Bash Scripting Day 2 Assignment

Mohamed Emary

January 28, 2025

1 Assignment Questions

After creating all the scripts, give them the execution permission:
`chmod +x scriptname`

1.1 Question 1

> Create a script that asks for the user's name then sends a greeting to them.

```
#!/usr/bin/bash

read name
echo Hello $name
```

1.2 Question 2

> Create a script called `s1` that calls another script `s2` where:
> a. In `s1` there is a variable called `x`, its value is 5.
> b. Try to print the value of `x` in `s2` by two different ways.

Solution 1:

```
############## Inside s1.sh

#!/usr/bin/bash
# Make this as a variable here and source it from the other file
x=5

############## Inside s2.sh

#!/usr/bin/bash
# source s1.sh to get the value of x
source s1.sh
echo $x
```

Solution 2:

```
############## Inside s1.sh

```

```
#!/usr/bin/bash
# Pass the value of x as an argument to the other file
x=5
./s2.sh "$x"

############# Inside s2.sh

#!/usr/bin/bash
# Check if the argument is passed and print it
if [ $# -ne 0 ]; then
  echo $1
fi
```

1.3 Question 3

> Create a script called `mycp` where:
>
> a. It copies a file to another.
>
> b. It copies multiple files to a directory.

```
#!/usr/bin/bash

echo Please select what you want to use this script for:
echo 1. Copying file into another file
echo 2. Copying files into a directory
read opt

if [ $opt -eq 1 ]; then
  echo Enter the name of source file
  read src
  if [ -f $src ]; then
    echo Enter the name of destination file
    read dest
    cp $src $dest
    echo File copied successfully
  else
    echo Source file does not exist
  fi
elif [ $opt -eq 2 ]; then
  echo "Enter your files and the directory you want to copy them to"
  # here we accept multiple arguments in an array
  # then we check if the last argument is a directory
  #
  # there should be more validation on files using loops but i didn't
  # add it because it haven't been covered yet in course content
  read -a args          # -a for array
  dest_dir=${args[-1]} # last argument
  if [ -d "$dest_dir" ]; then
    # "${args[@]::${#args[@]}-1}" is used for array slicing
    # ref https://unix.stackexchange.com/a/82061
    cp "${args[@]::${#args[@]}-1}" "$dest_dir"
```

```
    echo Files copied successfully
  else
    echo "The last argument is not a directory"
  fi
else
  echo Invalid option
fi
```

1.4 Question 4

> Create a script called `mycd` where:
>
> a. It changes the directory to the user's home directory if it is called without arguments.
>
> b. Otherwise, it changes the directory to the given directory.

```
#!/usr/bin/bash

if [ $# -eq 0 ]; then
  cd $HOME
else
  cd $1
fi
```

1.5 Question 5

> Create a script called `myls` where:
>
> a. It lists the current directory if it is called without arguments.
>
> b. Otherwise, it lists the given directory.

```
#!/usr/bin/bash

if [ $# -eq 0 ]; then
  ls
else
  ls $1
fi
```

1.6 Question 6

> Enhance the above script to support the following options individually:
>
> a. `-l`: list in long format.
>
> b. `-a`: list all entries including the hidden files.
>
> c. `-d`: if an argument is a directory, list only its name.
>
> d. `-i`: print inode number.
>
> e. `-R`: recursively list subdirectories.

```
if [ $# -eq 0 ]; then
  ls
elif [ $# -eq 1 -a -d $1 ]; then
  ls $1
elif [ $1 = "-l" ]; then
```

```
  echo "Listing in long format"
elif [ $1 = "-a" ]; then
  echo "Listing all entries including the hidden files"
elif [ $1 = "-d" ]; then
  echo "Listing only the directory name"
elif [ $1 = "-i" ]; then
  echo "Printing inode number"
elif [ $1 = "-R" ]; then
  echo "Recursively listing subdirectories"
else
  echo "Invalid Input"
fi
```

1.6.1 *Bonus*:

> Enhance the above script in question 6 to support the following synopsis:
> - `myls -option1 -option2`
> - `myls -option2 -option1`
> - `myls -option1option2`
> - `myls -option2option1`

```
#!/bin/bash

# No arguments case
if [ $# -eq 0 ]; then
  ls
  exit 0
fi

# Check if first argument is an option
if [ $1 == -* ]; then
  # Get letters after dash
  options=${1:1}
  cmd="ls"

  # Check each letter is valid
  if [ $options =~ [^ladiR] ]; then
    echo "Invalid option"
    exit 1
  fi

  # Add valid options to command
  [ $options == *l* ] && cmd="$cmd -l"
  [ $options == *a* ] && cmd="$cmd -a"
  [ $options == *d* ] && cmd="$cmd -d"
  [ $options == *i* ] && cmd="$cmd -i"
  [ $options == *R* ] && cmd="$cmd -R"

  # Execute with directory if provided
  if [ -n "$2" ]; then
```

```
    $cmd "$2"
  else
    $cmd
  fi
else
  # Just directory argument
  ls "$1"
fi
```

1.7 Question 7

> Create a script called `mytest` where:
>
> a. It checks the type of the given argument (file/directory).
>
> b. It checks the permissions of the given argument (read/write/execute).

```
#!/usr/bin/bash

echo Please select what you want to use this script for:
echo '1. Check the type of the given argument (file/directory).'
echo '2. Check the permissions of the given argument (read/write/execute).'
read opt

if [ $opt -eq 1 ] || [ $opt -eq 2 ]; then
  echo enter your file/dir
  read filedir

  if [ $opt -eq 1 ]; then
    if [ -f $filedir ]; then
      echo $filedir is a file
    elif [ -d $filedir ]; then
      echo $filedir is a directory
    else
      echo $filedir is not a file or directory
    fi
  elif [ $opt -eq 2 ]; then
    if [ -r $filedir ]; then
      echo $filedir has read permission
    fi
    if [ -x $filedir ]; then
      echo $filedir has execute permission
    fi
    if [ -w $filedir ]; then
      echo $filedir has write permission
    fi
  fi
else
  echo Invalid option
fi
```

1.8 Question 8

> Create a script called `myinfo` where:
>
> a. It asks the user about their login name.
>
> b. It prints full info about files and directories in their home directory.
>
> c. It copies their files and directories as much as possible to the `/tmp` directory.
>
> d. It gets their current processes status.

```
#!/usr/bin/bash

echo "Enter your login name: "
read login_name

echo "This is your home directory content: "
ls -alh "/home/$login_name" | less # paging with less

echo -e "\nPress enter to copy a file from your home directory to /tmp"
read
cp "/home/$login_name/.bashrc" /tmp
echo ".bashrc has been copied to /tmp"

echo -e "\nPress enter to show your processes"
read
ps -u $login_name
```

PS: I have copied the `.bashrc` *file only because my home directory is very large.*