Assignment — Bash DB

CS 265 Advanced Programming Techniques

Overview

- You will create an interactive utility called **`assn2`** which maintains a simple database (a flat text file), as described below
- You will write this in Bash
- Your program will continue to accept input until the user signals **quit**.

- Develop your script on the Drexel cs.tux.drexel.edu servers
- Test that your script correctly implements the required functionality.

What to Do

When invoked, display the following prompt:

(Note the space after the prompt.)

Requirements

After startup, the script should repeatedly perform these steps, until **quit** is signalled:

1. Print a prompt consisting of a percent sign followed by a space

   ```
   % 
   ```

   To allow the user to enter input on the same line as the prompt, consider `printf` , or `echo -n`.

2. Read a line from standard input
3. Dispatch the command to the appropriate handler
   - Consider the left-most token to be the command
   - Maybe pass the arguments to a function written for each command
   - Note, the **quit** signal will probably be handled specially

Commands

`quit`
: Break out of your input loop, quit the program

`setdb`
: Set the filename for the databse file

`add`
: Add or update an item to (int) the database

`delete`
: Remove an item from the database

`printdb`
: Print all the lines from the database

If the command is not recognised, respond with:

```
Unrecognised command
```

. Otherwise, dispatch the command to an appropriate handler.

`quit`

```
quit
```

Signal **quit** . Extra arguments are ignored.

`setdb`

```
setdb <db>
```

Set current database to be be *<db>* .

- If the file exists and is readable, make the file the current database, respond with:

```
Database set to <db>
```

- If the file exists but is not readable, do *not* update the current database, respond with

```
File <db> not readable
```

- If the file does not exist, your script will create the empty named file and respond

```
File <db> created. Database set to <db>
```

- If no filename is supplied, respond with:

```
Missing Argument
```

- If too many arguments are supplied, respond with:

```
Extra arguments ignored
```

, then consider the first argument as the filename, proceed as above.

Note: You are not required to handle quotes or backslashes. You don't need to worry about escaped characters. E.g., you are not responsible for filenames with spaces.

However, our script should be able to handle a full path filename, such as **`/home/user/cs265/A2/db.txt`**

`add`

`add <product> <price>`

- If the database has not been set by a previous setdb command, your script must respond

```
Database has not been set.
```

- If there are not two arguments, respond with:

```
Incorrect syntax.
```

- Otherwise, process the update:
  - If the *<product>* does not already exist, append a new line

```
<product>:<price>
```

    to the database, and respond to stdout:

```
<product>:<price> has been added to the database
```

  - If the *<product>* exists, update it the with the new *<price>* respond with

```
<product> has been updated to <price>
```

HINT: Consider using `sed` or `AWK` for implementing the update.

`delete`

`delete <product>`

- If the database has not been set by a previous setdb command, your script must respond

```
Database has not been set.
```

- If there is not ! one argument, respond with:

```
Incorrect syntax.
```

- If the product does not exist in `<db>`, your script must respond

```
<product> does not exist in `<db>`.
```

- Otherwise, the script must delete the corresponding line from `<db>` and respond

```
<product> has been deleted from `<db>`.
```

HINT: Again you can look at sed or AWK for implementing the delete.

printdb

```
printdb
```

- If the database has not been set by a previous setdb command, your script must respond

```
Database has not been set.
```

- Ignore any arguments
- If the database is set, then pretty-print all the lines from the database

```
  Product      Price
------------  -------
Reever        $  3.25
Thingabob     $121.64
Marlinespike  $ 12.37
```

Leave at least 12 columns for the product name. At least 6 columns for the price, including the decimal and fractional portion, but not the `$`. 2 numbers after the decimal. 2 spaces after column 1, before the `$` .

If the database has been set, the script should print all lines from the database.

What to Submit

- Do not submit a .doc file, a .zip file, or a .pdf file. These formats are not correct for this assignment and will not be accepted.
- Make sure you develop and test your script on tux. If your script does not run on tux, it will not be considered correct.