

Linux administration with bash. Home task

A. Create a script that uses the following keys:

1. When starting without parameters, it will display a list of possible keys and their description.
2. The --all key displays the IP addresses and symbolic names of all hosts in the current subnet
3. The --target key displays a list of open system TCP ports.

The code that performs the functionality of each of the subtasks must be placed in a separate function

Here is our script:

```
#!/bin/bash
```

```
# function to display list of possible keys and their description
```

```
display_keys() {
```

```
    echo "Possible keys and their description:"
```

```
    echo "--all: Displays the IP addresses and symbolic names of all hosts in the current subnet"
```

```
    echo "--target: Displays a list of open system TCP ports"
```

```
}
```

```
# function to display IP addresses and symbolic names of all hosts in the current subnet
```

```
display_all() {
```

```
    echo "IP addresses and symbolic names of all hosts in the current subnet:"
```

```
    arp -a
```

```
}
```

```
# function to display a list of open system TCP ports
```

```
display_target() {
```

```
    echo "List of open system TCP ports:"
```

```
    netstat -tuln | grep -E '^tcp'
```

```
}
```

```
if [ $# -eq 0 ]; then
```

```
    display_keys
```

```
else
```

```
    while [ $# -gt 0 ]; do
```

```

case $1 in
    --all)
        display_all
        shift
        ;;
    --target)
        display_target
        shift
        ;;
    *)
        echo "Invalid key. Use --help for a list of possible keys"
        exit 1
        ;;
esac

done

fi

```

This script starts by defining three functions: `display_keys`, `display_all`, and `display_target`. The first function will display the possible keys and their descriptions when the script is run without any parameters. The second function will use the `arp` command to display the IP addresses and symbolic names of all hosts in the current subnet. The third function will use the `netstat` command to display a list of open system TCP ports.

Then the script checks if any parameter has been passed or not. If no parameter passed it calls the `display_keys` function else it uses a while loop to iterate over the passed parameters, and a case statement is used to check if the parameter is `--all` or `--target` and calls the corresponding function.

Here is the output on my VM:

```

ubuntu18@server-1:~/homework$ ./script1.sh
Possible keys and their description:
--all: Displays the IP addresses and symbolic names of all hosts in the current
subnet
--target: Displays a list of open system TCP ports
ubuntu18@server-1:~/homework$ ./script1.sh --all
IP addresses and symbolic names of all hosts in the current subnet:
? (10.15.86.10) at 08:00:27:07:c8:48 [ether] on enp0s9
? (10.86.15.10) at 08:00:27:07:c8:49 [ether] on enp0s8
_gateway (192.168.0.1) at 34:60:f9:19:06:ed [ether] on enp0s3
ubuntu18@server-1:~/homework$ ./script1.sh --target
List of open system TCP ports:
tcp      0      0 127.0.0.53:53          0.0.0.0:*             LISTEN
tcp      0      0 0.0.0.0:22            0.0.0.0:*             LISTEN
tcp      0      0 127.0.0.1:631         0.0.0.0:*             LISTEN
tcp6     0      0 :::22                 :::*                  LISTEN
tcp6     0      0 :::1:631              :::*                  LISTEN

```

B. Using Apache log example create a script to answer the following questions:

1. From which ip were the most requests?
2. What is the most requested page?
3. How many requests were there from each ip?
4. What non-existent pages were clients referred to?
5. What time did site get the most requests?
6. What search bots have accessed the site? (UA + IP)

Here is our script:

```
#!/bin/bash
```

```
# set the log file location
```

```
log_file='/home/ubuntu18/homework/example_log.log'
```

```
# 1. From which ip were the most requests?
```

```
echo "IP address with the most requests:"
```

```
grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" $log_file | sort | uniq -c | sort -nr | head -n 1
```

```
# 2. What is the most requested page?
```

```
echo "Most requested page:"
```

```
grep -oE "[A-Z]{3,4} [^\"]+" $log_file | awk '{print $2}' | sort | uniq -c | sort -nr | head -n 1
```

```
# 3. How many requests were there from each ip?
```

```
echo "Number of requests from each IP:"
```

```
grep -oE "\b([0-9]{1,3}\.){3}[0-9]{1,3}\b" $log_file | sort | uniq -c | sort -nr
```

```
# 4. What non-existent pages were clients referred to?
```

```
echo "Non-existent pages:"
```

```
grep "404" $log_file | awk '{print $7}' | sort | uniq -c | sort -nr
```

```
# 5. What time did site get the most requests?
```

```
echo "Time with the most requests:"
```

```
grep -oE "[[:digit:]]{2}/[[:alpha:]]{3}/[[:digit:]]{4}:[[:digit:]]{2}:[[:digit:]]{2}" $log_file | awk -F: '{print $2}' | sort | uniq -c | sort -nr | head -n 1
```

6. What search bots have accessed the site?

```
echo "Search bots that accessed the site (UA + IP):"
```

```
while read line; do
```

```
  if [[ $line =~ (bot|crawler|spider) ]]; then
```

```
    ip=$(echo $line | awk '{print $1}')
```

```
    ua=$(echo $line | awk -F\" '{print $6}')
```

```
    echo "IP: $ip, UA: $ua"
```

```
  fi
```

```
done < <(grep -oE "[^ ]+\"[^\"]+\" \"$log_file)
```

1. To find the IP address with the most requests, we can use *grep* to extract all the lines containing IP addresses, then use *sort* to sort the lines by IP address, and finally use *uniq* with the *-c* option to count the number of occurrences of each IP. The IP address with the highest count will be the one that made the most requests.
2. To find the most requested page, we can use *grep* to extract all the lines containing the request method and the requested page, then use *sort* to sort the lines by page, and finally use *uniq* with the *-c* option to count the number of occurrences of each page. The page with the highest count will be the most requested.
3. To find the number of requests from each IP, we can use the same method as in question 1, but instead of sorting by IP, we can sort by the count number.
4. To find non-existent pages, we can use *grep* to extract all lines containing the status code "404" and the requested page.
5. To find the time with the most requests, we can use *grep* to extract all lines containing the timestamp, then use *awk* to extract the hour or the minute from the timestamp, then use *sort* to sort the extracted values and finally use *uniq* with the *-c* option to count the number of occurrences of each hour or minute. The hour or minute with the highest count will be the one that had the most requests.
6. This script uses a while loop to read each line of the log file, and it uses the *grep* command to extract the IP and user agent from each line. The script then uses the *awk* command to extract the IP and user agent from the line, and it uses an if statement to check if the user agent contains the word "bot", "crawler", or "spider". If the user agent contains one of these words, the script will print the IP and user agent of that search bot.

```
ubuntu18@server-1:~/homework$ ./script2.sh
IP address with the most requests:
  29 94.78.95.154
Most requested page:
  190 /wp-content/uploads/2014/11/favicon.ico
Number of requests from each IP:
  29 94.78.95.154
  21 95.31.14.165
  19 176.108.5.105
  16 31.7.230.210
  14 144.76.76.115
  14 13.0.782.112
  12 217.69.133.239
  11 66.102.9.35
  11 5.255.251.28
  11 217.69.133.234
  11 188.123.232.29
  11 17.3.1.840
  10 91.121.209.185
  10 46.158.68.55
   9 93.170.253.156
   9 5.135.154.105
   9 217.69.133.236
   8 91.206.110.87
   8 82.193.140.164
   8 66.102.9.32
   8 217.69.133.235
   8 213.80.162.114
```

```
   8 195.24.255.94
   7 66.102.9.38
   7 31.173.84.130
   6 95.65.45.111
   6 66.249.66.204
   6 46.188.127.46
   6 217.69.134.12
   6 217.69.133.70
   6 217.69.133.240
   6 197.231.221.211
   6 195.211.65.102
   6 193.169.81.189
   5 85.26.183.33
   5 82.145.222.114
   5 80.93.117.22
   5 66.249.66.199
   5 46.161.15.91
   5 33.0.0.0
   5 217.69.134.29
   5 217.69.133.238
   5 194.190.172.103
```

5 178.216.120.195
5 176.59.52.28
4 85.140.79.235
4 85.140.3.12
4 46.211.97.106
4 46.173.188.61
4 217.116.232.205
4 213.87.147.151
4 212.112.119.234
4 207.46.13.3
4 193.201.224.8
4 193.109.165.82
4 17.3.1.369
3 95.59.16.51
3 93.90.82.222
3 92.42.128.111
3 92.115.237.194
3 85.26.253.211
3 85.175.246.117
3 82.207.56.26
3 82.113.123.4
3 81.5.66.178
3 80.255.89.222
3 80.234.24.195
3 66.249.92.197
3 62.192.231.56
3 5.197.108.102

3 5.196.1.129
3 46.211.64.4
3 46.211.154.121
3 46.162.1.107
3 46.119.8.161
3 40.77.167.19
3 37.76.144.10
3 37.114.132.13
3 31.52.247.81
3 31.41.10.55
3 31.193.90.131
3 31.130.249.87
3 2.61.172.119
3 217.69.133.237
3 217.66.152.15
3 213.221.43.38
3 213.186.8.43
3 213.134.214.170
3 212.74.202.86
3 212.115.253.100
3 194.50.9.105
3 193.200.32.117
3 193.124.179.142
3 188.18.226.179
3 185.168.187.11
3 178.90.59.26
3 178.34.55.47
3 178.214.194.138

```
3 176.97.103.24
3 17.4.1.576
3 150.129.64.157
3 146.185.223.129
3 136.243.95.186
3 136.243.95.154
3 117.213.10.129
3 112.209.25.228
3 110.36.209.82
2 95.91.244.229
2 95.47.14.206
2 95.215.48.58
2 95.213.218.57
2 95.134.32.242
2 95.108.141.46
2 95.105.67.60
2 94.43.218.97
2 94.253.16.43
2 94.23.230.134
2 94.19.255.114
2 93.81.227.242
2 93.185.218.39
2 93.177.36.83
2 93.171.225.250
2 92.47.195.10
2 91.55.0.215
2 91.234.219.102
2 91.233.111.96
```

A long list with the number of requests from other ips...

```
1 10.5.2.582
1 10.10.0.0
1 0.0.0.0
Non-existent pages:
10 /wp-content/themes/cassia/css/fonts/flexslider-icon.eot?
2 /ukhod-za-soboj/pokhudenie/marinovannyjj-imbir-dlya-pokhudeniya.html
2 /.svn/wc.db
2 /.svn/format
2 /LISU.woff
2 /LISU.ttf
2 /.hg/hgrc
2 /.git/config
2 /.bzl/branch-format
2 /.bash_history
2 /
1 /wp-content/uploads/2014/11/favicon.ico
1 /wp-content/themes/cassia/css/fonts/flexslider-icon.woff
1 /wp-content/themes/cassia/css/fonts/flexslider-icon.ttf
1 /wp-content/themes/cassia/css/fonts/flexslider-icon.svg
1 /wp-content/plugins/stats-counter/img/icon.png
1 /ukhod-za-soboj/pokhudenie/dieti/map.css
1 /ukhod-za-soboj/pokhudenie/dieti/dieta-grechnevaya-s-kefirom.html%D1%87
%D0%B8%D1%82%D0%B0%D0%B9
1 /ukhod-za-soboj/molodost/omolozhenie-lica-posle-50-let.html
1 /ukhod-z
1 /sxd/info.php
1 /rtc_wizard_index.htm
1 /register.aspx
```

```

1 /otzivi-obzori/yuviderm-otzivi-obzori/yuviderm-ne-ochen-dovolna-22-goda
.html
1 /google89150ef520da50eb.html
1 /ehsteticheskaya-medicina/register.aspx
1 /ehsteticheskaya-medicina/injekcii/register.aspx
1 /ehsteticheskaya-medicina/injekcii/biorevitalizaciya/register.aspx
1 /ehsteticheskaya-medicina/injekcii/biorevitalizaciya/preparaty-dlya-bio
revitalizacii.html/register.aspx
1 /ehsteticheskaya-medicina/injekcii/biorevitalizaciya/preparaty-dlya-bio
revitalizacii.html
1 /backup/info.php
1 /apple-touch-icon-precomposed.png
1 /apple-touch-icon.png
Time with the most requests:
253 11
Search bots that accessed the site (UA + IP):
IP: http://example.com/ehsteticheskaya-medicina/injekcii/otlichie-botoksa-ot-ko
zhnykh-napolnitel.html", UA:
IP: http://example.com/ehsteticheskaya-medicina/injekcii/otlichie-botoksa-ot-ko
zhnykh-napolnitel.html", UA:
IP: http://example.com/ehsteticheskaya-medicina/injekcii/botoks-v-domashnikh-us
loviyakh.html", UA:

```

C. Create a data backup script that takes the following data as parameters:

1. Path to the syncing directory.
2. The path to the directory where the copies of the files will be stored.

In case of adding new or deleting old files, the script must add a corresponding entry to the log file indicating the time, type of operation and file name. [The command to run the script must be added to crontab with a run frequency of one minute

Here is our script:

```
#!/bin/bash
```

```
# Get the current date and time
```

```
current_time=$(date +%Y-%m-%d_%H:%M:%S)
```

```
# Syncing directory path
```

```
sync_dir=$1
```

```
# Backup directory path
```

```
backup_dir=$2
```

```
# Log file path
```

```
log_file=backup.log
```



```
# Copy all files from the syncing directory to the backup directory
```

```
rsync -av $sync_dir $backup_dir --ignore-existing
```

```
# Find any new or deleted files in the syncing directory
```

```
new_files=$(find $sync_dir -type f -cnewer $backup_dir)
```

```
for file in $new_files; do
```

```
    echo "$current_time: Added $file" >> $log_file
```

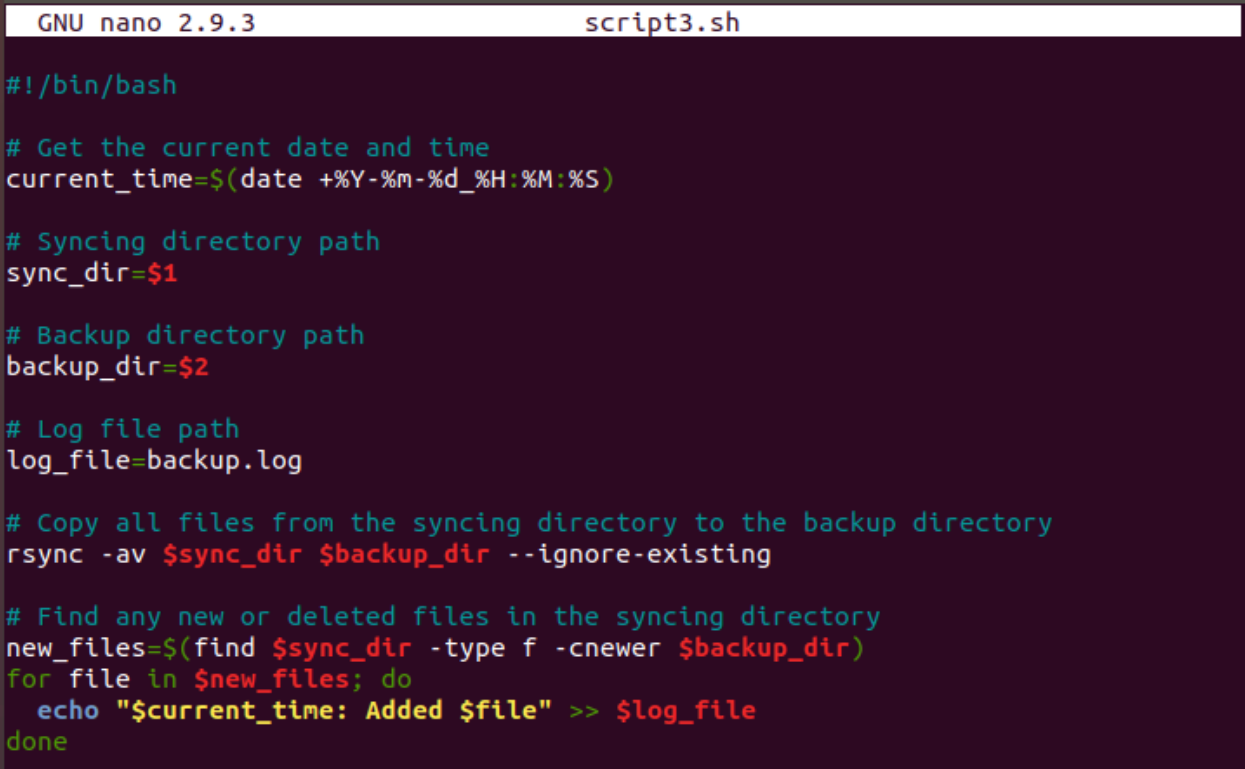
```
done
```

```
deleted_files=$(find $backup_dir -type f ! -newer $sync_dir)
```

```
for file in $deleted_files; do
```

```
    echo "$current_time: Deleted $file" >> $log_file
```

```
done
```



```
GNU nano 2.9.3                                script3.sh

#!/bin/bash

# Get the current date and time
current_time=$(date +%Y-%m-%d_%H:%M:%S)

# Syncing directory path
sync_dir=$1

# Backup directory path
backup_dir=$2

# Log file path
log_file=backup.log

# Copy all files from the syncing directory to the backup directory
rsync -av $sync_dir $backup_dir --ignore-existing

# Find any new or deleted files in the syncing directory
new_files=$(find $sync_dir -type f -cnewer $backup_dir)
for file in $new_files; do
    echo "$current_time: Added $file" >> $log_file
done
```

Let's open the crontab file and add the task:

```
ubuntu18@server-1:~/homework$ crontab -e
no crontab for ubuntu18 - using an empty one

Select an editor. To change later, run 'select-editor'.
 1. /bin/nano      <---- easiest
 2. /usr/bin/mcedit
 3. /usr/bin/vim.tiny
 4. /bin/ed

Choose 1-4 [1]: 1
crontab: installing new crontab
```

```
GNU nano 2.9.3 /tmp/crontab.YI7QSl/crontab Modified

# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command
* * * * * /home/ubuntu18/homework/script3.sh /home/ubuntu18/homework/syncing_d$
```

```
GNU nano 2.9.3 /tmp/crontab.YI7QSl/crontab Modified

# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
#
# m h dom mon dow   command

$syncing_directory /home/ubuntu18/homework/backup_directory
```

Let's test the execution of our script:

```
ubuntu18@server-1:~/homework$ ls
apache_logs.txt  example_log.log  script2.sh  syncing_directory
backup_directory  script1.sh      script3.sh
ubuntu18@server-1:~/homework$ cp example_log.log /home/ubuntu18/homework/syncin
g_directory
ubuntu18@server-1:~/homework$ cp script1.sh /home/ubuntu18/homework/syncing_dir
ectory
ubuntu18@server-1:~/homework$ cp script2.sh /home/ubuntu18/homework/syncing_dir
ectory
ubuntu18@server-1:~/homework$ rm /home/ubuntu18/homework/syncing_directory/scri
pt1.sh
ubuntu18@server-1:~/homework$ ls /home/ubuntu18/homework/backup_directory
syncing_directory
ubuntu18@server-1:~/homework$ ls /home/ubuntu18/homework/syncing_directory
apache_logs.txt  example_log.log  script2.sh
ubuntu18@server-1:~/homework$ ls /home/ubuntu18/homework/backup_directory/synci
ng_directory
apache_logs.txt  example_log.log  script1.sh  script2.sh
```

Let's check the log file:

```
GNU nano 2.9.3 /home/ubuntu18/backup.log
2023-01-23_07:16:01: Deleted /home/ubuntu18/homework/backup_directory/syncing_$
2023-01-23_07:17:01: Deleted /home/ubuntu18/homework/backup_directory/syncing_$
2023-01-23_07:18:01: Deleted /home/ubuntu18/homework/backup_directory/syncing_$
2023-01-23_07:19:01: Deleted /home/ubuntu18/homework/backup_directory/syncing_$
2023-01-23_07:20:01: Deleted /home/ubuntu18/homework/backup_directory/syncing_$
2023-01-23_07:21:01: Deleted /home/ubuntu18/homework/backup_directory/syncing_$
2023-01-23_07:22:01: Deleted /home/ubuntu18/homework/backup_directory/syncing_$
2023-01-23_07:23:01: Added /home/ubuntu18/homework/syncing_directory/example_l$
2023-01-23_07:23:01: Deleted /home/ubuntu18/homework/backup_directory/syncing_$
2023-01-23_07:23:01: Deleted /home/ubuntu18/homework/backup_directory/syncing_$
2023-01-23_07:24:01: Added /home/ubuntu18/homework/syncing_directory/example_l$
2023-01-23_07:24:01: Deleted /home/ubuntu18/homework/backup_directory/syncing_$
2023-01-23_07:24:01: Deleted /home/ubuntu18/homework/backup_directory/syncing_$
2023-01-23_07:25:01: Added /home/ubuntu18/homework/syncing_directory/script1.sh
2023-01-23_07:25:01: Added /home/ubuntu18/homework/syncing_directory/example_l$
2023-01-23_07:25:01: Deleted /home/ubuntu18/homework/backup_directory/syncing_$
2023-01-23_07:25:01: Deleted /home/ubuntu18/homework/backup_directory/syncing_$
2023-01-23_07:26:01: Added /home/ubuntu18/homework/syncing_directory/script1.sh
2023-01-23_07:26:01: Added /home/ubuntu18/homework/syncing_directory/example_l$
2023-01-23_07:26:01: Added /home/ubuntu18/homework/syncing_directory/script2.sh
2023-01-23_07:26:01: Deleted /home/ubuntu18/homework/backup_directory/syncing_$
2023-01-23_07:26:01: Deleted /home/ubuntu18/homework/backup_directory/syncing_$
2023-01-23_07:26:01: Deleted /home/ubuntu18/homework/backup_directory/syncing_$
[ Read 127 lines ]
```