Assignment 3

A deduping archiver

Due: Friday 26 May, 9pm

Your task for this assignment is to write an archiver – a program that bundles files together and compresses them. Unlike existing archivers such as `tar`, your archiver, `dear`, will offer several options for handling duplicate files.

Basic functionality

You must write two programs: `dear` and `undear`.

`dear` is a *deduping archiver*. You should be able to run it like this:

`dear [`*`options`*`] `*`outfile indir`*

This should use `tar` to archive the files in the directory *`indir`* into a file called *`outfile`*`.tar`. Depending on the command-line options chosen, this tarfile may then be compressed using `gzip`, `bz2`, or `compress`.

Before using `tar` to create the archive, your program must check for duplicates. A *duplicate*, in this sense, is a file that has the same name and contents as another file in the archive. In order to save space, `dear` should only archive one copy of any file that has a duplicate, and keep a record of the location of the duplicate. How you store this record is up to you but it must be stored in the tarfile, so that users don't have to keep track of two files in order to be able to dearchive. We suggest that you try to keep your record of duplicates reasonably small. The whole point of this program is to minimize the size of the archive file!

Your system should not modify or delete any files in *`indir`*, even if they are duplicates. (If *`outfile`*'s path puts it under *`indir`*, you should print an error message and terminate.)

The user will be able to specify how duplicate files are handled when the files are unarchived, using `undear`.

The options that `dear` must support are:

- `-g`: compress result with `gzip`
- `-b`: compress result with `bzip2`
- `-c`: compress result with `compress`

If no compression is specified, `dear` should not compress the files but should leave the archive as a tarfile.

The options that `undear` must support are:

- `-d`: delete duplicate files
- `-l`: unarchive duplicate files as soft links to the original (using `ln -s`)
- `-c`: unarchive duplicate files as copies of the original

It should not be necessary to implement command line options to tell `undear` how to decompress compressed archives. It should check the file extension to determine whether decompression is necessary: `.gz` for `gzip`, `.bz2` for `bzip2`, and `.Z` for `compress`.

Detecting duplicates

To check for duplicate files, you should first check for duplicate filenames. If you find two files with the same filename, you should check to see whether they are identical – there are a few

standard programs that let you do this. Be careful, though: don't assume that all files are text files.

Your program is not required to handle duplicate files that have *different* names. It must *not*, however, assume that all files that have the same name are duplicates – for example, if there are two files called `main.c` in the collection to be archived, they should only be deduped if their contents are the same. Overwriting one project's `main()` with another project's `main()` would be a disaster!

Language requirements

For this assignment, you may use **any combination of `bash` scripting and Perl scripting**. You may also use other programs provided their licensing terms permit it. If your system requires the installation of third-party packages, your `README` must say so as part of the installation instructions.

We can't stop you developing in OSX or Windows, but your system will be tested under xubuntu. If your installation instructions don't work or your system doesn't function correctly on the target system, you **will** lose marks.

Submission requirements

You should submit a tarball (`.tar.gz`) containing all the scripts necessary to run your two programs. This must include, in the root directory, a `README` file containing installations and usage instructions. This must include documentation of any known limitations of your system.

Marks breakdown

Marks for this assignment will be broken down as follows:

- Functional correctness: **5 marks**, comprising
  - Creating the deduped tarfile: **2 marks**
  - Creating/reading the record of duplicates: **2 marks**
  - Compressing/uncompressing correctly: **1 mark**
- Following submission instructions: **2 marks**
- Code structure, style, and readability: **3 marks**[1]

This assignment is worth **10%** of your total mark for FIT3042.

[1] This might seem like a lot of marks, but Perl scripts are very hard to read unless you approach their creation with a great deal of discipline – you have been warned!