DO NOT DISTRIBUTE WITHOUT WRITTEN PERMISSION FROM PROF. XIAONING DING

Search the files that satisfy certain requirements

Write a bash script that searches in a directory and its subdirectories for the files that are owned by a specific user and have read permissions for all users. The script takes two arguments. The first argument is the pathname of the directory and the second argument is a user id.

Note that your script needs to traverse the directory and check the files under its subdirectories, sub-subdirectories, etc. For the purposes of practice, DO NOT use `find` command, and DO NOT use the built-in option `-R` in `ls` commands (you can use `ls` command and its options other than `-R`). During the traversal, for each file (assuming file name saved in variable `filename`), your script need to 1) use command `ls -l ${filename}` to get the information of the file, 2) analyze the permission field and the owner field of the line generated by the above `ls` command using `grep` or `expr` and determine whether the file satisfies the requirements or not, and 3) if the file satisfies the requirements, print out the following information of the file:

- file name
- permissions (a group of 9 characters consists of r, w, or x, do not include the character for file type at the beginning of the line)
- time of creation or last modification.

For the format of the information printed out by ls -l, refer to these pages: https://cr.yp.to/ftp/list/binls.html, https://linuxize.com/post/how-to-list-files-in-linux-using-the-ls-command/ . Check the owner field and the read permissions to determine whether a file satisfies the requirements. For a file having read permissions for all users, `ls -l ${filename}` shows three "r" in the permission field.

When you extract the time of creation or last modification, your code should be flexible to handle two time formats: month+day+hour+minute for files modified/created within the last six months, and month+day+year for other files.

To use `grep` to process the line printed out by `ls -l`, you can use a pipe to connect `ls` command and `grep` command. For example, the following commands extracts the first field (- and permissions).

```
$ ls -l /bin/bash | grep -o '^.\{10\}'
-rwxr-xr-x
```

This exercise is for you to practice the use of regular expressions. DO NOT use commands find and cut in your script. To extract the desired information from a string, consider to use sub-string, and `grep -o`. This is particularly useful when extracting the modification/creation time field from the ls output. For example, to extract the second field from the ls output, you can first extract using regex and remove the first field using sub-string, and then use another grep -o and regex, which describe the pattern of the second field.

```
$ ls_output=`ls -l /bin/bash`
$ first_field=`echo "${ls_output}" | grep -o '^.\{10\}'`
$ len=${#first_field}
$ remaining=${ls_output:len+1} #cut the 1st field off
$ second_field=`echo "$remaining" | grep -o '^[0-9]\{1,\}'`
$ echo $second_field
1
```

Note: 1. The following trick can simplify the regular expressions used in grep -o: cut off the substring before the field that you want to extract and use the ^ anchor to match from the beginning of the

DO NOT DISTRIBUTE WITHOUT WRITTEN PERMISSION FROM PROF. XIAONING DING

remaining part. With this method, you only need to design a regular expression that matches the field that you want to extract; (you may leverage the spaces between fields;) you don't need to refine the regular expression to exclude the matches in other parts of the string (e.g., in the middle).
2. Use double quotes to enclose `$var` in echo command. Otherwise, the output of echo may not exactly match the contents in `var`. For example, if there are two consecutive spaces in `var`, without double quotes, `var` will be divided at the spaces, and the parts are printed out with a single space in between.

3. Escape ("\") parentheses and braces if you use BRE.

Testing: to test the script, run it with "root" and "/usr/share/docutils/writers/" as arguments, you should see the information of all 36 files. Randomly select a few of these files, and manually check whether the information printed out by your script matches the corresponding information printed out by *`ls -l`*. Run your script with your own username and "/usr/share/docutils/writers/" as arguments. You should not see any information printed out because you don't have any files in that directory.