PLC Bash

CSCI-344 Assignment

07/28/2019

1 Introduction

For this assignment you will be required to implement a small program in Bash as outlined in class.

You have been asked by a secret government agency to create a way to decode/encode documents. The agency uses a word replacement encoding. The idea is to have a key file containing an encoded word and its decoded replacement. The agency and agent will use this file to encode/decode data.

2 The Tasks

You must create three bash files for this assignment:

- `encoder.sh`: This will contain the code for encoding a plain text file into an encoded file using the provided secret key file.

  ```
  encoder.sh <plain_text_file> <secret_key_file> <encoded_file>
  ```

  This will read in the plain text file and replace all words in the file with its encoded value provided in the secret key file. It will write the results out to the encoded file. All line and spacing structure must be maintained between the plain text and encoded file.
  The original and secret key files must remain unchanged.
- `decoder.sh`: This will contain the code for decoding an encoded text file into a plain file using the provided secret key file.

  ```
  decoder.sh <encoded_file> <secret_key_file> <plain_text_file>
  ```

  This will read in the encoded text file and replace all words in the file with its decoded value provided in the secret key file. It will write the results out to the plain text file. All line and spacing structure must be maintained between the plain text and encoded file.
  The encoded and secret key files must remain unchanged.
- `tester.sh`: This file will test the functionality of the encoding and decoding. This will run both your scripts and ensure the encoding and decoding scripts work. After each script it will compare the output with the original plain text file. After encoding it should find them to be different, after decoding it should find them to be the same. You can use the diff command for this part. You have some freedom with the output of this, but it should output to the command line and provide enough information to track the progress. Run using:

  ```
  tester.sh <plain_text_file> <secret_key_file>
  ```

  The original files must remain unchanged.

The secret key file will contain <plain_text>, <secret_text> pairs. There will be only one pair per line. Example:

```
the, bar
hello, baz
professor, awesome
```

3 Example

Given the following secret_key.txt:

```
this, zoo
file, underwater
you, fast
need, waste
decode, walk
is, garage
to, run
read, father
it, animal
can, foo
a,water
secret, mountain
```

And the plain text file, plain_text.txt:

```
this is a secret file
to read this file you need
to decode it
can you read this
```

Running the command

```
./encoder.sh plain_text.txt secret_key.txt encoded_text.txt
```

will result in a file called encoded_text.txt containing:

```
zoo garage water mountain underwater
run father zoo underwater fast waste
run walk animal
foo fast father zoo
```

Then running

```
./decoder.sh encoded_text.txt secret_key.txt decoded_text.txt
```

should result in an exact match of the original plain text file and the new file created, decoded_text.txt.

You can use any bash functionality, as long as it runs on the CS lab machines. Therefore, you should test it on the CS machines before submitting. You cannot use external commands such as sed, perl, gawk, mv, etc. Everything must be done by Bash.

If a word cannot be matched in the secret key file, it is left as is in the encoded/decoded files. A word will either be a plain text word or a secret key text word, not both.

4 Submission

Zip the source file for your program into a zip file called `pa5.zip` and submit to the proper myCourses' Assignment box. A penalty will be enforced if submission/naming rules are not followed.

Late Submissions will not be accepted!

Questions will not be answered within 24 hours of the due date.

5 Grading

This phase will be graded as follows:

- (40%) Proper encoder functionality
- (40%) Proper decoder functionality
- (20%) Proper tester functionality.

The following penalties will be enforced above and beyond any points lost above:

- A grade of 0 will be given if it does not run without errors; this includes infinite loops. If a section of code is causing errors comment it out. This will allow me to still see it and potentially award partial credit.
- A grade of 0 will be given if it does not run on the CS machines.
- -10% if the files are not named as requested.
- -10% if the zip file is not named as requested.
- -20% if the submission is not zipped.