Lab 5

Backups

5.1 Objectives

Learn how to design and implement backups using basic UNIX tools.

5.2 Before you start

For this session you should be able to answer the following questions:

- How you can pack and unpack one or more files using the `tar` command?
- What is a hard link?
- What's the difference between copying two files (`cp file_a file_b`) and creating a hard link (`ln file_a file_b`)?

5.3 Introduction

One of the most important tasks of a system administrator is performing backups that can restore the complete system in an acceptable amount of time when a system failure that entails data loss occurs. These losses may be due to multiple factors such as hardware failures, software malfunction, human action (accidental or premeditated) or natural disasters [1].

Before making backups the system administrator should decide a policy taking into account aspects such as [2]:

- Select the correct type of physical media for backups taking into account the size, cost, speed, availability, usability and reliability.
- Deciding which files need a backup and where are those files. The most important files are configuration files and user files (usually located in */root* and */home*, respectively). Some files that do not require backups are the temporary files (*/tmp* and the system binaries (*/bin, /sbin/*).
- Decide the frequency and scheduling of backups. This depends on the variability of the data. A database may require multiple daily backups, while a web server can require only one daily copy, and other file systems may require only one weekly copy
- Analyze other aspects such as: where to store the copies to, how long the copies should be maintained, and how quickly you need to retrieve each file type.

Using the information above it is possible to decide a backup strategy. This includes deciding the frequency and type of copies. A common strategy is to make full and incremental copies. In this way, on one hand it is possible to reduce the time and size of data transfers, but, on the other hand, it also increases the complexity of the data restoration process.

A typical strategy is to make full copies weekly (also known as level 0) and incremental copies (known as level 1 or greater) daily. If the ratio of file change is very large we can modify the previous weekly

model for a model where each month there is more a copy of level 0, each week a copy of level 1 (incremental weekly) and every day a copy of level 2 (incremental daily).

Finally, you must decide the most suitable tools to implement the backup strategy that has been designed. In this laboratory we will use `tar` and `rsync`. Also, we will use the same disk as the physical environment to make backups. Take into account that in a real environment this is inconvenient because of the high risk that a data loss also affect the backups.

5.4 Partition to store the backups

To save the backup files that we will generate during the laboratory, create a new directory */backup*. Also create a new partition on the free space that you have and, finally, create a `btrfs` filesystem on it. Mount this partition in the */backup* directory so that only root has access permissions. Other users should not even be permitted to read the directory, given that the contents of the backups could be confidential.

❓ *What commands you have used to create the partition, format the filesystem, mount the partition and change the permissions of the /backup directory?*

To add more protections in this directory, it should be mounted on write mode only when writing the backups and the rest of the time in read-only mode. Normally, it should be necessary to unmount the partition before changing the write mode but you can change the mount options of a partition without unmounting it using the `remount` option.

```
Mount in read-only mode
# mount -o remount,ro /dev/usbX /backup

Mount in read-write mode
#mount -o remount,rw /dev/usbX /backup
```

❓ *What do you need to do in order to mount this new partition automatically in read-only mode at boot-time?*

5.5 Making backups using `tar`

5.5.1 full backups

Make a complete copy of the */root* directory (check that there are files in this directory) using the *tar* command. Use meaningful names for backup files: include information about the content, date and time that the backup, and if the backup is full or incremental, etc.

❓ *How you can include automatically the date in the backup file name? for example, backup-etc-level0-200912041030.tar*

> ☞ **Note:** Use the *date* command. As it will help you with the $() here command to paste the date as part of the filename.

? *What command you have used for making the full copy of the directory* `/root`*?*

? *If we want to compress the backup file which option you have to add to the* `tar` *command?*

? *Why is not generally good in terms of security to compress the backup file?*

Sometimes when you do the full copy it is necessary to exclude certain files. In order to do so you can construct a file with a list of files that should be excluded from the backup. Create again the full copy, but this time exclude from the copy the files listed in the *excludes* file. (Create this file and put some file names in there).

? *What command you have added to the* `tar` *command to exclude files?*

In addition to protecting the backup directory is important to have a mechanism that allows to verify that the backup files have not been modified after their creation. For this task it is common to use a digital signature mechanism, such as SHA512, which allow to verify the integrity of a file.

Once you've made the copy, use the `sha512sum` command (use `man sha512sum` to learn how to use this command) to generate a SHA512 hash and store the output in a file. Put a *.asc* as a file extension.

? *How you have used the* `sha512sum` *command to produce the sha512 signature?*

5.5.2 Incremental backups

To perform incremental copies modify some files in the */root* directory:

- Create new files and subdirectories

- Modify the contents of some files
- Use the `touch` command to change the modification date of some files.

For making incremental backups the `tar` command has the `--newer` option which creates a file that only includes the files that have been modified since a certain date. This date can be specified in two ways: first, placing it directly in the command line eg `--newer ="2022-01-28 12:10"`. The second way is to take the date from a file, in that case the date is taken from the last modification data of that file, for example: `--newer=./file`

Now make an incremental backup of the */root* directory with respect to the complete backup that yo made before using the `tar` command. Additionally create a `sha512sum` signature and save it to a different file

What command you used to make the incremental copy?

What potential problem has the use of the full backup file for obtaining the backup date when doing the incremental copy?

Note: take into account that the backup process can take a long time). How you can solve this problem?

Now, perform a second round of changes to the `/root` directory in order to perform a second incremental copy. Again:

- Create new files and subdirectories.
- Modify the contents of some files.
- Use the `touch` command to change the modification date of some files.
- Delete some of the files that you generated for the first copy incremental copy.

Make a second incremental copy of the */root* directory (with respect to the first incremental copy) using the `tar` command. Also create a `sha512sum` signature of the second incremental copy and put to it an appropriate name.

What command you have used to create the second incremental copy?

How can you verify that the content of the backup is the same as the original directory?

❓ *How can you verify, using the* `sha512sum` *command, the integrity of the backup?*

☞ **Note:** That means that the file has not been modified since it was created.

5.5.3 Restoring a backup

Rename the `/root` directory to `/root.old` to simulate the effect of a data loss. Now restore the directory from the backup (that means to restore the three files that were created: the complete copy, and both incremental).

❓ *In what order you should restore the files to the get the desired result?*

❓ *What commands you have used?*

❓ *What happened to the files that you had deleted before the second incremental copy?*

❓ *How you can detect that some files that have been deleted? When those file will be removed from the backups?*

Restoration of a fragment

Rename one of the subdirectories of the *root* directory to simulate that it was deleted. Then, restore only that particular subdirectory from the backup.

❓ *What commands you have used to retrieve only a portion of the backup?*

5.6 Making backups using `rsync`

Until now we have stored the backups in the same machine that contains the data, but what is most common is to have several machines to backup and store the copies on a central machine using the network. To do this we can use the `rsync` command. It allows to copy a directory (or set of files) to another directory via a network connection. `rsync` uses an efficient checksum algorithm to transmit only the differences between the two directories while, at the same time, compressing files for a faster transmission

This tool lets you copy files from or to a directory located in a remote machine, or directories from the same machine. What it does not allow is to copy directories between two remote machines. Moreover `rsync` allows copying links, devices, and preserve permissions, owners and groups. It also supports exclusion lists and remote connection using Secure Shell (SSH) among other possibilities (for more information see `man rsync`).

5.6.1 Making backups over a network

As mentioned earlier, `rsync` to back up a remote machine. This can be done with `rsh`, or by putting `rsync` in server mode, but it may be dangerous because a local machine in the network could be capturing the data of the connection. In order to solve these problems `rsync` allows secure connections using `ssh`.

What are the required steps for installing and activating the `ssh` server?

Important: Later we will need to perform ssh to the `root` account, for security reasons this is disabled by default. To enable `root` user through ssh you have to edit */etc/ssh/sshd_config*.

Search for the keyword `PermitRootLogin`, uncomment the line, i.e., remove the # at the beginning, and put the value to `yes` instead of `prohibit-password`.

5.6.2 Making full backups

Create a directory to make the `rsync` backups in the */backup* partition and then execute the following command: (Note: For the following command to work well it is necessary to activate the root account and put it a valid password)

```
# rsync -avz /root -e ssh root@localhost:/backup/rsync-backup/
```

What is the meaning of the options "avz" passed to `rsync`?

Now, create a file in the */root* directory, and try to do the same `rsync` command as before. Then, delete the file and execute again the `rsync` command.

What happened to the deleted file?

❓ *What option of `rsync` allows to an exact synchronization the two directories?*

How you can make a copy of all the files in the */root* directory except those that have a *.txt* extension?

❓ *What's the difference between making `rsync /source /destination` and `rsync /source /destination/`?*

5.6.3 Making reverse incremental backups

As seen in the previous section, every time you make a copy and synchronize, the directory where we you have the mirror is exactly like the source directory. This is a problem, in some situations, because it does not allow control over the changes made to the files. To solve this you can use use the `--backup` and `--backup-dir` options of `rsync`. The backups generated with these options are called inverse because the full copy is latest, as opposed to `tar`.

Here is a simple script to do reverse incremental backups with `rsync`. Fill it with the corresponding data.

```
 #!/bin/bash
SOURCE_DIR=
DEST_DIR=

# Excludes file: list of files to exclude
EXCLUDES=
# the name of the backup machine
BSERVER=
# put a date command for: year month day hour minute second
BACKUP_DATE=

# options for rsync
OPTS="--ignore-errors --delete-excluded --exclude-from=$EXCLUDES \
   --delete --backup --backup-dir=$DEST_DIR/$BACKUP_DATE -av"

# now the actual transfer
rsync $OPTS $SOURCE_DIR root@$BSERVER:$DEST_DIR/complet
```

Now create a file in the source directory and synchronize the backup with the script described above. Then, modify the new file and synchronize again. Finally delete the file and synchronize again.

❓ *What happens to the backup when the new file is modified?*

❓ *And, what happens when it is deleted?*

5.6.4 Snapshot-Style Backups

One possibility that gives `rsync` i s to make incremental backups where, using the properties of hard links, incremental copies appear like full copies [3].

First, we are going to analyze some properties of hard links.

Review of hard links

The file name does not represent the file itself, it is only are hard link to the `inode`. This allows a file (`inode` to have more than one hard link. For example if you have a file called *file_a* you can create a link to it called *file_b*:

```
# ln file_a file_b
```

Using the `stat` command it is possible to know how many hard links a file has:

```
# stat file_a
```

How you can detect if file_a and file_b belong to the same `inode`?

What happens to file_b if there are changes in file_a content?

What happens to file_b if there are changes in file_a permissions?

What happens to file_b if you copy another file, overwriting the file: `cp file_c file_a`?

And if it is overwritten with the `--remove-destination` option of `cp`?

And what happens to file_b if file_a is removed?

The `cp` command has an option (`-l`) to make a copy that is not a new file but a hard link. Another interesting option (`-a`) makes a copy recursively and preserving permissions of access, time and the owners of files.

Snapshot backups with the `cp` and `rsync`

You can combine `rsync` and `cp -al` to create multiple backups that seem full copies of a file system without requiring to spend the entire disk space required for all copies.

In summary it could be:

```
# rm -rf backup.3
# mv backup.2 backup.3
# mv backup.1 backup.2
# cp -al backup.0 backup.1
# rsync -a --delete source_directory / backup.0/
```

If the above commands are executed each day, the directories *backup.0*, *backup.1*, *backup.2* and *backup.3* appear as if they were full copies of the *source_directory* directory today, the day before today, two days before and three days before respectively. But, actually, the extra space will be equal to the size of the *source_directory* directory plus the total size of changes over the past three days. Exactly the same as a complete backup backups with three incremental that we have done before with the `tar` and `rsync` commands. The only problem is that the permissions and owner properties of copies of past days would be the same as the current copy.

There is an option that makes `rsync` to work with hard links directly `--link-dest` thus making unnecessary the `cp` command. Besides, it preserves the permissions and owners of previous copies.

With this option the previous commands will be like this:

```
# rm -rf backup.3
# mv backup.2 backup.3
# mv backup.1 backup.2
# mv backup.0 backup.1
# rsync -a --delete --link-dest=../backup.1 source_directory/ backup.0/
```

Script to make snapshot backups

Use the script `backup-script-snapshot.sh` which is available on the sftp server under the *sources*[1] directory to make backups of the */root* directory. First, modify the variables in the section ”File Locations” with the appropriate values to your system. After this, complete the section with the *rsync* command with the appropriate value for the snapshot backups.

How is the `rsync` command in the script to make the snapshot copies?

[1] The password may be found in Lab 1.

Now, make changes to files in the source directory (i.e. create a new file, modify the content and date of files, or delete some files) and execute again the script. Do this several times until you have a current copy and three previous copies.

? *What happens to the backup after the execution of the script?*

? *What is the size of the directory /backup.0 and other directories backup directories?*

Finally, we are going to do a restore. Rename the directory */root* to simulate a data loss. Perform a restore using this the most recent backup.

? *Where is the most recent data?*

? *Which command can be used to recover the data?*

Bibliography

[1] L. Wirzenius, J. Oja, S. Stafford, and A. Weeks, *The Linux System Administrators' Guide, version 0.9*. The Linux Documentation Project. TLDP. [Online]. Available: http://www.tldp.org/LDP/sag/sag.pdf

[2] E. Nemeth, G. Snyder, T. R. Hein, B. Whaley, and D. Mackin, *UNIX and Linux System Administration Handbook (5th Edition)*, 5th ed. Addison-Wesley Professional, 2017.

[3] Rubel, *Easy Automated Snapshot-Style Backups with Linux and Rsync*. [Online]. Available: http://www.mikerubel.org/computers/rsync_snapshots/