

Module #05 - File Directory Task

Step 01 – Creating a file directory relevant to the task.

```
ec2-user@ip-172-31-41-5: ~/fil [ec2-user@ip-172-31-41-5 ~]$ mkdir filedirectory
[ec2-user@ip-172-31-41-5 ~]$ ls
casetask.sh  iftest.sh  task2.sh  test2  textfile.txt
filedirectory  task1.sh  test1  testdir
[ec2-user@ip-172-31-41-5 ~]$ cd filedirectory/
[ec2-user@ip-172-31-41-5 filedirectory]$ echo -e "Line 1\nLine 2\nLine 3" > textfile1.txt
[ec2-user@ip-172-31-41-5 filedirectory]$ echo -e "Line 4\nLine 5\nLine 6" > textfile2.txt
[ec2-user@ip-172-31-41-5 filedirectory]$ echo -e "Line 7\nLine 8\nLine 9" >
textfile3.txt
[ec2-user@ip-172-31-41-5 filedirectory]$ touch img1.png
[ec2-user@ip-172-31-41-5 filedirectory]$ touch img2.png
[ec2-user@ip-172-31-41-5 filedirectory]$ touch img3.png
[ec2-user@ip-172-31-41-5 filedirectory]$ echo "#!/bin/bash\neco 'This is a script.'" > script1.sh
-bash: !/bin/bash\necho: event not found
[ec2-user@ip-172-31-41-5 filedirectory]$ vim script1.sh
[ec2-user@ip-172-31-41-5 filedirectory]$ cat script1.sh
#!/bin/bash

echo "This is the first script file"
[ec2-user@ip-172-31-41-5 filedirectory]$ vim script2.sh
[ec2-user@ip-172-31-41-5 filedirectory]$ cat script2.sh
#!/bin/bash

echo "This is the second script file"
[ec2-user@ip-172-31-41-5 filedirectory]$ mkdir subdir
[ec2-user@ip-172-31-41-5 filedirectory]$ cd subdir
[ec2-user@ip-172-31-41-5 subdir]$ touch unknown.xyz
[ec2-user@ip-172-31-41-5 subdir]$ pwd
/home/ec2-user/filedirectory/subdir
[ec2-user@ip-172-31-41-5 subdir]$ cd ..
[ec2-user@ip-172-31-41-5 filedirectory]$ ls
img1.png  img3.png  script2.sh  textfile1.txt  textfile3.txt
img2.png  script1.sh  subdir      textfile2.txt
[ec2-user@ip-172-31-41-5 filedirectory]$ cd subdir
[ec2-user@ip-172-31-41-5 subdir]$ ls
unknown.xyz
[ec2-user@ip-172-31-41-5 subdir]$
```

The primary directory was created using **‘mkdir filedirectory’**.

Text files with multiples lines were created using **‘echo -e “Line1\nLine2\nLine3” > textfile1.txt’**. In here, **‘-e’** was used to allow the interpretation of **‘\n’** as a newline character. After executing this code line, the textfile1.txt file will contain three multiple lines as below.

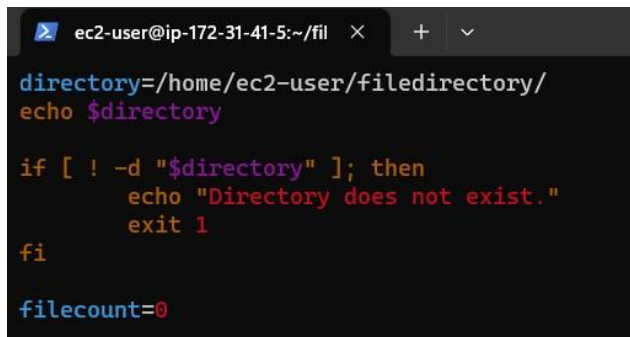
```
ec2-user@ip-172-31-41-5: ~/fil Line 1
Line 2
Line 3
..
..
..
..
..
..
..
```

Multiple image files were created using **‘touch img1.png’** respectively.

The script files were created using vim editor. **‘vim script1.sh’**.

Finally, a file with unknow extension was created. **‘touch unknown.xyz’**.

Step 02 – Creating the script file to fulfil the requirements.



```
ec2-user@ip-172-31-41-5:~/fil × + v
directory=/home/ec2-user/filedirectory/
echo $directory

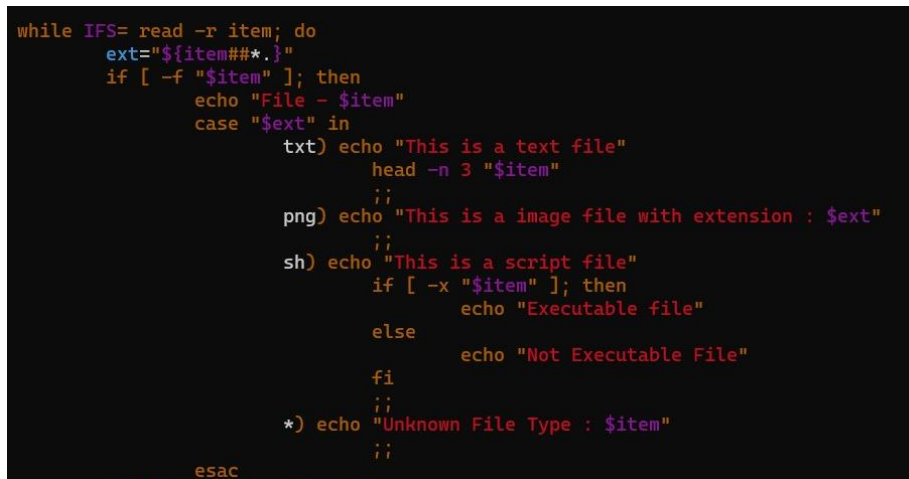
if [ ! -d "$directory" ]; then
    echo "Directory does not exist."
    exit 1
fi

filecount=0
```

In the first two lines, Assigning the file directory path to the created variable named **'directory'** and printing out the variable.

In the below if statement using **'! -d'**, The file directory is validated. If the file directory is not existing the code will print **"Directory does not exist"** and exiting the code with **'exit 1'** (Exiting the code with errors) or else the code will continue to the next step.

In the last code line **'filecount'** variable is initialized with value 0.



```
while IFS= read -r item; do
    ext="${item##*}"
    if [ -f "$item" ]; then
        echo "File - $item"
        case "$ext" in
            txt) echo "This is a text file"
                  head -n 3 "$item"
                  ;;
            png) echo "This is a image file with extension : $ext"
                  ;;
            sh) echo "This is a script file"
                  if [ -x "$item" ]; then
                      echo "Executable file"
                  else
                      echo "Not Executable File"
                  fi
                  ;;
            *) echo "Unknown File Type : $item"
               ;;
        esac
    fi
done
```

To iterate through the file directory a while loop has been used as instructed.

IFS stands Internal Field Separator. The IFS is used to determines how words are split into fields. By setting it to an empty string (IFS=), it ensures that leading and trailing whitespaces are preserved during the read operation.

The file extension will be saved in to user variable **'ext'** for file manipulation task. When this line is executed, it takes the value stored in the variable **'item'**, removes everything up to and including the last dot, and assigns the remaining part (the file extension) to the variable **'ext'**.

In the first if condition inside the while loop, 'if [-f "\$item"]' it's check whether the '\$item' contains a file. If true, the other code lines will execute. Else if it will check whether it is a directory using 'elif [-d "\$item"]' or else the script will print 'Unknown : File path'.

To manipulate the files a case statement is used. The case statement will check for the cases of file extensions (**value in ext**) as mentioned above.

If the (**ext = txt**) it will print the top 3 lines of the text file by executing the 'head -n 3 "\$item"' line. The head function used to get the top 3 lines. The tail function can be used to get the bottom lines.

As per the requirement the image files will be identified using file extension (**ext = png**). And a message will be displayed as "This is an image file with extension : \$ext".

The script files are also identified with the extension (**ext = sh**), And the identified script file is checked whether they are executable or not using 'if [-x "\$item"]'.

```
elif [ -d "$item" ]; then
    echo "Dir - $item"
    # $directory "$item"
else
    echo "Unknown : $item"
fi
filecount=$((filecount+1))
done <<< "$(find "$directory" -maxdepth 1)"
echo "No. of Processed Files : $filecount"

"processfile.sh" 41L, 821B
```

As per the requirement of the task, to access a subdirectory has failed.

The filecount was gradually incremented by after each iteration using arithmetic operation 'filecount=\$((filecount+1))'.

"\$(find "\$directory" -maxdepth 1)" This find command is searching for files and directories in the specified directory (**\$directory**) with a maximum depth of 1. The result is a list of file and directory paths. The result string will be passed as an input to the loop.

Finally, the no. of processed files will be displayed using the 'filecount' variable.

Step 03 – Visualizing the results of the script.

```
ec2-user@ip-172-31-41-5:~/fil x + v
[ec2-user@ip-172-31-41-5 filedirectory]$ touch unknown.xyz
[ec2-user@ip-172-31-41-5 filedirectory]$ ls
img1.png img2.png img3.png processfile.sh script1.sh script2.sh subdir textfile1.txt textfile2.txt textfile3.txt unknown.xyz
[ec2-user@ip-172-31-41-5 filedirectory]$ ./processfile.sh /home/ec2-user/filedirectory
/home/ec2-user/filedirectory/
Dir - /home/ec2-user/filedirectory/
File - /home/ec2-user/filedirectory/textfile1.txt
This is a text file
Line 1
Line 2
Line 3
File - /home/ec2-user/filedirectory/textfile2.txt
This is a text file
Line 4
Line 5
Line 6
File - /home/ec2-user/filedirectory/textfile3.txt
This is a text file
Line 7
Line 8
Line 9
File - /home/ec2-user/filedirectory/img1.png
This is a image file with extension : png
File - /home/ec2-user/filedirectory/img2.png
This is a image file with extension : png
File - /home/ec2-user/filedirectory/img3.png
This is a image file with extension : png
File - /home/ec2-user/filedirectory/script1.sh
This is a script file
Not Executable File
File - /home/ec2-user/filedirectory/script2.sh
This is a script file
Not Executable File
Dir - /home/ec2-user/filedirectory/subdir
File - /home/ec2-user/filedirectory/processfile.sh
This is a script file
Executable file
File - /home/ec2-user/filedirectory/unknown.xyz
Unknown File Type : /home/ec2-user/filedirectory/unknown.xyz
No. of Processed Files : 12
[ec2-user@ip-172-31-41-5 filedirectory]$
```

Challenges

- Looping through the file directory using the while loop.
- Finding a method to manipulate different types of files.

Improvements

- During the task it was observed that the different file types can be identified using MIME type. MIME stands for Multipurpose Internet Mail Extensions. Common MIME types which will be useful in this task are;
 - text/plain for plain text files
 - image/jpeg for JPEG images
 - application/x-sh for bash scripts