Opening Exercise: Reaching Basic Bash Proficiency

Welcome to develeap's Linux and Bash exercise!

Before the first day of the training, we expect every one of our trainees to be comfortable with the unix command line, and able to perform basic tasks:

- File and folder manipulation
- Script writing
- etc.

There are many excellent resources on the web to learn the Unix command line (and in our case - Bash). We recommend the free ebook "The Linux Command Line".

If you are not using linux on your laptop, we **strongly** recommend that you install ubuntu (possibly as a dual boot) and use it as your primary operating system. All work in the course will be on linux.

The following exercise is meant to help you review what you know, and make sure you come prepared for your first day. You are **required** to fulfil the instructions and upload your script to the link provided in the email explaining this task.

Exercise 1: Manipulating files and folders

Write a Bash script called **unpack.sh** which can unpack multiple packed files (a.k.a "archives"), and even traverse folders recursively and unpack all archives in them - regardless of the specific algorithm that was used when packing them. Following is the exact synopsis:

```
unpack [-r] [-v] file [file...]
```

Given a list of filenames as input, this script queries each target file (parsing the output of the **file** command) for the type of compression used on it. Then the script automatically invokes the appropriate decompression command, putting files in the same folder. If files with the same name already exist, they are overwritten.

Unpack should support 4 unpacking options - **gunzip**, **bunzip2**, **unzip**, **uncompress**. Adding more options should be VERY simple.

Note that file names and extensions have no meaning - the only way to know what method to use is through the **file** command!

If a target file is not compressed, the script takes no other action on that particular file. If the target is a directory then it decompresses all files in it using the same method.

Command **echo**s number of archives decompressed

Command returns number of files it did NOT decompress

Additional switches:

-v - verbose - **echo** each file decompressed & warn for each file that was NOT decompressed

-r - recursive - will traverse contents of folders recursively, performing unpack on each.

The following page contains some examples:

```
~$ unpack my-zip-file
 Decompressed 1 archive(s)
```
What happened? The contents of my-zip-file were written into ~. Command returned 0 (success)

```
~$ unpack my-zip-file my-bz2-file
 Decompressed 2 archive(s)
```
What happened? The contents of my-zip-file and my-bz2-file were written into ~. Command returned 0 (success)

```
~$ unpack some-text-file
 Decompressed 0 archive(s)
```
What happened? Command returned 1 (failure for 1 file)

```
~$ unpack -v *
 Unpacking my-bz2-file…
 Unpacking my-zip-file...
 Ignoring some-text-file
 Decompressed 2 archive(s)
```

What happened? Assuming the directory ~ contained some-text-file, my-zip-file and my-bz2-file then the contents of my-zip-file and my-bz2-file were written into ~ and command returned 1 (failure for 1 file)

```
~$ unpack -v some-folder
Unpacking a.zip...
Unpacking b.bz2...
Decompressed 2 archive(s)
```

What happened? Assuming that some-folder had 2 archives and no other files then they were unpacked. If it had any subfolders, they were ignored and command returned 0 (success)

```
~$ unpack -r some-folder
Decompressed 17 archive(s)
```

What happened? Assuming that some-folder (and all of its subfolders) had a total of 20 files of which 17 where archives, this would extract them all - each at its location - and return 3 (failure for 3 files)

The attached zip file contains some archives, to help you test the script as you write it.