Total Points: 100

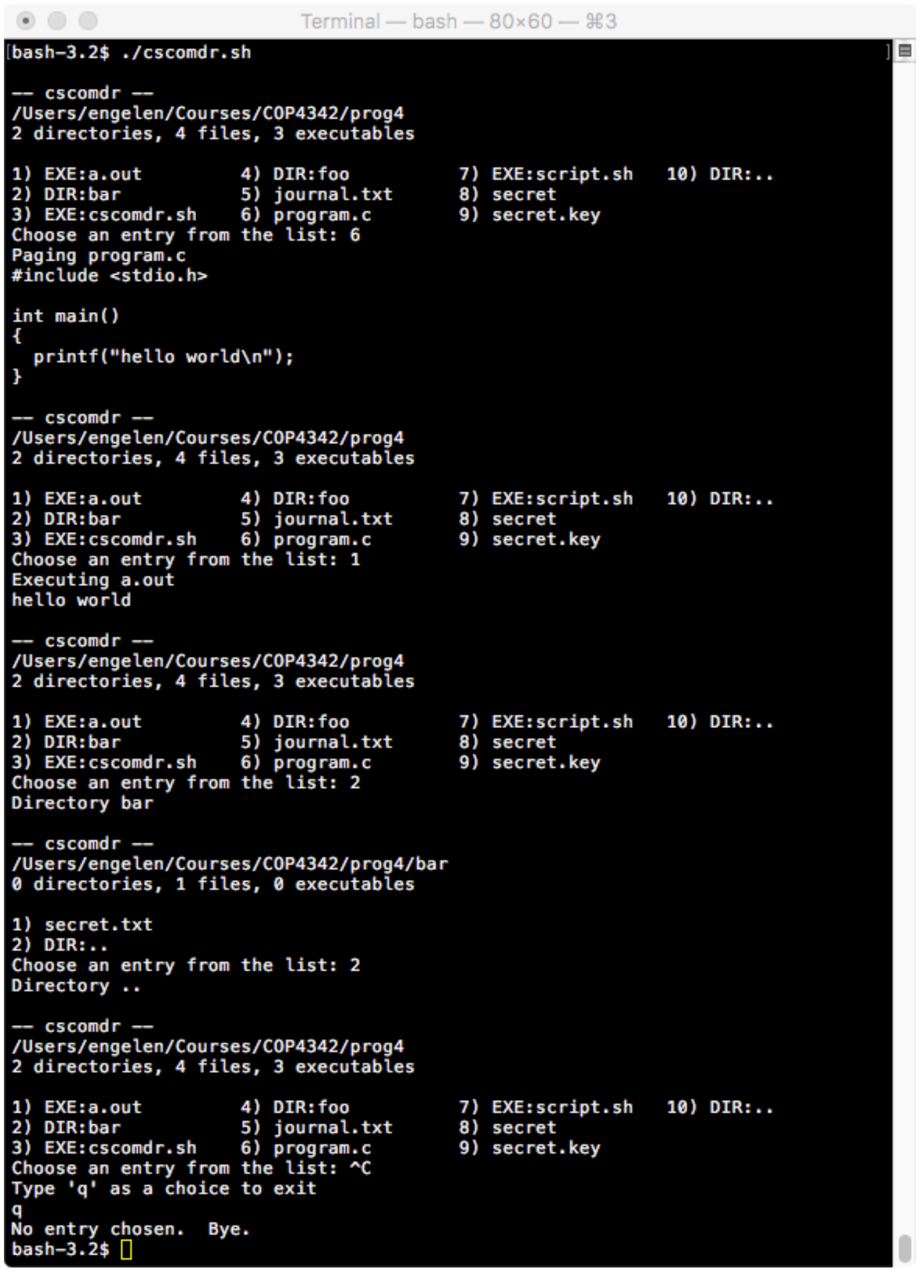
COP4342 – UNIX Tools

Assignment #4: A "shell commander"

Instructor: Dr. Robert Van Engelen   Teaching Assistant: Joydeep Das   Spring 2018

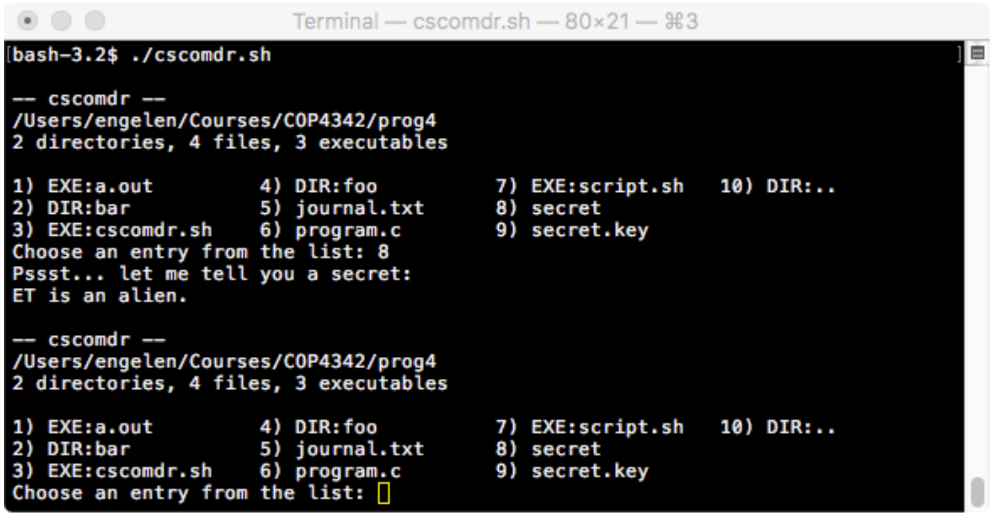
Description: A "shell commander" script to navigate directories and open files via a simple user interface (UI).

Assignment: In this programming assignment, you will implement a script cscomdr.sh that presents a simple UI to select directories to navigate to, files to open for reading and executables to execute. The UI presents a menu showing the directories, regular files and executables to select. See the example in the figure below:



```
bash-3.2$ ./cscomdr.sh

-- cscomdr --
/Users/engelen/Courses/COP4342/prog4
2 directories, 4 files, 3 executables

1) EXE:a.out          4) DIR:foo          7) EXE:script.sh   10) DIR:..
2) DIR:bar            5) journal.txt      8) secret
3) EXE:cscomdr.sh     6) program.c        9) secret.key
Choose an entry from the list: 6
Paging program.c
#include <stdio.h>

int main()
{
  printf("hello world\n");
}

-- cscomdr --
/Users/engelen/Courses/COP4342/prog4
2 directories, 4 files, 3 executables

1) EXE:a.out          4) DIR:foo          7) EXE:script.sh   10) DIR:..
2) DIR:bar            5) journal.txt      8) secret
3) EXE:cscomdr.sh     6) program.c        9) secret.key
Choose an entry from the list: 1
Executing a.out
hello world

-- cscomdr --
/Users/engelen/Courses/COP4342/prog4
2 directories, 4 files, 3 executables

1) EXE:a.out          4) DIR:foo          7) EXE:script.sh   10) DIR:..
2) DIR:bar            5) journal.txt      8) secret
3) EXE:cscomdr.sh     6) program.c        9) secret.key
Choose an entry from the list: 2
Directory bar

-- cscomdr --
/Users/engelen/Courses/COP4342/prog4/bar
0 directories, 1 files, 0 executables

1) secret.txt
2) DIR:..
Choose an entry from the list: 2
Directory ..

-- cscomdr --
/Users/engelen/Courses/COP4342/prog4
2 directories, 4 files, 3 executables

1) EXE:a.out          4) DIR:foo          7) EXE:script.sh   10) DIR:..
2) DIR:bar            5) journal.txt      8) secret
3) EXE:cscomdr.sh     6) program.c        9) secret.key
Choose an entry from the list: ^C
Type 'q' as a choice to exit
q
No entry chosen.  Bye.
bash-3.2$
```

Hunting for secrets: your cscomdr.sh should decrypt files named “secret”. It is assumed that these “secret” files contain a base64-encoded AES-256-cbc-encrypted message that is decrypted using the key stored in “secret.key” located in the same directory as a “secret” file:


Terminal — cscomdr.sh — 80×21 — ⌘3

```
bash-3.2$ ./cscomdr.sh

-- cscomdr --
/Users/engelen/Courses/COP4342/prog4
2 directories, 4 files, 3 executables

1) EXE:a.out         4) DIR:foo          7) EXE:script.sh   10) DIR:..
2) DIR:bar           5) journal.txt      8) secret
3) EXE:cscomdr.sh    6) program.c        9) secret.key
Choose an entry from the list: 8
Pssst... let me tell you a secret:
ET is an alien.

-- cscomdr --
/Users/engelen/Courses/COP4342/prog4
2 directories, 4 files, 3 executables

1) EXE:a.out         4) DIR:foo          7) EXE:script.sh   10) DIR:..
2) DIR:bar           5) journal.txt      8) secret
3) EXE:cscomdr.sh    6) program.c        9) secret.key
Choose an entry from the list: 
```

- to decrypt the message stored in “secret”, you will need to execute the following OpenSSL command **openssl enc -d -a -k $key -aes-256-cbc -in secret** where **$key** is a secret symmetric key to decrypt the “secret” file contents and display “Pssst… let me tell you a secret:” followed by the decrypted secret. The OpenSSL command **openssl enc** with option **-d** decrypts the file specified by **-in secret**, using the key specified by **-k $key**, and option **-a** specifies base64 content.
- the symmetric key **$key** to decrypt a “secret” file is stored in a file named “secret.key” located in the same directory as the file “secret”. Your script should read the key from this file first and store it in the **$key** variable before decrypting “secret” with the OpenSSL command
- if the “secret.key” file does not exist in the current directory or is not readable, then you must ask the user to enter the key with “Please enter the key: ”
- note that the symmetric key is also used to encrypt plain text files, say “secret.txt”, to produce “secret” using the following command **openssl enc -e -a -k $key -aes-256-cbc -in secret.txt -out secret** where **$key** is the non-empty key and option **-e** is used for encryption of the file specified by **-in secret.txt**. The file saved is specified by **-out secret** and is saved with base64 content as specified by option **-a**. Use this OpenSSL command to create your “secret” test cases.

Command line options: cscomdr.sh takes no options and no arguments (no need to check these in your script).

Exit code: cscomdr.sh should exit with zero when no entry from the menu was chosen, e.g. typing ‘q’ then ENTER.

Helpful reading: see the textbook “quiz shell script” for inspiration (pages 504-510). Some helpful hints: to implement cscomdr.sh you will likely need the following:

- an array to store the menu items (directory entries, see quiz function choose_subj() for inspiration)
- **${#VAR[*]}** to get the array size, where **VAR** is an array
- **for** loop(s) to iterate over the menu items in the array to set/update them
- **test** (or its **[…]** short form) to check for directory, file, or executable and use this in an **if** body to add “DIR:”, “EXE:” to an array entry representing a directory or executable, respectively

- **((expr))** to update numerical variables in **expr** (see the quiz example for inspiration)
- **select** to implement your UI menu to let the user select an entry
- **case** to execute an action after the user makes a choice (make clever use of the fact that entries begin with "DIR:" and "EXE:"!)
- slicing **${VAR:NUM}** to get the part of a string in **VAR** from string position **NUM**
- **cd** to change the working directory
- **less** to page the contents of a file
- **eval** to execute a program that is given as an expression (with variables)
- **trap** to intercept ^C (CONTROL-C) and display "Type 'q' as a choice to exit" (see the quiz example)
- note that the last entry in the menu is always "DIR:.." which should be added explicitly to the array before displaying the menu with **select** in your script
- you should not display hidden files in the menu (i.e. files starting with a dot), except for ".." that should be explicitly added by the script code
- use a hashbang to ensure that your script runs with bash only
- you may assume that "secret.key" has only one line with one word constituting the secret key
- your script is **not required to handle file and directory names that have spaces** (blanks/tabs)
- when a file or directory named **NAME** is not accessible (**cd** or **less** fail) then your script should display "**NAME** is locked" without displaying error messages (i.e. use **2> /dev/null**), and return to the UI

Testing with "Easter egg hunt": on linprog navigate with your script to **/home/faculty/engelen/COP4342**. Select the file "easter-egg-hunt.txt" to get started.

Submission Requirements: A tar file named 'yourCSusername_asg4.tar' containing the following files:-

[1]. The ./cscomdr.sh script source code.

Grading Policy: Points distribution:-

[1]. Code Readability	(20 points)
[2]. Test Cases	(80 points)

Individual parts of the implementation will not be graded separately for correctness. There will be several cases to test the implementation logic as a whole. Also, keep in mind that your code will be tested on *linprog*. Students should test their code thoroughly on the *linprog* server before submission

Miscellaneous:

Honor code policy will be strictly enforced. Write the code by yourself and protect your submission.

Key Concepts: *bash programming, file manager*

Shell commands: *cd, select, case, if, for, while, eval, trap, pwd, echo, exit, function, read, variables and arrays.*