Linux For Embedded Systems

For Arabs

Course 102:
Understanding Linux

Ahmed ElArabawy

Lecture 24:
Archiving and Compression of Files

Archiving Files

- Archiving is to combine a group of files organized in a tree structure into one file
- This enables easier handling, such as for backup or transfer purposes
- This is a separate procedure from compression
- The tool used for archiving files is "***tar***"

Archiving Files (tar Command)

$ tar <options> <destination archive file> <directories/files to archive>

- This command is used to archives a group of files/directories into a single <u>tar file</u>
- It is also used to un-archive a tar file into its original directory tree structure
- To archive (tar) a group of files and directories,

  $ tar cvf <archiveFile.tar> <a set of files and directories>

  - 'c' for create
  - 'v' for verbose
  - 'f' for file
- Examples:

  $ tar cvf my-docs.tar ~/Documents/pdfs/

  $ tar cvf selected-files.tar ~/file-1.txt ~/Documents/file-2.txt ~/*.pdf

Working with a tar file (tar Command)

- Now we have the tar file, and we can do the following with it,
  - To un-archive (un-tar) a file into all of its original components

    $ tar xvf <archiveFile.tar>
    - '**x**' for extract
  - To extract some of the files/dirs from a tar file

    $ tar xvf <archiveFile.tar> <files/dir to untar>
  - To show the contents of a tar file

    $ tar tvf <archiveFile.tar>
- Examples:

```
$ tar cvf tar-file.tar  *.pdf   *.txt
$ tar tvf tar-file.tar
$ tar xvf tar-file.tar  *.pdf
$ tar xvf tar-file.tar
```

Using tar inside a find Command

- Let us say, we need to archive all pdf files in our home directory.
- We will need the **find** command to search for those files
- The outcome of the **find** command is then passed to the **tar** command
- This can be achieved by using a <u>pipe</u>,

```
$ find ~ -name '*.pdf' | tar cvf file.tar
```

- Another way to achieve the same target is to use the **tar** command as the user defined command to be executed inside the **find** command

```
$ find ~ -name '*.pdf' -exec tar cvf file.tar '{}' '+'
```

Compressing Files

- Compressing files is the procedure of reducing the size of the file by a tool that removes any redundant data in the file
- There are multiple formats for compressed files,
  - '**.gz**' ➔ use the tools **gzip**, and **gunzip**

    $ gzip <file> file is compressed into file.gz

    $ gunzip <file.gz> file.gz is flattened into file
  - Better compression '**.bz2**' ➔ use the tools **bzip2** and **bunzip2**

    $ bzip2 <file>

    $ bunzip2 <file.bz2>
  - Even better compression '**lzma**' ➔ use the tools **lzma** and **unlzma**

    $ lzma <file>

    $ unlzma <file.lzma>
- Note that these tools,
  - They deal with a single file and not a group of files, to be able to compress a group of file in a directory tree, you need to archive them first
  - They replace the file with its compressed/flattened version. The original file is deleted

Accessing Compressing Text Files

- If a text file is compressed to a **.gz** format, we can still access it without uncompressing it using a set of tools (Z-tools)

```
$ gzip my-file.txt
```

- To view the file contents, we use the **zcat** command (similar to **cat** command for uncompressed text files)

```
$ zcat my-file.gz
```

- If the file is long, and we need to view it page by page, we can use the commands **zmore** and **zless** (similar to **more** and **less** commands for uncompressed text commands)

```
$ zless my-file.gz
$ zmore my-file.gz
```

- We can also search inside the compressed text document via the **zgrep** and **zegrep** commands (similar to **grep** & **egrep**)

More on Compressing Files

- Note that compression preserves the permissions and timestamp of the files. After uncompressing the compressed file, we end up with the same permissions and timestamps of the original file
- It is not useful to try to compress an already compressed file, it will probably increase its size due to
  - No reduction in file size since reduction already happened the first time
  - Added more overhead data (meta-data) at the second compression time

Mixing Archive and Compression

- Archiving means combining multiple files organized in a tree structure into a single file
- Compression is to reduce the size of a single file
- So if we want to perform both, we can do this in two steps,

```
$ tar cvf  pdf-files.tar *.pdf      (this will generate the pdf-files.tar)
$ gzip pdf-files.tar                (this will generate the pdf-file.tar.gz)
```

- We can replace the **gzip** with any other compression tool (depending on the desired compression format)
- We can perform both tasks (archiving and compression) in a single step

Using tar for Archive + Compress

```
$ tar <options>  <compressed file> <files or folders>
$ tar <options>  <compressed file>
```

- With the right set of options to the **tar** command, we can perform both archiving and compression (to the desired compression format)
  1. Add the option '**z**' to perform **gzip** or **gunzip**
  2. Add the option '**j**' to perform **bzip2** or **bunzip2**
  3. Add the option '**--lzma**' to perform **lzma** or **unlzma**
- Examples:

```
$ tar cvzf  my-file.tar.gz   ~/Documents/*.doc
$ tar xvzf  my-file.tar.gz

$ tar xvjf file.tar.bz2
$ tar cvf --lzma my-file.tar.lzma  ~/my-project/
$ tar tzvf my-file.tar.gz
```

Other Archive +Compression Tools (the rar Command)

- The **rar** tool can be used to archive + compress or extract + deflate files to/from the **.rar** format archive file
- The tool need to be installed first

```
$ sudo apt-get install rar
```

- The rar tool is very powerful tool, we will only cover the basics of it
- To add a file (group of files) to a rar archive

```
$ rar a my-archive.rar ~/project/*.pdf
```

  - This will add the pdf files to the my-archive.rar archive.
    - If the archive does not exist, it will be created
    - If archive exists, it will be appended with these files
- To lock the archive to stop more file additions

```
$ rar k my-archive.rar
```

Accessing the RAR Archive file (the rar Command)

- To list the contents of the archive,

```
$ rar l my-archive.rar
```

- To delete a file from the archive

```
$ rar d my-archive.rar   my-file.pdf
```

- To extract the files in the current directory without maintaining the original hierarchy (flat set of files, without creating subdirectories

```
$ rar e my-archive.rar
$ rar e my-archive.rar  file-1.pdf
```

- To extract the files in the proper file hierarchy (to maintain the directory structure). This created subdirectories inside the current directory

```
$ rar x my-archive.rar
$ rar x my-archive.rar file-2.pdf
```

Refreshing the Archive (The rar Command)

- Let us assume we are archiving all your project files

```
$ rar a my-archive.rar  ./my-project
```

- Then we modified some of the files that has been archived
- Now we need to update the archive with the new modified files

```
$ rar f my-archive.rar     (only refreshes the local files)
$ rar f my-archive.rar *  (refreshes all subdirectories as well)
```

Other Archive +Compression Tools (the zip/unzip tool)

- The tools ‘**zip**’ and ‘**unzip**’ can perform windows format ‘**.zip**’
- Note that those tools perform both archiving and compression

```
$ zip -r file.zip ~/my-documents/      (r for recursive)
$ unzip file.zip
```

- Note:
  - If we use ‘**zip**’ on a directory to **file.zip** and that file already exists, then **file.zip** will be updated,
    - New files added
    - Modified files updated

Checking File Integrity (The md5sum Command)

$ md5sum <file to be protected> > <checksum file>

$ md5sum -c   <checksum file>

- If we have a file and you need to make sure it is not modified or corrupted over time or after transferring it, we can protect its integrity by calculating a checksum and store it

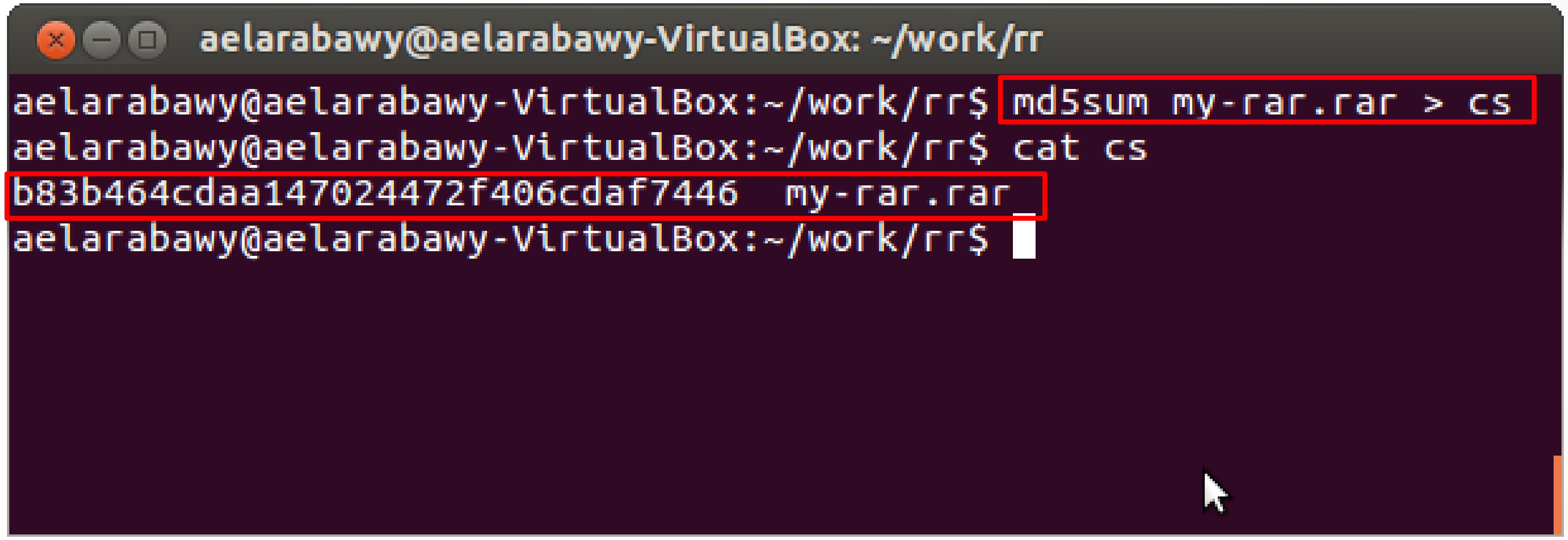
```
$ md5sum my-file.txt > checksum-file
```

- Then at a later time or after some procedure, that may affect its, we can recalculate the checksum and compare it to the original

```
$ md5sum -c checksum-file
```

- We should get a status if the checksum matches the current file or not
- Note that the checksum file will contain a 128 bit message digest of the file content

Checking File Integrity (The md5sum Command)


aelarabawy@aelarabawy-VirtualBox: ~/work/rr

```
aelarabawy@aelarabawy-VirtualBox:~/work/rr$ md5sum my-rar.rar > cs
aelarabawy@aelarabawy-VirtualBox:~/work/rr$ cat cs
b83b464cdaa147024472f406cdaf7446  my-rar.rar
aelarabawy@aelarabawy-VirtualBox:~/work/rr$ 
```

Checking Integrity for Multiple Files

- In case we need to check integrity for multiple files, we can use one of the following methods
  - We can archive the directory structure to be protected, and perform an md5sum on the archive file

```
$ rar a my-archive.rar ~/my-project/
$ md5sum my-archive.rar  > csum
```

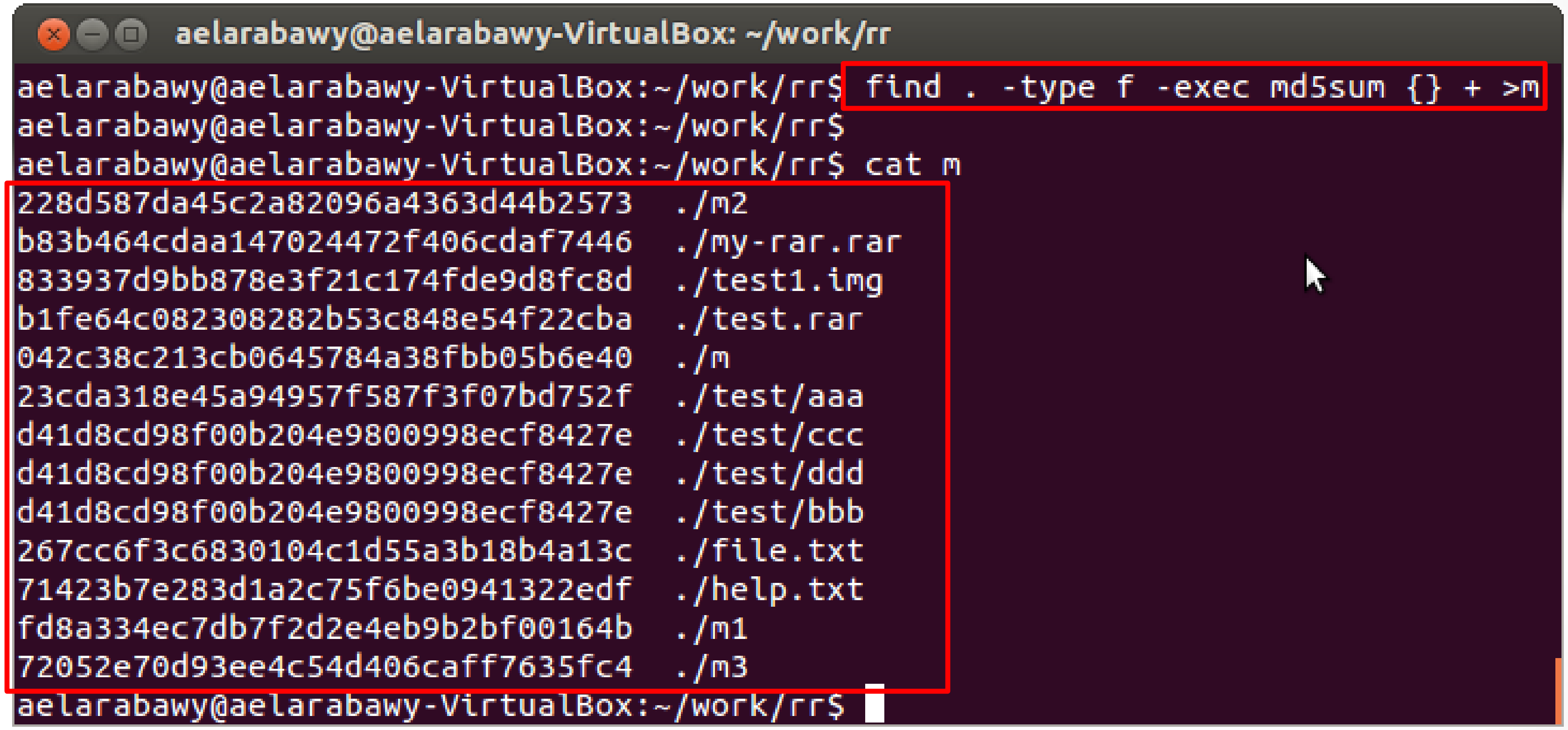
  - We can use shell expansion wildcards to select the files to be protected

```
$ md5sum * > csum
$ md5sum *.java > csum
```

  - We can use the find command to select the files to be protected

```
$ find  ~ -type f   -exec  md5sum {} + > csum
$ find  ~ -type f   -name \*.pdf   | md5sum > csum
```

Checking Integrity for Multiple Files



aelarabawy@aelarabawy-VirtualBox: ~/work/rr

```
aelarabawy@aelarabawy-VirtualBox:~/work/rr$ find . -type f -exec md5sum {} + >m
aelarabawy@aelarabawy-VirtualBox:~/work/rr$
aelarabawy@aelarabawy-VirtualBox:~/work/rr$ cat m
228d587da45c2a82096a4363d44b2573  ./m2
b83b464cdaa147024472f406cdaf7446  ./my-rar.rar
833937d9bb878e3f21c174fde9d8fc8d  ./test1.img
b1fe64c082308282b53c848e54f22cba  ./test.rar
042c38c213cb0645784a38fbb05b6e40  ./m
23cda318e45a94957f587f3f07bd752f  ./test/aaa
d41d8cd98f00b204e9800998ecf8427e  ./test/ccc
d41d8cd98f00b204e9800998ecf8427e  ./test/ddd
d41d8cd98f00b204e9800998ecf8427e  ./test/bbb
267cc6f3c6830104c1d55a3b18b4a13c  ./file.txt
71423b7e283d1a2c75f6be0941322edf  ./help.txt
fd8a334ec7db7f2d2e4eb9b2bf00164b  ./m1
72052e70d93ee4c54d406caff7635fc4  ./m3
aelarabawy@aelarabawy-VirtualBox:~/work/rr$
```

Linux 4

Embedded Systems

http://Linux4EmbeddedSystems.com