QUESTION: 1

cron, anacron and system timer units all run the scheduled tasks in linux and unix systems.

cron

```
userone@linux$ crontab -l
@yearly /home/userone/annual-maintenance
*/10 * * * * /home/userone/check-disk-space
```

I logged in to my linux server as userone and gave the **crontab –l** command. That gave the above output. From this output we can come to know the crontab format. It is like

Field	Description	Allowed Value
MIN	Minute field	0 to 59
HOUR	Hour field	0 to 23
DOM	Day of Month	31-Jan
MON	Month field	12-Jan
DOW	Day Of Week	0-6
CMD	Command	Any command to be executed

To edit my crontab entries I ran **crontab –e** command. Here crontab = command to be executed. e=edit option

We can schedule a job for more than one instance. We can schedule a job to run at a particular time. We can use @daily , @monthly , @yearly , @reboot keywords.

Keyword	Equivalent
@yearly	0 0 1 1 *
@daily	0 0 * * *
@hourly	0 * * * *
@reboot	Run at startup.

By default crontab sends the job output to the user who sets the cron job. We can redirect the output to some other user using MAIL variable in the crontab.

In cron the minimum time unit is minute. We cannot run cron jobs in seconds. We can add the cron jobs by running the commands in a separate file **(example cronfile.txt)** and adding the file entries to crontab by running the command **crontab cronfile.txt**.

Crontab is ideal for servers.

Anacron

It is the cron for laptops and desktops

When I ran cat **/etc/anacrontab** I got the following output.

7 15 test.daily /bin/sh /home/userone/backup.sh

Field 1 is Repeating period: This is a numeric value that specifies the number of days.
1 – daily
7 – weekly
30 – monthly
N – This can be any numeric value. N indicates number of days

Field 2 is Delay: This indicates the delay in minutes. i.e Waiting time. X number of minutes anacron should wait before executing the job after the the machine starts.

Field 3 is Job identifier: It is the name for the job's timestamp file. It should be unique for each job. This will be available as a file under the /var/spool/anacron directory. This file will contain a single line that indicates the last time when this job was executed.

```
# ls -1 /var/spool/anacron/
test.daily
cron.daily
cron.monthly
cron.weekly

# cat /var/spool/anacron/test.daily
20150504
```

Field 4 is command: Command or shell script that needs to be executed

Now **7 15 test.daily /bin/sh /home/userone/backup.**sh job runs once in 7 days , with a 15 min delay after the reboot. In this test.daily is the job identifier.

Comparison between cron and anacron

Cron is manly suitable for servers. i.e for the systems which will run continuously. Because during the time of execution if the system is down then the cron task will not get executed at all.

Anacron is suitable for desktops and laptops which will not run continuously. In this during the time of run if the system is down then the task will run when the systems starts next time.

Cron job can be used when we need to execute one jobs at particular time only. Anacron can be used for running the tasks irrespective of the time.

Cron job can be scheduled by any normal user. Usually anacron can be scheduled by only super users.

In cron the minimum granularity is minute. In anacron it is days.

Cron jobs can run any number of times like once in a minute. But anacron jobs can run only once in a day.

QUESTION: 2

To access the terminal of a remote system we need to use terminal command. But if we use terminal command the user name and passwords what we are sending over internet can be hacked by the hacker and can be misused.
To prevent this type of hacking SSH (Secure Shell) is used. In SSH login the usernames and passwords are travelling as encrypted text. Hacker can interfere and hack this user names and password file too. But since both this user name ad
password are encrypted the hacker cannot find out the user name and password.

Now let us see how the user name and passwords are encrypted.
There two ways of encrypting is there.

Symmetric Encryption.
A secret key is applied to our data that is going through net to some person. the receiver also will be having the secret key. The receiver cans decrypt the key using his private key. This is symmetric encryption.

Asymmetric encryption.
In this two keys will be there. One public key that will be available with everyone in the public. The sender and the receiver both will be having private keys. Sender will encrypt the message using a public key and send the encrypted data to the receiver.
The receiver will open the data using his private key.

When we are connecting one remote server using ssh the remote server will share its public key with our server. Remote servers tells that it knows about the private key what we are using.
Then communication will happen. We need to check whether we are connecting to the correct remote server. that we can check it using so many ways. Once we test that it is a genuine remote server we need to make an entry into our memory. For that purpose ~/.ssh/known_hosts. We need to make an entry into that file. Then when we try to login to the remote server second time the login will be quick and safe

Man-in-middle attack

Suppose ServerOne wishes to communicate with ServerTwo and want to do some data transfer.Meanwhile, HackerServerwishes to intercept the conversation to eavesdrop and possibly (although this step is unnecessary) deliver a false message to ServerTwo.

First, ServerOne asks ServerTwo for his public key. If ServerTwo sends his public key to ServerOne, but HackerServer is able to intercept it, a man-in-the-middle attack can begin. HackerServersends a forged message to ServerOne that claims to be from ServerTwo, but instead includes Mallory's public key.

ServerOne, believing this public key to be ServerTwo's, encrypts her message with Mallory's key and sends the enciphered message back to ServerTwo. HackerServeragain intercepts, deciphers the

message using her private key, possibly alters it if she wants, and re-enciphers it using the public key ServerTwo originally sent to ServerOne. When ServerTwo receives the newly enciphered message, he believes it came from ServerOne.

We know that the known hosts are in ~ssh/known_hosts file. If any hone hack our system then he will come to know all the known hosts of our system to which our host can make ssh connection easily. This will put the entire hosts in big risk. To avoid these hashed hosts files are used. In this it looks like

|1|dTVaYG/giqH3nvoLfGECyOsiMDs=|RzAc9qu1lG+3ZtajFbaVuL02SZA= ssh-rsa ...
|1|base64(salt)|base64(hash)|

So if the system is compromised the hacker cannot do anything now.

QUESTION: 3

I will write one script test.sh with the following contents

/root/test.sh

```
mount -t iso9660  /dev/dvdrom   /test

mv   /spare/home/tar.gz   /test

rm -rf   /spare/home*
```

I will run the following cron jobs for taking backups

```
crontab -e

00 20 * * 0 cd /spare;    tar --incremental -xzpf home_sunday.tar.gz /home

00 20 * * 1 cd /spare;    tar --incremental -xzpf home_monday.tar.gz /home

00 20 * * 2 cd /spare;    tar --incremental -xzpf home_tuesday.tar.gz /homee

00 20 * * 3 cd /spare;    tar --incremental -xzpf home_wednesday.tar.gz /home
```

00 20 * * 4 cd /spare; tar --incremental -xzpf home_thursday.tar.gz /home

00 20 * * 5 cd /spare; tar --incremental -xzpf home_full.tar.gz /home

00 21 * * 5 /root/test.sh

00 20 * * 4 cd /spare; tar --incremental -xzpf home_saturday.tar.gz /home

Meaning

crontab –e

Edit cron jobs

00 20 * * 0 cd /spare; tar --incremental -xzpf home_sunday.tar.gz /home

What it does ? Sunday night at 8:00 PM , Take an incremental backup of /home directory and save it in /spare.

00 20 * * 1 cd /spare; tar --incremental -xzpf home_monday.tar.gz /home

What it does ? Monday night at 8:00 PM , Take an incremental backup of /home directory and save it in /spare.

00 20 * * 2 cd /spare; tar --incremental -xzpf home_tuesday.tar.gz /home

What it does ? Tuesday night at 8:00 PM , Take an incremental backup of /home directory and save it in /spare.

00 20 * * 3 cd /spare; tar --incremental -xzpf home_wednesday.tar.gz /home

What it does ? Wednesday night at 8:00 PM , Take an incremental backup of /home directory and save it in /spare.

00 20 * * 4 cd /spare; tar --incremental -xzpf home_thursday.tar.gz /home

What it does ? Thursday night at 8:00 PM , Take an incremental backup of /home directory and save it in /spare.

00 20 * * 5 cd /spare; tar --incremental -xzpf home_full.tar.gz /home

What it does ? Friday night at 8:00 PM , Take an incremental backup of /home directory and save it in /spare.

00 21 * * 5 /root/test.sh

What it does ? Friday night at 9:00 PM run this test.sh

/root/test.sh

What it does ? Test.sh script is stored in /root directory

mount -t iso9660 /dev/dvdrom /test

What it does ? Mount the DVDRAM as a iso9660 file system drive and its mount point is /test

mv /spare/home/tar.gz /test

What it does ? Move tar.gz file from home directory from /spare hard disk to DVDROM

rm -rf /spare/home*

What it does ? Remove all the contents of /spare

00 20 * * 6 cd /spare; tar --incremental -xzpf home_saturday.tar.gz /home

What it does ? Saturday night at 8:00 PM , Take an incremental backup of /home directory and save it in /spare.