

LAB 01 - LINUX BACKUP

Adrien Barth, Jeremy Zerbib

TASK 1: PREPARE THE BACKUP DISK

1.

Which disks and which partitions on these disks are visible?

`/dev/sda` and `/dev/sda1` are visible as `ll /dev/hd* /dev/sd*`.

As for the `hd*` part, `ll` could not ready anything as there is no disk in IDE plugged in.

```
ls: cannot access '/dev/hd*': No such file or directory
brw-rw---- 1 root disk 8, 0 Sep 25 22:02 /dev/sda
brw-rw---- 1 root disk 8, 1 Sep 25 22:02 /dev/sda1
```

Which partitions are mounted? Use the command `mount` without parameters to find out.

```
root@ubuntu:/home/adrien# mount | grep /dev/sd
/dev/sda1 on / type ext4 (rw,relatime,errors=remount-ro)
```

2.

Which new files appeared?

`dev/sdb/`

3.

```
root@ubuntu:/home/adrien# parted /dev/sdb
GNU Parted 3.2
Using /dev/sdb
Welcome to GNU Parted! Type 'help' to view a list of commands.
(parted) print
Error: /dev/sdb: unrecognised disk label
Model: VMware, VMware Virtual S (scsi)
Disk /dev/sdb: 21.5GB
Sector size (logical/physical): 512B/512B
Partition Table: unknown
Disk Flags:
(parted) mktable
New disk label type? msdos
(parted) print free
Model: VMware, VMware Virtual S (scsi)
Disk /dev/sdb: 21.5GB
```

Sector size (logical/physical): 512B/512B

Partition Table: msdos

Disk Flags:

Number	Start	End	Size	Type	File system	Flags
	32.3kB	21.5GB	21.5GB		Free Space	

(parted) mkpart

Partition type? primary/extended? primary

File system type? [ext2]? fat32

Start? 0

End? 10000

Warning: The resulting partition is not properly aligned for best performance.

Ignore/Cancel? ignore

(parted) mkpart

Partition type? primary/extended? primary

File system type? [ext2]? ext4

Start? 10001

End? 21000

(parted) q

Information: You may need to update /etc/fstab.

4.

```
root@ubuntu:~# mkfs.vfat /dev/sdb1
```

```
mkfs.fat 4.1 (2017-01-24)
```

```
root@ubuntu:~# mkfs.ext4 /dev/sdb2
```

```
mke2fs 1.44.1 (24-Mar-2018)
```

```
Creating filesystem with 2685184 4k blocks and 671744 inodes
```

```
Filesystem UUID: d9402135-6cbc-4cd3-984a-c2c6482c0094
```

```
Superblock backups stored on blocks:
```

```
32768, 98304, 163840, 229376, 294912, 819200, 884736, 1605632, 2654208
```

```
Allocating group tables: done
```

```
Writing inode tables: done
```

```
Creating journal (16384 blocks): done
```

```
Writing superblocks and filesystem accounting information: done
```

```
root@ubuntu:~# mkdir /mnt/backup1 /mnt/backup2
```

```
root@ubuntu:~# mount /dev/sdb1 /mnt/backup1
```

```
root@ubuntu:~# mount /dev/sdb2 /mnt/backup2
```

5.

```
root@ubuntu:/home/adrien# ll /dev/sd*
```

```
brw-rw---- 1 root disk 8, 0 Sep 25 06:37 /dev/sda
```

```
brw-rw---- 1 root disk 8, 1 Sep 25 06:37 /dev/sda1
```

```
brw-rw---- 1 root disk 8, 16 Sep 25 06:48 /dev/sdb
```

```
brw-rw---- 1 root disk 8, 17 Sep 25 06:48 /dev/sdb1
```

```
brw-rw---- 1 root disk 8, 18 Sep 25 06:48 /dev/sdb2
```

6.

```
/dev/sdb1      9.4G  8.0K  9.4G   1% /mnt/backup1
```

```
/dev/sdb2      11G   41M  9.5G   1% /mnt/backup2
```

TASK 2: PERFORM BACKUPS USING TAR AND ZIP

1.

```
cd /mnt/backup1/  
tar -czf backup.tar.gz /home
```

```
zip -r backup.zip /home
```

Do the files in the archive have a relative path so that you can restore them later to any place?

```
tar: Removing leading `/' from member names
```

We can see that during the making of the archive, `tar` removed the leading "/" to make the path relative.

Regarding the `zip` command, it seems that the relative path is stored.

2.

```
cd /mnt/backup1  
tar -tf backup.tar.gz  
unzip -l backup.zip
```

3.

```
cd /tmp  
tar -xf /mnt/backup1/backup.tar.gz  
unzip /mnt/backup1/backup.zip -d zip
```

4.

```
find /home/ -newermt "2016-09-23 10:42:33" | tar cz -T - -f  
incrementalBackup1.tar.gz
```

TASK 3: BACKUP OF FILE METADATA

Using `tar`

We first started to create a temporary directory and add a file in it using `touch`:

```
cd Desktop/  
mkdir test testRestore  
cd test  
touch test  
ls -al #Using this will show the owner of the file  
stat test
```

```
jeremy@jeremy: ~/Desktop/test
File Edit View Search Terminal Help
jeremy@jeremy:~/Desktop/test$ touch test
jeremy@jeremy:~/Desktop/test$ ls -al
total 8
drwxr-xr-x 2 jeremy jeremy 4096 Okt  2 15:08 .
drwxr-xr-x 4 jeremy jeremy 4096 Okt  2 15:08 ..
-rw-r--r-- 1 jeremy jeremy   0 Okt  2 15:08 test
jeremy@jeremy:~/Desktop/test$ stat test
  File: test
  Size: 0                Blocks: 0                IO Block: 4096   regular empty file
Device: 801h/2049d      Inode: 280607           Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/   jeremy)   Gid: ( 1000/   jeremy)
Access: 2019-10-02 15:08:33.022803238 +0200
Modify: 2019-10-02 15:08:33.022803238 +0200
Change: 2019-10-02 15:08:33.022803238 +0200
 Birth: -
jeremy@jeremy:~/Desktop/test$
```

Using the `stat` command, you can see all the metadata you need (last modified, etc.) . Then we created a new user in order to change the ownership of that file.

```
sudo adduser adrien
sudo chown adrien test
```

After the change of ownership, you want to backup and restore the file using the command `tar`

```
ls -al #Using this will show the new owner of the file
stat test
```

```
jeremy@jeremy: ~/Desktop/test
File Edit View Search Terminal Help
jeremy@jeremy:~/Desktop/test$ stat test
  File: test
  Size: 0                Blocks: 0                IO Block: 4096   regular empty file
Device: 801h/2049d      Inode: 280607           Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1002/   adrien)   Gid: ( 1000/   jeremy)
Access: 2019-10-02 15:08:33.022803238 +0200
Modify: 2019-10-02 15:08:33.022803238 +0200
Change: 2019-10-02 15:13:17.476959340 +0200
 Birth: -
jeremy@jeremy:~/Desktop/test$ ls -al
total 8
drwxr-xr-x 2 jeremy jeremy 4096 Okt  2 15:08 .
drwxr-xr-x 4 jeremy jeremy 4096 Okt  2 15:08 ..
-rw-r--r-- 1 adrien jeremy   0 Okt  2 15:08 test
jeremy@jeremy:~/Desktop/test$
```

```
cd ..
tar -czf /mnt/backup1/test.tar.gz test/
cd testRestore
tar -xf /mnt/backup1/test.tar.gz
stat test/test
```

```

jeremy@jeremy:~/Desktop/test$ cd ..
jeremy@jeremy:~/Desktop$ tar -czf /mnt/backup1/test.tar.gz test/
jeremy@jeremy:~/Desktop$ cd testRestore/
jeremy@jeremy:~/Desktop/testRestore$ tar -xf /mnt/backup1/test.tar.gz
jeremy@jeremy:~/Desktop/testRestore$ stat test/test
  File: test/test
  Size: 0                Blocks: 0          IO Block: 4096   regular empty file
Device: 801h/2049d      Inode: 23222       Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/   jeremy)   Gid: ( 1000/   jeremy)
Access: 2019-10-02 15:16:31.069707391 +0200
Modify: 2019-10-02 15:08:33.000000000 +0200
Change: 2019-10-02 15:16:31.069707391 +0200
 Birth: -
jeremy@jeremy:~/Desktop/testRestore$ ls -al test/test
-rw-r--r-- 1 jeremy jeremy 0 Okt  2 15:08 test/test
jeremy@jeremy:~/Desktop/testRestore$

```

Everything is kept the way it was prior to the backup except the user that is changed to the current owner of the directory `testRestore`.

Using `zip`

The same steps can be repeated for this part and you can see that every metadata is kept the way it was except for the owner that was changed to `backup1`'s owner.

Before executing the `zip` command, we ran the same commands that we did with `tar` and we checked that the `stat` and `ls -la` were the same values as before.

```

jeremy@jeremy: ~/Desktop/testRestore
File Edit View Search Terminal Help
updating: test/test (stored 0%)
jeremy@jeremy:~/Desktop$ cd testRestore/
jeremy@jeremy:~/Desktop/testRestore$ ls
test
jeremy@jeremy:~/Desktop/testRestore$ rm -rf test/
jeremy@jeremy:~/Desktop/testRestore$ unzip /mnt/backup
backup1/ backup2/
jeremy@jeremy:~/Desktop/testRestore$ unzip /mnt/backup1/
archives.zip test.zip
jeremy@jeremy:~/Desktop/testRestore$ unzip /mnt/backup1/test.zip -d .
Archive:  /mnt/backup1/test.zip
  creating: ./test/
  extracting: ./test/test
jeremy@jeremy:~/Desktop/testRestore$ ls
test
jeremy@jeremy:~/Desktop/testRestore$ ls -la test/test
-rw-r--r-- 1 jeremy jeremy 0 Okt  2 15:08 test/test
jeremy@jeremy:~/Desktop/testRestore$ stat test/test
  File: test/test
  Size: 0                Blocks: 0          IO Block: 4096   regular empty file
Device: 801h/2049d      Inode: 23220       Links: 1
Access: (0644/-rw-r--r--)  Uid: ( 1000/   jeremy)   Gid: ( 1000/   jeremy)
Access: 2019-10-02 15:25:42.000000000 +0200
Modify: 2019-10-02 15:08:33.000000000 +0200
Change: 2019-10-02 15:26:19.435743576 +0200
 Birth: -
jeremy@jeremy:~/Desktop/testRestore$

```

TASK 4: SYMBOLIC AND HARD LINKS

Using `tar`

We are going to start by creating a directory and into it two files : `file1` and `file2`. The directory is in `/tmp`

```
jeremy@jeremy:/tmp$ mkdir test
jeremy@jeremy:/tmp$ ls
test
jeremy@jeremy:/tmp$ cd test/
jeremy@jeremy:/tmp/test$ touch file1
jeremy@jeremy:/tmp/test$ touch file2
jeremy@jeremy:/tmp/test$
```

Then we are creating some links with the commands :

```
ln -s file1 SL #Symbolic link
ln file2 HL #Hard Link
```

```
jeremy@jeremy:/tmp/test$ ls -al
total 8
drwxr-xr-x 2 jeremy jeremy 4096 Okt  7 22:06 .
drwxrwxrwt 3 root   root   4096 Okt  7 22:06 ..
-rw-r--r-- 1 jeremy jeremy    0 Okt  7 22:06 file1
-rw-r--r-- 1 jeremy jeremy    0 Okt  7 22:06 file2
jeremy@jeremy:/tmp/test$ ln -s file1 SL
jeremy@jeremy:/tmp/test$ ls -al
total 8
drwxr-xr-x 2 jeremy jeremy 4096 Okt  7 22:07 .
drwxrwxrwt 3 root   root   4096 Okt  7 22:07 ..
-rw-r--r-- 1 jeremy jeremy    0 Okt  7 22:06 file1
-rw-r--r-- 1 jeremy jeremy    0 Okt  7 22:06 file2
lrwxrwxrwx 1 jeremy jeremy    5 Okt  7 22:07 SL -> file1
jeremy@jeremy:/tmp/test$ ln file2 HL
jeremy@jeremy:/tmp/test$
```

Then we archive the folder in the `backup1`, then create a `restore` folder and *unarchive* the archive into it. Then we check the links.

```
sudo tar -czf /mnt/backup1/testLinks.tar.gz test/
sudo ls -al /mnt/backup1/testLinks.tar.gz
mkdir restore
cd restore/
tar -xf /mnt/backup1/testLinks.tar.gz
ls
cd test
ls -al
```

```

jeremy@jeremy:/tmp$ sudo tar -czf /mnt/backup1/testLinks.tar.gz test/
jeremy@jeremy:/tmp$ sudo ls -al /mnt/backup1/testLinks.tar.gz
-rw-r--r-- 1 jeremy jeremy 202 Okt  7 22:10 /mnt/backup1/testLinks.tar.gz
jeremy@jeremy:/tmp$ mkdir restore
jeremy@jeremy:/tmp$ cd restore/
jeremy@jeremy:/tmp/restore$ tar -xf /mnt/backup1/testLinks.tar.gz
jeremy@jeremy:/tmp/restore$ ls
test
jeremy@jeremy:/tmp/restore$ cd test
jeremy@jeremy:/tmp/restore/test$ ls -al
total 8
drwxr-xr-x 2 jeremy jeremy 4096 Okt  7 22:08 .
drwxr-xr-x 3 jeremy jeremy 4096 Okt  7 22:13 ..
-rw-r--r-- 1 jeremy jeremy   0 Okt  7 22:06 file1
-rw-r--r-- 2 jeremy jeremy   0 Okt  7 22:06 file2
-rw-r--r-- 2 jeremy jeremy   0 Okt  7 22:06 HL
lrwxrwxrwx 1 jeremy jeremy   5 Okt  7 22:07 SL -> file1
jeremy@jeremy:/tmp/restore/test$

```

We can then conclude that the links are saved via the `tar` command.

Using `zip`

We used the folder used prior to this to zip and unzip. The commands are as follows :

```

zip -r /mnt/backup1/testLinks.zip test/
cd restore/
unzip /mnt/backup1/testLinks.zip -d .
cd test/
ls
ls -la

```

```

jeremy@jeremy:/tmp$ zip -r /mnt/backup1/testLinks.zip test/
updating: test/ (stored 0%)
updating: test/cd.zip (stored 0%)
updating: test/SL (stored 0%)
updating: test/HL (stored 0%)
updating: test/file2 (stored 0%)
updating: test/file1 (stored 0%)
jeremy@jeremy:/tmp$ cd restore/
jeremy@jeremy:/tmp/restore$ unzip /mnt/backup1/testLinks.zip -d .
Archive:  /mnt/backup1/testLinks.zip
  creating: ./test/
  extracting: ./test/cd.zip
  extracting: ./test/SL
  extracting: ./test/HL
  extracting: ./test/file2
  extracting: ./test/file1
jeremy@jeremy:/tmp/restore$ cd test/
jeremy@jeremy:/tmp/restore/test$ ls
cd.zip  file1  file2  HL  SL
jeremy@jeremy:/tmp/restore/test$ ls -la
total 12
drwxr-xr-x 2 jeremy jeremy 4096 Okt  7 22:31 .
drwxr-xr-x 3 jeremy jeremy 4096 Okt  7 22:35 ..
-rw-r--r-- 1 jeremy jeremy  156 Okt  7 22:31 cd.zip
-rw-r--r-- 1 jeremy jeremy   0 Okt  7 22:06 file1
-rw-r--r-- 1 jeremy jeremy   0 Okt  7 22:06 file2
-rw-r--r-- 1 jeremy jeremy   0 Okt  7 22:06 HL
-rw-r--r-- 1 jeremy jeremy   0 Okt  7 22:06 SL
jeremy@jeremy:/tmp/restore/test$

```

We can see that the links are not kept with `zip`.