System Commands

OPPE 1 Set 1 and Set 2 Combined

25 Jun 2022 [Set 1]

Section 1 [Attempt any 3]

Question 1 [15 mark]

Write a Bash script that reads two inputs from stdin.

- The first line is a directory name present in the current directory.
- The second line is a filename.

Print `present` if the file with filename given as second input is present in the directory name given as the first input else print `absent`.

Hint: Refer the cheat sheet for file operators.

Test case description: Input are the two lines of input that your script should read. Output is the output printed by your script.

Solution

```
read dir
read file
[[ -a "$dir/$file" ]] && echo present || echo absent
```

Question 2 [15 marks]

Write a Bash script that reads an integer value (say `n`) from the standard input, creates a directory named `image_files`(in the current working directory) then creates `n` empty files inside the directory `image_files` each file with the name as `image_i_rgb.txt` , where `i` is the number of the file between `1` to `n`. (n>=1) For example, if `n` is given as 3, the following three files should be created in the directory `image_files`: `image_1_rgb.txt`, `image_2_rgb.txt` and `image_3_rgb.txt`. Initially the directory `image_files` does not exist in the current working directory.

Test Case description: Input is the value of n and output is the the sorted list of files created. Printing of output is taken care by evaluation script, your script does not need to print anything.

Solution

```
mkdir image_files
cd image_files
read n
for i in $(seq 1 $n);
do
    touch "image_${i}_rgb.txt"
done
```

Question 3 [15 marks]

Write a Bash script that reads an integer value from STDIN, that is the length of the side of a square and prints the **area** and **perimeter** of this square. Print only the numeric values of the area and perimeter on first and second line respectively.

Test Case description: Input is an integer that your script should read. Output is the output printed by your script.

Solution

```
read len
area=$(($len*$len))
echo $area
peri=$((4*$len))
echo $peri
```

Question 4 [15 marks]

Create the file/directory structure as shown below. The current working directory is the root(`/`) directory.

```
/potato_recipes
├── french_fries
│   ├── ingredients.txt
│   └── procedure.txt
├── frites -> /potato_recipes/french_fries/
└── potato_wedges
    ├── ingredients.txt
    └── procedure.txt
```

- `potato_recipes` is a directory in the root(`/`) directory.
- `french_fries` and `potato_wedges` are the directories present inside the directory `potato_recipes`.

- `frites` is a symbolic link to the directory `french_fries`.
- `procedure.txt` and `ingredients.txt` are the text files present inside directories `french_fries` and `potato_wedges`.

Test Case Description: No input is required here. Output is the list of directories and subdirectories under the directory 'potato_recipes'. Printing output is taken care by the evaluation script, your script does not need to read input or print any output here.

Solution

```
mkdir -p /potato_recipes/{potato_wedges,french_fries}
touch /potato_recipes/{french_fries,potato_wedges}/{ingredients,procedure}.txt
ln -s /potato_recipes/french_fries/ /potato_recipes/frites
```

Section 2 [Attempt any 3]

Question 1 [20 marks]

There are several files in the current working directory. Write a Bash Command/Script to print the names of all files that ends with the extension `.txt` and contains greater than 10 lines of text. Hint: Use a combination of commands `wc` and `cut` or combination of `cat` and `wc` to get only the numeric value of the count of lines in the file.

Test Case Description: Each line of input contains the name of the file and number of lines of text in it, line 'EOF' marks the end of input. Reading Input and using it to create files is taken care by evaluation script, your script does not need to read any input. Output is the filenames printed by yor script.

Solution

```
for file in *.txt; do
 if [ `wc -l $file | cut -d " " -f 1` -gt 10 ]; then
  echo $file
 fi
done
```

Question 2 [20 marks]

Consider a shell variable array named as `number_arr` containing integers. This array is already created. Write a Bash script that prints the sum of all odd positioned integers in the array `number_arr`. i.e. sum of 1st, 3rd, 5th, 7th and so on, elements of the array.

Test case description: Input is a list of integers, that are the elements of the array in the same sequence. Reading input and creating array is handled by the suffix code. last line in the input marks the end of file. Your script does not need to read any input.

Output is output printed by your script.

Solution

```
sum=0
flag=1
for i in "${number_arr[@]}"; do
 if [ $flag -eq 1 ]; then
  sum=$(($sum+$i))
  flag=0
 else
  flag=1
 fi
done
echo $sum
```

Question 3 [20 marks]

The current working directory contains different types of files, each file type is identified by its extension. For example files `prog.cpp` and `movie.avi` have extensions `cpp` and `avi` respectively.

Write a Bash script to move all the files in the current working directory to the respective destination directories as mentioned in the table below. Currently there are no sub directories in the current working directory. Create all destination directories directly under the current working directory.

Note: There is only one `.` in each filename. And the first letter of each directory name is capital.

Destination Directory	Extensions
Image	jpg, png, gif, tiff
Video	mov, mp4, mkv, avi
Program	sh, fish, py, c, cpp, java, perl, js
Other	No extension or any other extension that is not specified above

Test case description: Input is the list of files in the current workind directory. Reading input and creating files is done by evaluation script, your script does not need to read any input. Output is the list of directories and files in the current working directory, that is printed using evaluation script. Your script does not need to print anything, just move the files as described.

Solution

```
for file in *; do
    [ -d "$file" ] && continue
    case "${file##*.}" in
        jpg|png|gif|tiff)
            mv $file Image
            ;;
        mov|mp4|mkv|avi)
            mv $file Video
            ;;
        sh|fish|py|c|cpp|java|perl|js)
            mv $file Program
            ;;
        *)
            mv $file Other
            ;;
    esac
done
```

Question 4 [20 marks]

The file `dpkg.log` present in the current directory contains the log of package operations. Write a Bash command/script to print all the unique package names installed on 26th of May 2022. Print only unique package names in sorted order one on each line. Check the public test case for the format of information contained in `dpkg.log`. Each line in the log file is associated with one package operation. For the installed packages the word `installed` is logged after `status`. For install package log line the package name is contained in the 5th field(if separated by space). In the 5th field the package name is the string from start till the character before `:`. The date and time are denoted by the first two fields of the log line.

For example: The three lines of the logs below gives information that only two packages were installed on `2022-05-26`, with names as `accountsservice` and `gir1.2-accountsservice-1.0`. Note in the example below that `half-installed` does not qualify as installed.

```
2022-05-26 10:23:53 status installed accountsservice:amd64 22.07.5-2ubuntu1.3
2022-05-26 10:23:53 trigproc dbus:amd64 1.12.20-2ubuntu4 <none>
2022-05-26 10:23:44 status installed gir1.2-accountsservice-1.0:amd64 22.07.5-
2ubuntu1.3
2022-05-26 10:24:02 status half-installed gir1.2-webkit2-4.0:amd64 2.36.0-2ubuntu1
2022-05-27 11:13:26 status installed logrotate:amd64 3.19.0-1ubuntu1.1
```

Hint: Use a combination like `Your command | sort | uniq` to print distinct values in sorted order. `sort` command will sort the results from `Your command` and `uniq` command will select select distinct values from sorted results.

Test case description: Input is the contents of the log file. Reading input is taken care by the evaluation script, your script does not need to read the input. Output is the names of packages printed by your script.

Solution

```
grep 2022-05-26 dpkg.log \
    | grep " installed " \
    | cut -d " " -f5 \
    | cut -d ":" -f 1 \
    | sort | uniq
```

---

26 Jun 2022 [Set 1]

Section 1 [Attempt any 3]

Question 1 [15 mark]

Write a Bash script that reads two numbers `a` and `b` as first and second input respectively from standard input and print `special` if they satisfy the two conditions below,

- `a` and `b` both are divisible by 3
- `a` is greater than `b`

If any of the above condition is not satisfied then print `not-special`.

Test case description: Input are the two numbers that your script should read. Output is the output printed by your script.

Solution

```
read a
read b
[[ $a -gt $b \
&& $((a%3)) -eq 0 \
&& $((b%3)) -eq 0 ]] && echo special || echo not-special
```

Question 2 [15 marks]

Current working directory contains two sub-directories named as `source` and `destination` . There are an unknown number of files in the directory `source` and some of these files are named in the format `image_X_rgb.txt` where `X` is an integer or string(can be empty), some example filenames in this format are `image_1_rgb.txt`, `image_config_rgb.txt`, `image_23_rgb.txt`, `image_ _rgb.txt` etc.

Write a Bash Command/Script to move all the files in the directory `source` having name in the format described above to the directory named `destination`.

Test case description: Input is the names of the files in the 'source' directory. Reading input is taken care by evaluation script. Your script does not need to read any input or print any output. Output is the list of files in the 'destination' directory, printing output is taken care by evluation script.

Solution

```
mv source/image_*_rgb.txt destination/
```

Question 3 [15 marks]

Write a bash command to print just the `n`th line of a given text file named `file.txt`, `n` is a bash variable which is already assigned an integer value and can be used in your command. `file.txt` is located in the current working directory. Assume that n<= number of lines in the file.

For example if `n`=4, print only the 4th line in the file `file.txt`.

Test case description: Input contains the contents of the file till the line before second last line, second last line is the end of file marker and the last line is the value of shell variable `n` . Reading of input is taken care by the evaluation script, your script need not read any input. Output is the output printed from your script, i.e. the required line from file.

Solution

```
head -$n file.txt | tail -1
```

Question 4 [15 marks]

Write a Bash Command/script that creates a file named `documents.txt`(in the current working directory) containing all the possible file names in the format `file_XYZ.txt` where X is a lower case alphabet, Y is also a lower case alphabet and Z is a number between 0 and 4. Few examples of file names in this format are 'file_dh3.txt', 'file_sd1.txt', 'file_ja0.txt', 'file_at2.txt'.

Hint: Use echo to solve this with a single command.

Test case description: Input is some marker used by evaluation script. Your script does not need to read any input here. For public test case the output is the top 100 and last 100 filenames(after sorting) from the file created by your script as per the problem statement. Your script does not need to print anything to STDOUT, it just needs to generate the required file `documents.txt` as described.

Solution

```
echo file_{a..z}{a..z}{0..4}.txt > documents.txt
```

Section 2 [Attempt any 3]

Question 1 [20 marks]

Write a bash script that takes any number of command line arguments.

- Checks if all the arguments are positive integers, then prints two integer values one on each line, as below
  - First value is the sum of all odd numbered arguments that is 1,3, $5 etc.
  - Second value is the sum of all even numbered arguments that is 2,4, $6 etc.
- Else, if any of the arguments is not a positive integer then prints the word `Error` and exits with exit code `255` .

Hint: Use regular expression to check if an argument is a positive integer or not.

Test case description: Input is the set of arguments that will be read by evaluation script and passed to your script as arguments. Your script does not need to read the input. Output is whatever you script prints. In case of error, additional to your scripts output exit code is also printed through evaluation script.

Solution

```
count=0
oddsum=0
evensum=0
neg='^-'
zero='^0\.?0*$'
for i in "$@"; do
  ((count++))
  if $(echo $i | egrep -q '^[[:digit:].]+$'); then
  (( count%2 == 1 )) && oddsum=$((oddsum+i)) || evensum=$((evensum+i))
  else
    echo Error && exit 255
  fi
done
echo $oddsum
echo $evensum
```

Question 2 [20 marks]

Write a Bash script to print all the numbers between `a` and `b` (both `a` and `b` inclusive), that are divisible by both 7 and 11. `a` and `b` both are the shell variables, each already assigned an integer value such that `a` is less than `b`. Your output should contains numbers in sorted order one on each line.

Hint: Use seq command to generate a sequence of values.

Ex: `seq 0 4` will give

```
0
1
2
3
4
```

Test case description: Input contains two integers which are read by suffix code and stored in shell variables `a` and `b`. Your script does not need to read the input. Output is the output from your script.

Solution

```
    for i in $(seq $a $b); do
        [[ $((i%7)) -eq 0 \
        && $((i%11)) -eq 0 ]] \
        && echo $i
    done
```

Question 3 [20 marks]

Consider a bash script named as `message.sh` that prints a message to STDOUT. This script is located in the current working directory, and you are the owner of this script. You do not know if you have execute permission on this script file.

Write a Bash script that perform the below actions.

- If you have execute permission on the script `message.sh`, then
    - run the script `message.sh`,
    - and **append**(at the end) the output from the script to a file named `messages.log` , this log file already exists and is located in the current working directory.
- If you do not have execute permission on the script file `message.sh`, then
    - add the execute permission for owner on the script `message.sh`, and then run this script.
    - and also redirect the output from the script to a new file named `newMessages.log`.

Test case description: Input is a random integer that is read and used by evaluation script. Public test case one is when owner has execute permission on the script, and second test case is when owner does not have execute permission on the script. Output is printed by evaluation script, which is the contents of the file `messages.log` or `newMessages.log` according to the test case. Your script does not need to read the input or print any output.

Solution

```
if [ -x message.sh ]; then
 bash message.sh >> messages.log
else
 chmod u+x message.sh
 bash message.sh > newMessages.log
fi
```

Question 4 [20 marks]

The file `dpkg.log` present in the current working directory contains the log of package operations from 27th May to 30th May. Check the public test case input for the format of information contained in `dpkg.log`. Each line in the log file is associated with one package operation. For the installed packages the word `installed` is logged after `status`. When the package is uninstalled the word `not-installed` is logged after `status`. For install package log line the package name is contained in the 5th field(if separated by space). In the 5th field the package name is the string from start till the character before `:`. The date and time are denoted by the first two fields of the log line.

For example: In the three lines of the logs below gives information of two packages installed, named as `accountsservice`

```
2022-05-26 10:23:53 status installed accountsservice:amd64 22.07.5-2ubuntu1.3
2022-05-26 10:23:53 trigproc dbus:amd64 1.12.20-2ubuntu4 <none>
2022-05-26 10:23:44 status not-installed accountsservice:amd64 22.07.5-2ubuntu1.3
```

Shell variable `pkg` stores a package name. Write a Bash Command/script to print

- `"installed"` if the package with the name given in `pkg` was still installed on 30th May.
- else print `"not-installed"`. (i.e. if the package with the name given in `pkg` was 'not installed at all' or 'uninstalled after successfully installing' between 27th May to 30th May.)

Hint: The latest line with

1. `YYYY-MM-DD HH:MM:SS status installed <pkg>:<platform> <installed-version>` indicates the status installed for the package `<pkg>`.
2. `YYYY-MM-DD HH:MM:SS status not-installed <pkg>:<platform> <installed-version>` indicates the status not installed (an attempt is made before and/or uninstalled) for the package `<pkg>`.

Test case description: Input is the contents of the log file. Reading input is taken care by the evaluation script, your script does not need to read the input. Output is output printed from your script.

Solution

```
egrep "status (not-)?installed $pkg" dpkg.log \
    | tail -1 \
    | cut -d " " -f 4
```