Backups

Estimated completion time: 45 – 90 minutes.

I. Objective

The objective of this lab assignment is to introduce students to features of common backup utilities, and to perform a few basic experiments to determine the effects of verification, compression and encryption on backups. These procedures assume the student has access to an accompanying spreadsheet to aid in the analysis. This spreadsheet highlights the cells that require input.

Note that these experiments are unscientific because there are not enough test cases (such as a large variety of backup sizes) and a lack of variety in backup destinations (such as external hard disk vs. internal hard disk vs. tape). These tests also ignore the differences between a VM and a physical machine. Nevertheless, despite these limitations, it is hoped that good observations and hypotheses can be made.

II. Getting Started

Heads up: There is a small appendix with some minimal help for common Unix commands.

A. **Boot your Linux system or VM. If necessary, log in and then open a terminal window and cd to the labtainer/labtainer-student directory. The pre-packaged Labtainer VM will start with such a terminal open for you. Then start the lab:**

labtainer backups2

Note the terminal displays the paths to three files on your Linux host:

1) This lab manual

2) The lab report template

3) A spreadsheet to be filled in as part of this lab.

On most Linux systems, these are links that you can right click on and select "Open Link". **If you chose to edit the lab report on a different system, you are responsible for copying the completed report back to the displayed path on your Linux system before using "stoplab" to stop the lab for the last time.**

Use "sudo su" now to elevate your privileges.

III.Tar Introduction

One of the common Unix command-line utilities for creating backups is the `tar` command. `tar` is short for *tape archive*, which is an indication of its age (back when tape was the only backup option). The original approach to backups was to copy files to a backup tape in a way that could easily be read back off the medium if the data needed to be copied or restored somewhere else. But `tar` can be used to do much more than make a copy to tape, as you can tell from the nearly 150 options that can be given on the command-line. Only a few of its potential uses will be used in this assignment.

The first basic use of the `tar` command is the creation of an archive file, which is often referred to as a *tarball*. Tarballs are similar to zip files, which were popularized on Microsoft Windows. Zip files and tarballs are different in at least two ways: 1) they use different file formats; and 2) `zip` compresses the archive while `tar` does not compress by default.

In this exercise you will **not** be backing up the root file system. Instead you will work with a separate files system mounted on /lab_mnt that contains a copy of the /usr/bin directory.

1. Determine the amount of data to be backed up.

   As shown below, use the `du` command (which stands for *disk usage*) to determine how much data exists under the `/lab_mnt` hierarchy. The "-s" option stands for *summarize*, and the 'b' option stands for *return the size in bytes*.

   ```
   du -sb /lab_mnt
   ```

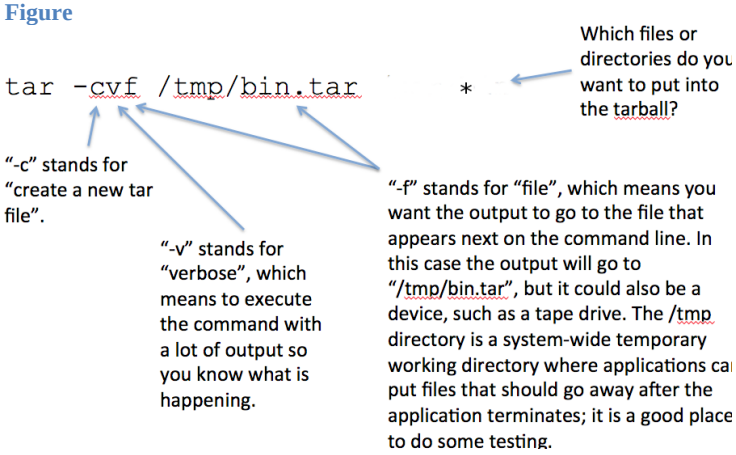
   Notation #1: ***How many bytes did the `du` command report? [Enter the number into the highlighted/shaded part of spreadsheet]***

2. Use `tar` to back up the data.

   Execute the following command to create a tarball from all the files and directories in the file system mounted on `/lab_mnt`. The command is explained in the next figure.

   ```
   cd /lab_mnt
   tar  -cvf  /tmp/lab_mnt.tar  *
   ```

Figure



3. List some of the metadata for the tarball you just created:

   ```
   ll /tmp/lab_mnt.tar
   ```

   Notation #2: ***How big is the tarball (in bytes)?*** The number of bytes is given just before the date. ***[Enter the number in the spreadsheet.]***

4. Look at the tarball content.

   To see that the tarball contains more than just the content of files, do the following to list the contents of the tarball:

   ```
   tar -tvf /tmp/lab_mnt.tar
   ```

   In the above command, the 't' option stands for *list*, 'v' still stands for *verbose*, and 'f' still stands for *file*. Taken together, the command means, "list the contents of the `/tmp/lab_mnt.tar` tarball in a verbose manner".

   The output shows that in addition to the file data, the tarball also has the metadata necessary to restore the file to a prior state.

5. Verify the tarball.

   A very, very important task after creating a backup is to verify that the backup is good, especially when using some kinds of "unreliable" media, such as tape or optical discs. The `tar` command comes with two options for verifying a tarball, which cause `tar` to read each file from the tarball and then compare it to the original file on the disk. To test one of these verification features, first do the following to cause a change to the metadata for an **existing** file:

   ```
   touch usr/bin/base64
   ```

Now run the `tar` command with a verification option, as shown below (where '-d' stands for *difference*):

```
tar -df /tmp/lab_mnt.tar
```

You can see that the command highlighted the change in m-time. Note that this *difference* option can be used at any time in the future to compare the contents in a tarball to the existing state of the files on disk.

6. Drop files from memory.

   Execute the following command to tell the kernel to drop any cached files in RAM so we can get repeatable, consistent times. [If we want to measure performance, then we want to make sure that every invocation of `tar` must do the same amount of work.]

   ```
   sync;  sysctl -w vm.drop_caches=2
   ```

7. Recreate the tarball.

   This time you will omit the verbose output and use the `time` command[1] to measure how long it takes to finish:

   ```
   time tar -cf /tmp/lab_mnt.tar *
   ```

   Notation #3: ***Referring to the* real *time[2] that was displayed as part of the output of the above command, how many seconds did it take to create the tarball? [Enter the number in the spreadsheet.]***

8. Clean up.

   Delete the tarball and clear the caches:

   ```
   rm /tmp/lab_mnt.tar

   sync;  sysctl -w vm.drop_caches=2
   ```

---

1 A previous lab showed one way of measuring the passage of time. This approach is a more accurate way of measuring time.

2 The **user** and **sys** times refer to the amount of CPU time used by the application and by the kernel, respectively. The **real** time refers to actual passage of time, which we will use in this exercise.

IV. Dump Introduction

Another common Unix backup utility is called `dump`. When backing up an entire disk partition, `dump` provides support for different *dump levels*, which allow the user to determine whether a full backup of all files is wanted, or just those files that have changed since some moment in the past, such as all files that have changed since the last full backup. When just a portion of a disk partition is selected, the only choice `dump` gives you is a full backup.

1. Use `dump` to back up the data.

   This time, use `dump` to back up the file system mounted on `/lab_mnt`, as shown below. The resulting file is called a *dump file*. [The "-0" below (the number *zero*) means *full backup*.]

   ```
   time dump -0 -f /tmp/lab_mnt.dump /lab_mnt
   ```

   Notation #4: ***How many seconds did it take to create the dump file? [Enter the number in the spreadsheet.]***

   Notation #6: ***How big is the dump file? [Enter the number in the spreadsheet.]***

2. Clear the caches:

   ```
   sync;  sysctl -w vm.drop_caches=2
   ```

3. Back up the data to a remote archive server. We will use ssh to send the results of the dump command to a remote server named "archive".

   Confirm you can reach the archive server:

   ```
   ssh student@archive
   ```

   When prompted by ssh to continue connecting, type "yes".

   Assuming you were able to reach the archive server, exit your ssh session:

   ```
   exit
   ```

   Pipe the dump command through ssh to the remote archive server:

   ```
   time (dump -0 -f - /lab_mnt | ssh student@archive "cat >
        lab_mnt.dump")
   ```

   Notation #5*:* ***How many seconds did it take to create the remote dump file? [Enter the number in the spreadsheet.]***

V. Dump and Verification

A very important step in a backup procedure is the verification that the backup was successful and good. This section will examine the impact that verification has on a backup. The `dump` command cannot be used to verify a dump file, but instead one must

use the complementary `restore` command with the "-C" option (which stands for *check*).

1. Delete a file that was backed up earlier:

   ```
   rm /lab_mnt/usr/bin/cheese
   ```

2. Verify the local backup using the following `restore` command:

   ```
   time restore -Cf /tmp/lab_mnt.dump -D /lab_mnt
   ```

   Notation #7: ***How many seconds did it take to verify the local dump file? [Enter the number in the spreadsheet.]***

   Note that the restore command showed you that it detected a missing file near the end of the output.

3. Verify the remote backup using the following `restore` command:

```
time (ssh student@archive "cat lab_mnt.dump" | restore -C -D /lab_mnt -f -)
```

   Notation #8: ***How many seconds did it take to verify the remote dump file? [Enter the number in the spreadsheet.]***

   Note that the restore command showed you that it detected a missing file near the end of the output.

4. Restore the file from the dump file after first changing to directory into the file system:

   ```
   cd /lab_mnt
   restore -i -f /tmp/lab_mnt.dump
   ```

   This will put you into an interactive session with the `restore` command, as you can see from the "`restore>`" prompt. At the prompt, enter the following command to add the file we need restored to a list of files to be restored. [If we wanted to restore several files we would continue to add them in the same way.]

   ```
   add usr/bin/cheese
   ```

   Now enter the command to copy from the dump file to its original location:

   ```
   extract
   ```

   When it asks for the volume number, enter **1**.
   When it asks if you want to set the owner/mode, enter **n**.

Enter **quit** to leave the `restore` shell.

Verify that the file has returned:

```
ll usr/bin/cheese
```

5. Clear the caches:

```
sync;  sysctl -w vm.drop_caches=2
```

VI. Dump and Compression

This section will examine the impact compression has on a backup. Note that a more scientific experiment would also consider the impact on different kinds of data being backed up. For example, if you are backing up a lot of files that are already compressed (such as video or music files), then time could be wasted on an effort that adds little benefit.

1. Do the following to measure how long it takes to create and compress the dump file (where the 'z' option indicates compression).

```
cd /lab_mnt
time dump -0 -z -f /tmp/lab_mnt.dump.z usr/bin
```

Notation #9*: How many seconds did it take to create and compress the dump file? [Enter the number in the spreadsheet.] Note that it should have taken longer to dump & compress than to just dump. If it took you less time, then you may have skipped over the step to clear the cache.*

Notation #10*: How big is the compressed dump file? [Enter the number in the spreadsheet.]*

2. Delete the dump file **and** clear the caches:

```
rm /tmp/lab_mnt.dump.z

sync;  sysctl -w vm.drop_caches=2
```

VII. Dump and Encryption

This section will examine the impact that encryption has on a backup procedure. There is no separate encryption option with the `dump` command, so you will need to use a separate encryption utility. For this test you will use the `openssl` command, which you may have used in a CS3600 lab.

1. Encrypt the dump file using the long command given below[3]. [The encryption key is supplied on the command line as "hi".]

```
time (openssl enc -aes-256-cbc -k hi -in /tmp/lab_mnt.dump >
                    /tmp/lab_mnt.dump.aes256)
```

Notation #11: ***How many seconds did it take to encrypt the dump file? [Enter the number in the spreadsheet.]***

Notation #12: ***How big is the encrypted dump file? [Enter the number in the spreadsheet.]***

2. Clean up in preparation for the next section:

```
rm /tmp/lab_mnt.dump

rm /tmp/lab_mnt.dump.aes256

sync;  sysctl -w vm.drop_caches=2
```

3 There is no universally accepted way of adding a file suffix to show that a file has been encrypted with `openssl`, nor what cipher was used, but it seems wise to use *something* that gives an indication of how to decrypt it later, which explains the use of "aes256" at the end of the file.

VIII. Dump with Everything

In this section you will look at the performance of using dump to create a backup file, compress it, verify it, and then encrypt it. Note that doing compression **before** encryption is important; if it is done the other way around, compression may actually **increase** the file size.

1. Re-create and compress a dump file:

```
cd /lab_mnt
time dump -0 -z -f /tmp/lab_mnt.dump usr/bin
```

Notation #13: ***How many seconds did it take to dump and compress the data? [Enter the number in the spreadsheet.]***

2. Verify the compressed dump file:

```
time restore -Cf /tmp/lab_mnt.dump
```

Notation #14: ***How many seconds did it take to verify the compressed data? [Enter the number in the spreadsheet.]***

3. Encrypt the compressed dump file:

```
time (openssl enc -aes-256-cbc -k hi -in /tmp/lab_mnt.dump >
                  /tmp/lab_mnt.dump.aes256)
```

Notation #15: ***How many seconds did it take to encrypt the compressed dump file? [Enter the number in the spreadsheet.]***

IX. Clean up

After completing the lab report using the supplied template, save the spreadsheet and go to the terminal on your Linux system that was used to start the lab and type:

```
stoplab
```

If you modified the lab report and/or spreadsheet on a different system, you must copy that completed file into the directory path displayed when you started the lab, and you must do that before typing “stoplab”. When you stop the lab, the system will display a path to the zipped lab results on your Linux system.

X. Submisssion

Provide the zip file to your instructor, e.g., via the Sakai site.

Appendix – Some Unix Commands

`cat` — Display the contents of a text file to the terminal.

```
cat filename
```

`cd` — Change directory

```
cd  location
```

With no "location", you will be taken to your home directory.

`chmod` — Change the DAC permissions on a file or directory.

```
chmod permissions objectname
```

Consult Lab 1 or the man page for how to represent the permissions.

`diff` — Show the difference between two text files

```
diff file1 file2
```

`echo` — Display a string on the terminal.

```
echo string
```

Often used in scripts or to redirect to a file.

`find` — Find an object with a given kind of attribute. The basic use of find is to list all the files and directories in a given hierarchy:

```
find directorypath –print
```

`grep` — Search for a string within one or more files.

```
grep  string  file(s)
```

`ls` — List the contents and/or attributes of a directory or file

```
ls  location
ls  file
```

With no "location" or "file" it will display the contents of the current working directory.

`man` — Manual

```
man command
```

Displays the manual page for the given "command". To see another page press the space bar. To see one more line press the Enter key. To quit before reaching the end of the file enter 'q'.

`mkdir` — Create a new directory

```
mkdir directoryname
```

`more`	Display a page of a text file at a time in the terminal `more file` To see another page press the space bar. To see one more line press the Enter key. To quit before reaching the end of the file enter ‘q’.
`mv`	Move a file and/or change its name. `mv currentname newlocation_andor_name`
`pwd`	Display the present working directory `pwd`
`stat`	Displays metadata about a file system object. `stat filename`
`su`	Super user (change to root)
`time`	Measure the time it takes to execute a command to completion. `time command`
`touch`	Change the modification date on the given file. If the file does not exist, it will be created. `touch filename`
`wc`	Count the number of words in a given text file. Given the “-l” option (for “lines”), it will return the number of lines in a text file: `wc –l filename`