CSCI415—Systems Programming

Spring 2014

Programming Assignment 6

A Message Archive with Shell Scripts

This project requires that you design and implement a shell script that allows us to treat a directory of files as a message archive. Your script will allow maintenance and display of the archive. You are not allowed to use any of your own compiled code for this assignment. You must use the scripting facilities provided by `/bin/sh` on the departmental Linux systems; you may use *embedded* `awk` scripts if they help your programming task.

An Archive Directory

The structure of an archive directory is as follows:

- All files, with the exception of a file named `bounds` (see below), have names which look like an integer value, one or greater.
- Each of the files with numeric names is a text file with the following format:
  - The first line starts with the string "Subject: " followed by a string that extends to the end of the line (the subject of the message).
  - The second line starts with the string "Date: " followed by a string that extends to the end of the line (the date on which the message was posted).
  - The third line is `=====` (five `=` characters). This line is a separator between header information above and message text below.
  - The remainder of the file consists of lines of text with no prescribed format.
- The directory will contain a file named `bounds` which contains a single line of text:

  `first:last`

  Here, `first` denotes the file with a name that appears as the smallest integer of all message files in the directory. The value `last` denotes the file whose name is the largest integer value.

Here is an example of what a message archive directory might contain.

```
% cd sample
% ls -l
total 12
-rw-r--r--1 kearns root      95 Apr 1 10:18 18
-rw-r--r--1 kearns root     123 Apr 1 10:19 19
-rw-r--r--1 kearns root     121 Apr 1 10:20 20
-rw-r--r--1 kearns root       6 Apr 1 10:40 bounds
%
```

The bounds file in the example contains a single line:

```
18:20
```

Note that there can be no gaps in the message sequence in a valid message archive. Files whose names do not look like integers and are not named bounds may be in the directory, but they should not affect the operation of your script.

Managing the Message Archive

You must design and implement a shell script, `archive`, that allows an administrator to automate certain management functions for the archive. The archive script may be invoked six different ways:

- Initializing the archive:

  ```
  archive -c dirname
  ```

  will check to see if the specified directory exists and will terminate with an error message if that is the case. Otherwise, it will create the directory and install a bounds file containing

  ```
  0:0
  ```

  to indicate an empty archive.

- Adding a message to the archive stored in the directory *`dirname`*:

  ```
  archive dirname -a filename -s subject_string
  ```

  will add the contents designated text file to the archive. The subject line will be constructed from the specified subject string. The date line will be constructed automatically. The message should reside in a file whose name is 1 greater than the name of the file with the "biggest" name currently in the archive. The bounds file must be adjusted appropriately. The argument order may also be:

  ```
  archive dirname -s subject_string -a filename
  ```

  Note that all arguments are mandatory. The subject string must contain only printable alphabetic and numeric characters, spaces, and the punctuation characters listed in the Notes section below (you must enforce this). The subject string can be no longer than 31 characters; the filename no more than 511 characters (again, these limits must be enforced by your script).

- Deleting a message from the archive stored in the directory *`dirname`*:

  ```
  archive dirname -d msgnum
  ```

  will remove the designated file, the one with name *`msgnum`*. It will also adjusts `bounds`, perhaps after compacting the file names. File name compaction requires that two conditions hold:

C1. The message files in the archive must always be numbered consecutively; and

C2. The upper bound (bigger number in the bounds file) can never be decreased.

For example, in the sample archive, the operation

```
archive sample -d 19
```

will remove file 19 from the archive directory. In order to satisfy conditions C1 and C2, file 18 must be renamed as 19. If, from the original example we do an

```
archive sample -d 20
```

then file 20 is deleted, old 19 is renamed 20 and old 18 is renamed 19. Finally, if from the original example we do

```
archive sample -d 18
```

file 18 is deleted, and no compaction is required.

In all three cases, the bounds file must be changed to

```
19:20
```

- Getting a subject list from an archive:

  ```
  archive dirname -S
  ```

  results in the script generating output (on its stdout). There will be one line of output per message in the archive. In the context of the sample archive directory, the output will be

  ```
  18 This is a test   [Wed Apr 2 10:17:35 EDT 2014]
  19 This is a test 2 [Wed Apr 2 10:18:35 EDT 2014]
  20 This is a test 3 [Wed Apr 2 10:19:35 EDT 2014]
  ```

  The text immediately to the right of the file name is the subject of the message (with the "Subject: " removed). The text between the square brackets is the date on which the message was inserted in the archive (the date line of the file with the leading "Date: " removed). It would be nice if your output were nicely aligned, but if you separate strings by a space, it's good enough.

- Searching the archive stored in the directory *dirname*:

  ```
  archive dirname -ss string
  archive dirname -sb string
  ```

  will scan the archive for occurrences of the designated string (not a more general regular expression). If the `-ss` option is used, only the subject lines are searched; if `-sb` is used, only the body of the message is searched. The output from either search is a list of message numbers where matches are found. A message should be listed at most once. If there are no matches, there should be no output.

Notes

- If any operation cannot take place (for example, if we name a file that does not exist when trying to add or delete a message), it should terminate with an appropriate error message.
- You do not need to concern yourself with multiple simultaneous invocations of `archive`. The operative assumptions are that only you can run archive and you are careful not to try to do several things at once (running archive in several windows).
- You should also not be concerned with protection for the directory or files in the directory (as long as the TA can access files and directories to test).
- Treat `archive` as a system program. You should make it resistant to stupid or malicious behavior.
- The punctuation characters comma, period, colon, semicolon, exclamation mark, and question mark may appear in a subject. If necessary, the user (whoever is actually executing `archive`) must shield characters on the command line from interpretation by the shell. To be more explicit, only the following types of characters are allowed in subjects: upper and lower case alphabetics, digits, spaces, and the aforementioned punctuation characters.
- Repeating the programming restrictions: the archive program must be a shell script, interpreted by `/bin/sh` on a departmental system. You may use `awk` scripts as part of your solution as long as they are embedded in the shell script. Note that you are not required to use `awk` as part of your script.

Submission

In the directory where you develop archive, do the following:

- `gpg -r CSCI315 --sign -e archive`
- `cp archive.gpg ~`
- `chmod 0644 ~/archive.gpg`

You must create your submission file by the deadline specified on the course assignments web page.