Sourik Dhua

Program logic:

First the script copies the “ls -a” output to a file called lsfile.txt

Reason: In Linux, I named some files with spaces. In the script, I used ‘ls -a’ and directly looped through its output to find that filenames with space are treated as different files. So, to prevent, that I am creating this temporary file. Also, this file will be deleted at end of script.

Then to check errors such as invalid extensions(given folder should have web safe image files), the script separates the extension and checks if it is a valid one. I have used an array containing a set of valid extensions. This can be modified as required.

The script allows/processes filenames without extensions and private files(dot at beginning) as well as these are websafe.

Then I have truncated the filenames as required using tr. I made them lowercase and allowed only _ (as we are replacing space by underscore so I am not removing already existing ones)and ‘.’ As dot is required for extensions. Also, spaces are replaced by “_”.

After truncating, if file exists, three possibilities:

1. possibility is to ask user to overwrite but for large files with same truncated name, it is not practical(100 file with same truncated names).
2. This can be solved by asking if the user wants to overwrite all files with same modified names. So, ask the user once. But in this case, we have to create another dictionary which keeps track of yes or no for a particular file. Every time a file is in this case, it checks the dictionary if user had input yes or no and accordingly perform functionality.
3. Keep all of them (no user input) and rename them accordingly. Since, the folder has all image files needed for web page, overwriting might not be required(assuming all images are different), so I went with this.

Implementation of the 3rd possibility:

The dictionary called filenumbers maintains key as filename to their count as value. Every time a filename is modified, we check if it exists. If it does not exist, then it is directly renamed.

If it exists, then we increment the count. For example, after truncating, abcd.jpg is encountered a second time, so it will rename it as abcd1.jpg and store in dictionary as “abcd.jpg”:1 When it is encountered for third time, we take out the value for “abcd.jpg” from dictionary and add 1. So, for this, the value for abcd.jpg will be 2 and hence, file is renamed as abcd2.jpg. Then the pair abcd.txt:2 is stored in dictionary and it goes on like this. We do this till the filename is unique.

In this situation, I do the renaming based on if it is a private file, file with no extension with dot at end (required for not generating invalid extensions), file with no extensions and no dots or file with usual format: filename.ext. So, a private file: .image and .imag#e will be renamed as .image and .image1.

A file named resume one, res@ume one will be renamed as resume_one, resume_one1.

Finally I put an error handling to mv command used for renaming. In case, mv is restricted an error is thrown.

At end of script, lsfile.txt is deleted.

Testing:

First part : Sanity check

I have tested the script by creating files in a folder with the script inside.

First, I created certain documents to make sure the initial conditions are working right

Test1: I put in a couple of directories inside the main folder; the script displays that these directories are not files and hence, does not process it

Test2: Condition1: To check if it discard invalid image extensions , I put a few of other documents like abcd.html , ab cd.doc, abcd.jpeg. For the first two, it threw an error requesting to give proper extension for image files and processed the third one as it had jpeg extension.

Test3: Condition2: To check if it processes files starting with dot, put a few files with dot like .image#1, … , .br@@$c

For directory files which occur in "ls -a" such as "." And "..", it displays "NOT A FILE"

Regarding dots, multiple dots are fine in a filename(web safe) given extension is valid , I tested on my.resume.txt.

Test4: Condition3: To check if files with no extensions are used correctly, I put files such as im__g1, img2

In case all three conditions fail, it throws an ERROR: asking to put proper image extension

I checked if this part of the code is working properly.

Analysis :

Allowed extensions: jpeg, jpg, pngapng, bmp, gif, ico, svg, tiff, webp

I have entered the above list as valid in an array named image_valid_ext in the script. All extensions except these will throw an error requesting user to put proper file extension.

Note: Since files like abc.html(invalid image extension) will throw an error with a request to change extension, even private files like .brc.html will do the same

Due to this, image files named file1.1 and filename.22 will also throw error (although these are supposed to be websafe). I tested this by renaming a image file and running on mozilla firefox. As expected extension with html, doc, pdf did not result in correct output. But files like imag1.1 image.22 did show correct output.

But to allow files like image1.1 and block files like image.html or image.csv, we need to know exactly which extensions are allowed or which are not for image files.

For example, if I allow file1.1, that means abc.html will also be allowed (since I do not know the exact extensions to allow or prevent) giving a false result.

So, throwing an error requesting user to put a proper extension is a safer and better method and hence, I went with it. I have kept image_valid_ext array where extensions can be added and deleted easily according to requirements.

Second part:

1.Truncating filename :

Following are some test cases and their outputs.

Tests:

abcde.jpg → no change

ab@#cd*.jpg → abcd.jpg (special char)

AB.jpg → ab.jpg (capital)

my file .jpg → my_file_.jpg (spaces)

hello.jp#g → error: put proper image extension (wrong extensions)

… → no change

Fil#e.. jpeg → file..-jpeg (capital, special char, space combination)

.bsh@rc →.bshrc (dot at beginning like private files)

ö,bc → bc(different char set)

.bash.rc# → error: put proper image extension (private files with incorrect extensions)

@@@>.svg → .svg (special cases)

maized. → maized.

.jpg → no change

Hi%20there.tiff → hi20there.tiff

$$$ → 1 (all invalid char)

@#@. → 1

test.sh. → no change

.sourik. → no change

2.handling if filename is already existing

To test this: I put files with same name for example resume.jpg, resu#me.jpg, RESUME.jpg ➔ resume.jpg, resume1.jpg, resume2.jpg

Resume, res$u##me, resume1, resume 1 → resume, resume2, resume1, resume-1 (multiple names without ext)

.brc, .br$c → .brc, .brc1 (multiple same name starting from .)

res$u##me, resume1, resume 1, .brc, .br$c → resume, resume2, resume1, resume-1, .brc, .brc1 (having a set of multiple files with matched truncated name and another set of files having different matched name)

@@@.jpg and ###.jpg →.jpg, .jpg1

@@@1.jpg and ###1.jpg → 1.jpg and 11.jpg

… , #.$.. → … , ..1. (all dots)

$$$, ##, %% → 1 , 11, 12 (special handlingmultiple instances with all invalid char is turned to 1 and then more such instances are 1<count> such as 11,12,13 etc.)

$$$., ##. , ^%^.. → 1, 11, 12

NOTE: filenames which truncate to empty string or special filenames like '.' Or '..' or ''(empty) is named as 1 initially and later instances as 11, 12, 13… to make them valid names. (as shown in above example)

NOTE: In case mv is restricted as was in case of empty filename, '.' And '..', the script throws an error. I handled the special cases accordingly.

test.sh. , te#st.sh., test., t@#est. → test.sh. , test.sh1. , test. , test1.

Analysis: For the above test case, initially, test.sh1. would be test.sh1 (eliminating last dot), which seemed to be fine. But in case the script is run again, it would throw error saying invalid extension 'sh1'. So, I changed the naming convention putting the count before dot.

maized. , maiz!ed. , maized&&.–> maized. , maized1., maized2. (dot at end)

.sourik. , .sou@r$ik. , .sourik>. → .sourik. , .sourik1. , .sourik2. (dots on both start and end)

resume1.jpg, resume@1.jpg -> resume1.jpg, resume11.jpg (count is always added after truncated filename: resume1 in this case)