tmp/${user}_home_$(date +%Y-%m-%d_%H%M%S).tar.gz

tar -czf $output $input
echo "Backup of $input completed! Details about the output backup file:"
ls -l $output
```

backup.sh.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02/home

MINGW64 ~/Documents/IS-2043/Project-02
$ nano backup.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ ./backup.sh
tar: Removing leading `/' from member names
tar: /home/[redacted]: Cannot stat: No such file or directory
tar: Exiting with failure status due to previous errors
Backup of /home/[redacted] completed! Details about the output backup file:
-rw-r--r-- 1 [redacted] 197121 45 Oct  4 21:27 /tmp/[redacted]__home_2020-1
0-04_212712.tar.gz

MINGW64 ~/Documents/IS-2043/Project-02
$ ls
0_xvz       foobar          stderr.txt          task.sh*
backup.sh*  hello-world.sh* stdout.txt          welcome.sh*
date.sh*    home/           stdoutandstderr.txt

MINGW64 ~/Documents/IS-2043/Project-02
$ cd home

MINGW64 ~/Documents/IS-2043/Project-02/home
$ ls
```

backup.sh execution.

Input, Output and Error Redirections

I had to learn about standard error output (stderr) and standard output (stdout). One displays an error and the other displays an output. I also had to learn how to redirect both to a file.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02

MINGW64 ~/Documents/IS-2043/Project-02
$ ls -l foobar
ls: cannot access 'foobar': No such file or directory

MINGW64 ~/Documents/IS-2043/Project-02
$ touch foobar

MINGW64 ~/Documents/IS-2043/Project-02
$ ls -l foobar
-rw-r--r-- 1 [redacted] 197121 0 Oct  4 21:17 foobar
```

stderr and stdout.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02

MINGW64 ~/Documents/IS-2043/Project-02
$ ls foobar barfoo
ls: cannot access 'barfoo': No such file or directory
foobar

MINGW64 ~/Documents/IS-2043/Project-02
$ ls foobar barfoo > stdout.txt
ls: cannot access 'barfoo': No such file or directory

MINGW64 ~/Documents/IS-2043/Project-02
$ ls foobar barfoo 2> stderr.txt
foobar

MINGW64 ~/Documents/IS-2043/Project-02
$ ls foobar barfoo &> stdoutandstderr.txt

MINGW64 ~/Documents/IS-2043/Project-02
$ cat stdout.txt
foobar

MINGW64 ~/Documents/IS-2043/Project-02
$ cat stderr.txt
ls: cannot access 'barfoo': No such file or directory

MINGW64 ~/Documents/IS-2043/Project-02
$ cat stdoutandstderr.txt
ls: cannot access 'barfoo': No such file or directory
foobar
```

redirect messages to file and display them.

The tutorial requested that I add 2> /dev/null to the line beginning with tar in the backup.sh file to remove the tar error message.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02
  GNU nano 4.9.3                    backup.sh                    Modi
#!/usr/bin/bash

# This bash script is used to backup a user's home directory to /tmp/.

user=$(whoami)
input=/home/$user
output=/tmp/${user}_home_$(date +%Y-%m-%d_%H%M%S).tar.gz

tar -czf $output $input 2> /dev/null
echo "Backup of $input completed! Details about the output backup file:"
ls -l $output
```

backup.sh

The tutorial taught me that any keystroke is standard input stream (stdin).

Functions

To create a function, I created a file called function.sh and inputted what the tutorial stated. I then created an executable and ran it. It displayed the function.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02
  GNU nano 4.9.3                          function.sh
#!/usr/bin/bash

function user_details {
        echo "User Name: $(whoami)"
        echo "Home Directory: $HOME"
}

user_details
```

function.sh

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02

MINGW64 ~/Documents/IS-2043/Project-02
$ nano function.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ chmod +x function.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ ./function.sh
User Name: [redacted]
Home Directory: /c/Users/[redacted]
```

function.sh execution.

I then had to edit backup.sh to include functions for total number of files in a directory and total number of directories in a directory.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02
  GNU nano 4.9.3
#!/usr/bin/bash

# This bash script is used to backup a user's home directory to /tmp/.

user=$(whoami)
input=/home/$user
output=/tmp/${user}_home_$(date +%Y-%m-%d_%H%M%S).tar.gz

# The function total_files reports a total number of files for a given directory.
function total_files {
        find $1 -type f | wc -l
}

# The function total_directories reports a total number of directories
# for a given directory.
function total_directories {
        find $1 -type d | wc -l
}

tar -czf $output $input 2> /dev/null

echo -n "Files to be included:"
total_files $input
echo -n "Directories to be included:"
total_directories $input

echo "Backup of $input completed!"

echo "Details about the output backup file:"
ls -l $output
```

backup.sh.

Numeric and String Comparisons

To do comparisons, I created a file called comparison.sh and included what the tutorial stated. I then created an executable and ran the script.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02
  GNU nano 4.9.3                    comparison.sh
#!/usr/bin/bash

string_a="UNIX"
string_b="GNU"

echo "Are $string_a and $string_b strings equal?"
[ $string_a = $string_b ]
echo $?

num_a=100
num_b=100

echo "Is $num_a equal to $num_b ?"
[ $num_a -eq $num_b ]
echo $?
```

comparison.sh.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02

MINGW64 ~/Documents/IS-2043/Project-02
$ nano comparison.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ chmod +x comparison.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ ./comparison.sh
Are UNIX and GNU strings equal?
1
Is 100 equal to 100 ?
0
```

comparison.sh execution.

Conditional Statements

To do conditionals, I created a file called if_else.sh and included what the tutorial stated. I then created an executable and ran the script.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02
  GNU nano 4.9.3                          if_else.sh
#!/usr/bin/bash

num_a=400
num_b=200

if [ $num_a -lt $num_b ]; then
    echo "$num_a is less than $num_b!"
else
    echo "$num_a is greater than $num_b!"
fi
```

if_else.sh.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02

MINGW64 ~/Documents/IS-2043/Project-02
$ nano if_else.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ ./if_else.sh
400 is greater than 200!
```

if_else.sh execution.

I then had to edit backup.sh to include a sanity check to compare the difference between the total number of files before and after backup.

```
MINGW64:/c/Users/[redacted]/Documents/IS-2043/Project-02
  GNU nano 4.9.3
#!/usr/bin/bash

# This bash script is used to backup a user's home directory to /tmp/.

user=$(whoami)
input=/home/$user
output=/tmp/${user}_home_$(date +%Y-%m-%d_%H%M%S).tar.gz

function total_files {
        find $1 -type f | wc -l
}

function total_directories {
        find $1 -type d | wc -l
}

function total_archived_directories {
        tar -tzf $1 | grep  /$ | wc -l
}

function total_archived_files {
        tar -tzf $1 | grep -v /$ | wc -l
}

tar -czf $output $input 2> /dev/null

src_files=$( total_files $input )
src_directories=$( total_directories $input )

arch_files=$( total_archived_files $output )
arch_directories=$( total_archived_directories $output )

echo "Files to be included: $src_files"
echo "Directories to be included: $src_directories"
echo "Files archived: $arch_files"
echo "Directories archived: $arch_directories"

if [ $src_files -eq $arch_files ]; then
        echo "Backup of $input completed!"
        echo "Details about the output backup file:"
        ls -l $output
else
        echo "Backup of $input failed!"
fi
```

backup.sh

Positional Parameters

To do Positional Parameters, I created a file called param.sh and included what the tutorial stated. I then created an executable and ran the script. This script displays the number of parameters.

```
#!/usr/bin/bash

echo $1 $2 $4
echo $#
echo $*
```

param.sh

```
MINGW64 ~/Documents/IS-2043/Project-02
$ which bash > param.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ nano param.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ chmod +x param.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ ./param.sh

0


MINGW64 ~/Documents/IS-2043/Project-02
$ ./param.sh 1 2 3 4
1 2 4
4
1 2 3 4

MINGW64 ~/Documents/IS-2043/Project-02
$ ./param.sh hello bash scripting world
hello bash world
4
hello bash scripting world
```

param.sh execution.

I then had to edit backup.sh to include positional parameters.

```bash
#!/usr/bin/bash

# This bash script is used to backup a user's home directory to /tmp/.

if [ -z $1 ]; then
        user=$(whoami)
else
        if [ ! -d "/home/$1" ]; then
                echo "Requested $1 user home directory doesn't exist."
                exit 1
        fi
        user=$1
fi

input=/home/$user
output=/tmp/${user}_home_$(date +%Y-%m-%d_%H%M%S).tar.gz

function total_files {
        find $1 -type f | wc -l
}

function total_directories {
        find $1 -type d | wc -l
}

function total_archived_directories {
        tar -tzf $1 | grep  /$ | wc -l
}

function total_archived_files {
        tar -tzf $1 | grep -v /$ | wc -l
}

tar -czf $output $input 2> /dev/null

src_files=$( total_files $input )
src_directories=$( total_directories $input )

arch_files=$( total_archived_files $output )
arch_directories=$( total_archived_directories $output )

echo "Files to be included: $src_files"
echo "Directories to be included: $src_directories"
echo "Files archived: $arch_files"
echo "Directories archived: $arch_directories"

if [ $src_files -eq $arch_files ]; then
        echo "Backup of $input completed!"
        echo "Details about the output backup file:"
        ls -l $output
else
        echo "Backup of $input failed!"
fi
```

backup.sh

For Loop

To do a for loop, I created a file called forloop.sh and included what the tutorial stated. I then created an executable and ran the script. This script displays the result of the for loop.

```
  GNU nano 4.9.3                    forloop.sh
#!/usr/bin/bash

for i in 1 2 3; do
    echo $i
done
```

forloop.sh

```
MINGW64 ~/Documents/IS-2043/Project-02
$ nano forloop.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ chmod +x forloop.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ ./forloop.sh
1
2
3
```

forloop.sh execution.

I then had to create a file called items.txt that contain a list of items. I then inputted a for loop command that counted characters for each line.

```
  GNU nano 4.9.3                    items.txt
bash
scripting
tutorial
```

items.txt.

```
MINGW64 ~/Documents/IS-2043/Project-02
$ nano items.txt

MINGW64 ~/Documents/IS-2043/Project-02
$ cat items.txt
bash
scripting
tutorial

MINGW64 ~/Documents/IS-2043/Project-02
$ for i in $( cat items.txt ); do echo -n $i | wc -c; done
4
9
8
```

items.txt character count per line.

While Loop

To do a while loop, I created a file called while.sh and included what the tutorial stated. I then created an executable and ran the script. This script displays the result of the while loop.

```
  GNU nano 4.9.3                          while.sh
#!/usr/bin/bash

counter=0
while [ $counter -lt 3 ]; do
    let counter+=1
    echo $counter
done
```

while.sh.

```
MINGW64 ~/Documents/IS-2043/Project-02
$ nano while.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ chmod +x while.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ ./while
bash: ./while: No such file or directory

MINGW64 ~/Documents/IS-2043/Project-02
$ ./while.sh
1
2
3
```

while.sh execution.

```
  GNU nano 4.9.3                          while.sh
#!/usr/bin/bash

counter=2
while [ $counter -lt 3 ]; do
    let counter+=1
    echo $counter
done
```

while.sh counter edit.

```
MINGW64 ~/Documents/IS-2043/Project-02
$ ./while.sh
3
```

while.sh execution.

Until Loop

To do an until loop, I created a file called until.sh and included what the tutorial stated. I then created an executable and ran the script. This script displays the result of the until loop.

```
  GNU nano 4.9.3                                   until.sh
#!/usr/bin/bash

counter=6
until [ $counter -lt 3 ]; do
    let counter-=1
    echo $counter
done
```

until.sh.

```
MINGW64 ~/Documents/IS-2043/Project-02
$ nano until.sh

MINGW64 ~/Documents/IS-2043/Project-02
$ cat until.sh
#!/usr/bin/bash

counter=6
until [ $counter -lt 3 ]; do
    let counter-=1
    echo $counter
done

MINGW64 ~/Documents/IS-2043/Project-02
$ ./until.sh
5
4
3
2
```

until.sh execution.

```
  GNU nano 4.9.3                                   until.sh
#!/usr/bin/bash

counter=4
until [ $counter -lt 3 ]; do
    let counter-=1
    echo $counter
done
```

until.sh counter edit.

```
MINGW64 ~/Documents/IS-2043/Project-02
$ ./until.sh
3
2
```

until.sh execution.

I then had to edit backup.sh to include a for loop that backups all directories provided to the script on the command line upon its execution and create a backup function that includes every function in the script.

```
  GNU nano 4.9.3
#!/usr/bin/bash

# This bash script is used to backup a user's home directory to /tmp/.

function backup {

    if [ -z $1 ]; then
        user=$(whoami)
    else
        if [ ! -d "/home/$1" ]; then
                echo "Requested $1 user home directory doesn't exist."
                exit 1
        fi
        user=$1
    fi

    input=/home/$user
    output=/tmp/${user}_home_$(date +%Y-%m-%d_%H%M%S).tar.gz

    function total_files {
        find $1 -type f | wc -l
    }

    function total_directories {
        find $1 -type d | wc -l
    }

    function total_archived_directories {
        tar -tzf $1 | grep  /$ | wc -l
    }

    function total_archived_files {
        tar -tzf $1 | grep -v /$ | wc -l
    }

    tar -czf $output $input 2> /dev/null

    src_files=$( total_files $input )
    src_directories=$( total_directories $input )

    arch_files=$( total_archived_files $output )
    arch_directories=$( total_archived_directories $output )

    echo "########## $user ##########"
    echo "Files to be included: $src_files"
    echo "Directories to be included: $src_directories"
    echo "Files archived: $arch_files"
    echo "Directories archived: $arch_directories"

    if [ $src_files -eq $arch_files ]; then
        echo "Backup of $input completed!"
        echo "Details about the output backup file:"
        ls -l $output
    else
        echo "Backup of $input failed!"
    fi
}

for directory in $*; do
    backup $directory
done;
```

backup.sh.