Tuesday, March 1, 2022 7:33 PM

R Noah Padilla CS 3360 TR 1:30-2:50 3/2/2022

You need to find a secret message buried in an innocent-looking list of seafood information. ortunately you know how to extract it since you have intercepted an email with instructions:

Greetings Agent X... to decode, take the lines which contain a four-letter fish name. Each fish name has a two part mock SKU number, e.g. 4443-069. You'll sort by the first part, in ascending order. If it's not odd, ignore it. Each line has a one character payload, which you get from the second part of the SKU by adding 3, and using that as the Ascil code for a character, for example, 97 for 'a', 98 for 'b', and so on. Then if it's a letter from a to z, look it up and replace it with the corresponding character in the codebook. Case doesn't matter. For example,

```
hake 3-115
       snapper 5-219
       bass 181-99
       tuna 9-105
       char 007-106
       pike 846-723
would, after filtering and sorting, give the sequence below, which would then map as
follows:
       3-115
                              115 -> 118 -> v
       007-106
                              106 -> 109 -> m
        9-105
                              105 -> 108 -> |
       181-99
                              99 -> 102 -> f
so if the codebook includes
       fр
        m t
```

the final message will be "utep".

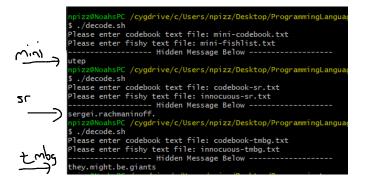
v u

The test data above is found in mini-fishlist.txt and mini-codebook.txt on the course homepage.

- Part 1. [8 points] Use Unix commands to discover the secret message in Innocuous-sr.csv using codebook-sr.txt. Hint: sort, join, awk, sed and grep may be helpful. Please use mostly Unix commands; do not, for example, write a Python program and just call it from bash.
- Part 2. [7 points] Write a bash script that automates this process, taking as input a fishy file and a codebook, and outputs the secret message found using the rules above.
- Part 3. [1 point] Run your scropt on innocuous-tmbg.txt using codebook-tmbg.txt and report the secrete message found.
- Hand in hardcopy including: 1) the secret message, 2) your bash script, and 3) a one-paragraph report noting any limitations, cleverness, or special features. Perfectionism is discouraged, and in particular your code need not be robust, as long as weaknesses are documented.

Due March 3

1) The secret messages for each of the files



| | <u>File</u> | Hidden Message |
|--|--------------------|-----------------------|
| | mini-fishlist.txt | utep |
| | innocuous-sr.txt | sergei.rachmaninoff. |
| | innocuous-tmbg.txt | they.might.be.giants. |

2) My bash script

| #!/bin/bash |
|---|
| read -p "Please enter codebook text file: " codebook #mini-codebook.txt codebook-sr.txt codebook-tmbg.txt read -p "Please enter fishy text file: " fishyfile #mini-fishlist.txt innocuous-sr.txt innocuous-tmbg.txt |
| #1) Save only the SKU's into their own file echo awk '{print \$2}' \$fishyfile sort -tnk1 > SKUs.txt |
| #1) Sorts the SKU's 2)Looks at left part of SKU and keeps odd ones 3) Adds 3 to the right value 4) turn the right side number to ASCII char 5) save it into a file echo awk -F "-" '{print \$1, \$2}' SKUs.txt awk '{if(\$1\%2==1){print \$1,\$2;}}' awk '{print \$1,\$2+3}' awk 'BEGIN{for(n=0;n<256;n++)chr[n]=sprintf("%c",n)}{print chr[\$2]}' > semideciphered.txt |
| #Puts the codebook into a hashtable/dictionary and prints them out - FOR DEBUGGING ONLY #awk 'FNR==NR {a[\$1] = \$2; next} END {for (key in a) { print key, a[key] , \$2 } }' codebook-sr.txt semideciphered.txt |
| echo " Hidden Message Below" |
| #1) puts the codebook into a dictionary then prints the decoded message using that dictionary 2) put into 1 line echo awk 'FNR==NR {a[\$1] = \$2; next} {print a[\$1]}' \$codebook semideciphered.txt awk 'NR{printf "%s",\$0;next;}1' |

3) Limitations, cleverness, or special features

Regarding <u>limitations</u>, I limited myself to use awk only and not grep nor sed because I found awk to be the simplest. Awk also seemed more popular than grep and sed when looking online when learning about them. Perhaps it was just my search input, but nonetheless awk was a great tool to learn since it gives me a good idea about scripting. Regarding <u>cleverness</u>, I thought the most clever part of this program was using a hashtable/dictionary data structure in awk. It was super helpful and was easy to understand despite the syntax and semantic difference to imperative programming languages. One of the features I consider to be <u>special</u> would be the input from the user - the codebook and fishy file are not hardcoded so its easier for the user to input the text files of their choice. A weakness that this script has is handling non-text files as well as incorrect file name inputs. Overall this assignment was a good learning experience and I truly enjoyed it.