Bash Practice Questions for Chapter 10:

These simple exercises will help you practice what you learned in the tenth chapter of the Bash Beginner Series on Linux Handbook.

All the problems in this exercise are advanced level and require you to have access to a test server be it in a VM, Cloud or Phsyical Server. If you don't have any, don't worry and don't feel too bad about it. Just read the problems and see the suggested solution to improve your understanding.

Exercise 1: Automate user management on servers

Creating a user on multiple servers maybe something that you would do on a daily basis as a sysadmin. It is a tedious a task. Write a bash script that automates the entire process.

Difficulty level: Advanced

Hints: First, create a text file that includes all the server hostnames or IP addresses where you wish to add the user on.

For example, you can create the file **servers.txt** that includes the names of five different servers:

```
kabary@handbook:~$ cat servers.txt
server1
server2
server3
server4
server5
```

These are server hostnames and the IP addresses are already assumed to have been included in the **/etc/hosts** file so that one can SSH into any of those servers. You can also use SSH config file. The script would first ask you to enter the username and the user id of the user who you want to add; then, it will loop over and connect to all the servers in the **servers.txt** file via SSH and add the requested user.

Exercise 2: Automate backups to a separate server

Write an automated script to backup archives of your files on a separate server.

Difficulty level: Advanced

Hints: Create an array named **backup_dirs** that stores all directory names that we want to backup. Next, create three other variables:

- **dest_dir:** To specify the backup destination directory.
- **dest_server:** To specify the backup destination server.
- **backup_time:** To specify the date of the backup.

Finally, for all the directories in the **backup_dirs** array, create a gzip compressed tar archive in **/tmp**, then use the scp command to send/copy the backup to the destination server. Finally, remove all the gzip archives from **/tmp**.

Exercise 3: Automate backups on a regular basis

Take backups based on exercise 2 every day at midnight on the separate backup server.

Difficulty level: Intermediate

Hint: Schedule the script described in exercise 2 to run as a cron job.

Exercise 4: Automate monitoring available disk space

Use the commands `df`, `tail`, `awk` and `cut` to grab the disk usage percentage of your system and check to see if it's bigger than or equal to 90%.

Difficulty level: Advanced

Hint: First, created a **filesystems** array that holds all the filesystems that you want to monitor. Then, for each filesystem, you grab the usage percentage with the given commands and check to see if it's bigger than or equal to 90%. If usage is above 90%, it should send an alert email indicating that the filesystem is running out of space.

Exercise 5: Automate monitoring disk space on a regular basis

Check disk usage based on exercise 4 every six hours on your system.

Difficulty level: Intermediate

Hint: Schedule the script described in exercise 4 to run as a cron job.

Exercise 6: Automate both disk space monitoring and taking backups on a regular basis

Use Cron to take backups based on exercise 2 every day at midnight on the separate backup server. Also check disk usage based on exercise 4 every six hours on your system.

Difficulty level: Intermediate

Hint: Schedule the script described in exercise 2 and 4 to run as a cron job.

Solutions to the Exercises

Solution 1: Automate user management on servers

> Creating a user on multiple servers maybe something that you would do on a daily basis as a sysadmin. It is a tedious a task. Write a bash script that automates the entire process.

```
#!/bin/bash

servers=$(cat servers.txt)

echo -n "Enter the username: "
read name
echo -n "Enter the user id: "
read uid

for i in $servers; do
echo $i
ssh $i "sudo useradd -m -u $uid ansible"
if [ $? -eq 0 ]; then
echo "User $name added on $i"
else
echo "Error on $i"
fi
done
```

Solution 2: Automate backups to a separate server

> Write an automated script to backup archives of your files on a separate server.

```
#!/bin/bash

backup_dirs=("/etc" "/home" "/boot")
dest_dir="/backup"
dest_server="server1"
backup_date=$(date +%b-%d-%y)

echo "Starting backup of: ${backup_dirs[@]}"

for i in "${backup_dirs[@]}"; do
sudo tar -Pczf /tmp/$i-$backup_date.tar.gz $i
if [ $? -eq 0 ]; then
echo "$i backup succeeded."
else
echo "$i backup failed."
fi
scp /tmp/$i-$backup_date.tar.gz $dest_server:$dest_dir
if [ $? -eq 0 ]; then
echo "$i transfer succeeded."
else
echo "$i transfer failed."
fi
done

sudo rm /tmp/*.gzecho "Backup is done."
```

Solution 3: Automate backups on a regular basis

> Take backups based on exercise 2 every day at midnight on the separate backup server.

```
kabary@handbook:~$ crontab -e
0   0   *   *   * /home/kabary/scripts/backup.sh
```

Solution 4: Automate monitoring available disk space

> Use the commands `df`, `tail`, `awk` and `cut` to grab the disk usage percentage of your system and check to see if it's bigger than or equal to 90%.

```
#!/bin/bash
filesystems=("/" "/apps" "/database")
for i in ${filesystems[@]}; do
usage=$(df -h $i | tail -n 1 | awk '{print $5}' | cut -d % -f1)
if [ $usage -ge 90 ]; then
alert="Running out of space on $i, Usage is: $usage%"
echo "Sending out a disk space alert email."
echo $alert | mail -s "$i is $usage% full" your_email
fi
done
```

Solution 5: Automate monitoring disk space on a regular basis

> Check disk usage based on exercise 4 every six hours on your system.

```
kabary@handbook:~$ crontab -e
0   */6 *       *       * /home/kabary/scripts/disk_space.sh
```

Solution 6: Automate both disk space monitoring and taking backups on a regular basis

> Use Cron to take backups based on exercise 2 every day at midnight on the separate backup server. Also check disk usage based on exercise 4 every six hours on your system.

```
kabary@handbook:~$ crontab -e
0   0   *   *   * /home/kabary/scripts/backup.sh
0   */6 *       *       * /home/kabary/scripts/disk_space.sh
```

Please note here, that we are using the same crontab file to execute both scripts on a regular basis.