TAR BACKUP

Full vs Incremental vs Differential backups

Before we see how to use `tar` to create incremental and differential backups, it is important to understand the difference between these types of backup.

First of all, we have to say that both incremental and differential backups are ultimately based on **full** or "level 0" backups: a full backup is a backup which, in a form of another, contains all the content of a specific filesystem in a determinate point in time. Creating full backups potentially requires a lot of time and space on disk: this obviously depends on the size of the data source. As a solution to these downsides, incremental and differential backup strategies can be implemented. After an initial full backup is created, subsequent ones, incremental or differential, will include only filesystem changes. What is the difference between the two?
Incremental and differential backups are similar in the sense that, as we already said, they both are **ultimately** based on full backups. What does change between the two is what they consider as a base to compute filesystem differences. An incremental backup is always dependent and based on the backup which immediately precedes it, either full or incremental itself; a differential backup, instead, uses always the initial full backup as a base.

Example of an incremental backup strategy

Suppose we want to create weekly backups of an hypothetical source directory implementing an **incremental backup strategy**. As a first thing, on Monday, we would create a full backup of the source directory. The next day, Tuesday, we would create a new backup, which would contain only files and directories which were created or modified in the source directory since the full backup occurred. The new backup will also keep track of files which were deleted ever since; it is what is called a "level 1" backup.

On Wednesday we would create a third backup, which, in turn, will "keep track" of all the differences which occurred since the backup we performed on Tuesday. This backup will therefore be dependent on the previous one directly, and indirectly on the first backup. We would keep on repeating the pattern for the rest of the week.

If some disaster should happen on Thursday, for example, to restore the filesystem status we had on Wednesday, we would need to restore, in order, all the backups we made since Monday; loosing one backup makes impossible to restore the ones which come after it.

Example of a differential backup strategy

An initial, full backup, is also the very first thing we need to do if we decide to implement a **strategy based on differential backups**. The level 0 backup is created on Monday, and one containing only the differences between it and the current status of the source directory is made on Tuesday. Until this point there are no differences with the incremental backups strategy.

Things change from the next day on. On Wednesday, instead of creating a backup based on the one we made the previous day, we would create one which is again based on the initial, full backup we made on Monday. We perform the same action the subsequent week days.

As you can see, in a differential backup strategy, each backup depends solely on the initial full backup, therefore to restore the status the filesystem had on a certain day, we only need the initial full backup, and the backup made on that day.

Once we have a grasp of the differences between the two approaches, we can see how to perform incremental and differential backups with tar.

Creating incremental backups

To create incremental backups with tar all we have to do is to combine two options:
`--create` and
`--listed-incremental`.
The former is what we use to specify we want to create an archive, the latter, instead, takes the path of a **snapshot** file as argument: this file is used by tar to store metadata about the status of the source filesystem at the time the backup is made. By reading it, when subsequent backups are made, tar can determine what files have been changed, added or deleted, and store only those. Let's see a practical example.

Suppose we want to create incremental backups of the `~/Document` directory, and store it on an external block device mounted on `/mnt/data` (here we will assume our user has write permissions on that directory). In our example, the ~/Document directory initially contains only two files: `one.txt` and `two.txt`. Here is the command we would run to create the backup:

```
$ tar --verbose --create --file=/mnt/data/documents0.tar  --listed-incremental=/mnt/data/documents.snar  ~/Documents
```

Let's examine the options we used above. We invoked tar with the `--verbose` option to make its output more explicit, and `--create` to specify what we want to do is to create an archive; we than passed the path where the archive should be created as argument to the `--file` option. Finally, by using the `--listed-incremental` option we instructed tar to create a differential backup, and store filesystem metadata in the `/mnt/data/document.snar` file (notice that the `.snar` extension is arbitrary – is just what is used for convention). Since this is the first time we run the command, **a full backup is created**. Here is the output of the command above:

```
tar: /home/egdoc/Documents: Directory is new
tar: Removing leading `/' from member names
/home/egdoc/Documents/
/home/egdoc/Documents/one.txt
/home/egdoc/Documents/two.txt
```

The archive and snapshot file have been created inside `/mnt/data`:

```
$ ls -l /mnt/data
-rw-r--r--. 1 egdoc egdoc 10240 Apr 16 07:13 documents0.tar
-rw-r--r--. 1 egdoc egdoc 113 Apr 16 07:13 documents.snar
drwx------. 2 root root 16384 Apr 9 23:27 lost+found
```

Suppose we now append a line to the `one.txt` file in the `~/Documents` directory:

```
$ echo "this is a new line" >> ~/Documents/one.txt
```

Additionally, we create a third file:

```
$ touch ~/Documents/three.txt
```

We run tar again, only changing the name of the destination archive. A **level 1 backup** is created. It does include only the file we modified (one.txt) and the one we just created (three.txt):

```
$ tar  --create --verbose --file=/mnt/data/documents1.tar --listed-
incremental=/mnt/data/documents.snar  ~/Documents

tar: Removing leading `/' from member names
/home/egdoc/Documents/
/home/egdoc/Documents/one.txt
/home/egdoc/Documents/three.txt
```

Once we launch the command, the content of the `documents.snar` is overwritten with metadata about the current status of the source directory.

To keep performing incremental backups, all we need to do is keep following this pattern. All we need to change each time, of course, is the name of the destination archive. Each new archive will contain only changes in the source directory which occurred since the previous backup was made.

Creating differential backups

As we just saw, creating incremental backups with tar is pretty easy. Creating **differential** backups is just as easy: all we need to change is how we handle the snapshot file. As we already mention, the difference between differential and incremental backups is that the former are always based on full backups.

Since each time we run tar like we did in the previous example, the content of the snapshot file is overwritten with metadata information about the status of the filesystem at the time the command is issued, we need to create a copy of the snapshot file generated when the full backup was made and pass its path to `--listed-incremental`, so that the original one remains untouched.

The first time we run the command just as we did above, so that a full backup is created:

```
$ tar --verbose --create --file=/mnt/data/documents0.tar
--listed-incremental=/mnt/data/documents.snar ~/Documents
```

When it's time to create the first differential backup, we need to create a copy of the snapshot file, which otherwise would be overwritten:

```
$ cp /mnt/data/documents.snar /mnt/data/documents.snar-1
```

At this point we invoke tar again, but we reference the copy of the snapshot:

```
$ tar --verbose --create --file /mnt/data/documents0.tar
--listed-incremental=/mnt/data/documents.snar-1 ~/Documents
```

To create differential backups, this pattern needs to be repeated each time we want to add a new backup.

Restoring a backup

How to proceed when we want to restore a backup created with tar, depends on what backup strategy we implemented. In all cases, the first thing to do is to restore the full backup, which in this case is `/mnt/data/documents0.tar`. Here is the command we would run:

```
$ tar --directory=/ --extract --verbose --file=/mnt/data/documents0.tar --
listed-incremental=/dev/null
```

In this case we invoked tar with the `--directory` option, to make so that tar moves into the given directory before starting the extraction. We used `--extract` to perform the extraction and `--verbose` to run in verbose mode, than we specified the path of the archive to be extracted with `--file`. Again, we used the `--listed-incremental` option, this time passing `/dev/null` as its argument. Why we did so?

When the `--listed-incremental` option is used together with `--extract`, tar attempts to restore from the specified archive, **deleting all the files in the destination directory which doesn't exist in the archive**. On restoration, the content of the snapshot file doesn't need to be read, so it's common practice to pass `/dev/null` as argument to the option.

Here is the output the command would return in our case:

```
tar: Deleting ‘home/egdoc/Documents/three.txt’
home/egdoc/Documents/one.txt
home/egdoc/Documents/two.txt
```

In this case, as you can see, the `three.txt` file existing in the `/home/egdoc/Documents` directory was deleted as part of the extraction, since when the backup was created the file didn’t exist.

If we used incremental backups, at this point, to restore the situation that existed on a specific day, we need to restore, in order, all the backups that was created since the full backup was created until the one created on that specific day. If we used differential backups, instead, since each differential backup is computed against the initial full backup, all we need to do is to restore the backup we created in that specific day.

Here is the output the command would return in our case:

```
tar: Deleting ‘home/egdoc/Documents/three.txt’
home/egdoc/Documents/one.txt
home/egdoc/Documents/two.txt
```

In this case, as you can see, the `three.txt` file existing in the `/home/egdoc/Documents` directory was deleted as part of the extraction, since when the backup was created the file didn’t exist.

If we used incremental backups, at this point, to restore the situation that existed on a specific day, we need to restore, in order, all the backups that was created since the full backup was created until the one created on that specific day. If we used differential backups, instead, since each differential backup is computed against the initial full backup, all we need to do is to restore the backup we created in that specific day.

Listing the content of an incremental archive

If we just want to list the content of an incremental archive, we can run tar together with the the `--list` option and repeat `--verbose` two times, together with `--listed-incremental`. Here is an example. Suppose we want to examine the content of the first level 1 backup we performed after the full backup. Here is what we would run:

```
$ tar --list --verbose --verbose --listed-incremental=/dev/null
--file=/mnt/data/documents1.tar
```

In our case, the commands returns the following output:

```
drwxr-xr-x egdoc/egdoc 30 2022-04-16 23:40 home/egdoc/Documents/
Y one.txt
Y three.txt
N two.txt

-rw-r--r-- egdoc/egdoc 19 2022-04-16 23:40 home/egdoc/Documents/one.txt
-rw-r--r-- egdoc/egdoc 0 2022-04-16 23:40 home/egdoc/Documents/three.txt
```

The output displays **the list of the files which existed in the source directory when the archive was created**. If the name of the file is preceded by a `Y` it means that the file is actually included in the archive, if it is preceded by a `N`, instead it is not. Finally if the name of the file is preceded by a `D` it means that it is included in the archive, but it is actually a directory.

In this case the `one.txt`, `two.txt` and `three.txt` were in place when the archive was created, however only `one.txt` and `three.txt` are preceded by a `Y`, and actually included in the archive, because they were the only ones that changed since the previous backup was made (in the example we appended a line to the former and created the latter after the full backup).