Assignment 1: Shell Scripting

CS3423 - Systems Programming — Steven O'Hara - UTSA

Spring 2020

For this assignment, you will use **bash** create a simple inventory system. The system will store basic information about items and allow the user to create, read, update, delete, and total these items.

This assignment requires only the utilities used so far in the lecture notes. Further, you **may not use** `sed`, `awk`, `find`, `grep`, `Python`, `perl`, any programming language, scripting language, or other tools not otherwise permitted. The only external tool you **may** utilize is `bc`, which can be used to perform floating-point arithmetic.

Storing Item Information

Item information will be stored in text files (ending with the extension `.item`).

1. Files will be stored in a directory named **`data`**, located within the same directory as your shell script.
2. Each file will be named based on its unique item number, an integer (henceforth referred to as `item_num`) with exactly **four** digits, followed by the extension **`.item`**.
3. An item file consists of exactly three lines, in the following format:
   - `item_name` *(string **with no whitespace**)* `simple_name` *(string)*
   - `unit_price` *(floating point number)* `cur_qty` *(unsigned integer)* `max_qty` *(unsigned integer)*
   - `item_desc` *(string)*
4. The following are the example contents of an item file named **`3293.item`**:

```
btl_water Bottled Water (24/pk)
15.99 23 50
Poland Springs natural spring drinking water
```

Script Execution

When the script is run, the following should occur.

1. Upon running your script, the user should be presented with the following menu:

```
Enter one of the following actions or press CTRL-D to exit.
  C - create a new inventory item
  R - read an existing inventory item
  U - update an existing inventory item
  D - delete an existing inventory item
  T - calculate total value of an inventory item
```

2. The user then enters a corresponding one-character selection (either upper or lowercase entries should be permissible), leading to one of the following actions:

   - `C`: create a new inventory record (thus, also creating its associated data file)

     (a) From the terminal, read the following fields, one at a time, each separated on its own line:

     i. `Item number:` *(four digit unsigned integer)*
        Example input: `3293`

     ii. `Item name:` *(string containing **no whitespace**)*
        Example input: `btl_water`

     iii. `Simple name:` *(string)*
        Example input: `Bottled Water (24/pk)`

     iv. `Unit price:` *(floating point number)*
        Example input: `15.99`

     v. `Current quantity:` *(unsigned integer)*
        Example input: `23`

     vi. `Maximum quantity:` *(unsigned integer)*
        Example input: `50`

     vii. `Description:` *(string)*
        Example input: `Poland Springs natural spring drinking water`

     (b) Using the values entered by the user, create a new file in the **`data`** subdirectory (which may or may not pre-exist) based on the file naming instructions above.

     (c) Update **`data/queries.log`** by adding the following line:
     `[date] CREATED: item_num - item_name - cur_qty / max_qty`
     where:

     i. `date` is the formatted output from the shell's **`` `date "+[%Y-%M-%d %H:%m:%S]"` ``** command,

     ii. `item_num` represents the item's internal identification number (e.g. 3293),

     iii. `item_name` represents the item's internal identification string (e.g. `btl_water`), and

     iv. `cur_qty` represents the item's initial quantity as of its entry into this inventory system (e.g. 20).

     v. `max_qty` represents the item's maximum allowable inventory quantity (e.g. 50).

     (d) If the item number already exists, abandon the "Create" operation and print the following example error message, then continue with the script.
     For example: **`ERROR: item 3293 already exists`**

   - `R`: read and display an existing item's inventory information

(a) Prompt the user for an inventory item's number:
`Enter an item number:`

(b) Search for the specified inventory item using the item number provided.

(c) Print the item's inventory information in the following format:

```
Item Number:    item_num
Item Name:      item_name
Simple Name:    simple_name
Unit Price:     unit_price
Quantity:       cur_qty  / max_qty
Description:    item_desc
```

(d) If the item is not found, print the following example error and continue with the script.
`ERROR: item 3293 not found`

- `U`: update an existing inventory record

(a) Prompt the user for the following fields one at a time:

i. Item number *(four digit unsigned integer)*

ii. Item name *(string containing **no whitespace**)*

iii. Simple name *(string)*

iv. Unit price *(floating point number)*

v. Current quantity *(unsigned integer)*

vi. Maximum quantity *(unsigned integer)*

vii. Description *(string)*

(b) Using the values entered by the user, search for the specified item in the `data` subdirectory using the given item number.

(c) Update each of the corresponding fields based on the user's input. **If the user input is blank for a particular field (except for the item number), keep the existing value currently utilized within the file.**

(d) Update `data/queries.log` by adding the following line:
`[date] UPDATED: item_num - item_name - cur_qty / max_qty`
where:

i. `date` is the formatted output from the shell's `` `date "+[%Y-%M-%d %H:%m:%S]"` `` command,

ii. `item_num` represents the item's internal identification number (e.g. 8299),

iii. `item_name` represents the item's internal identification string (e.g. `btl_water`), and

iv. `cur_qty` represents the item's currently-stocked quantity (e.g. 12).

v. `max_qty` represents the item's maximum allowable inventory quantity (e.g. 85).

(e) If the item number **does not** already exist, abandon the "Update" operation and print the following example error message, then continue with the script.
For example: `ERROR: item 8299 not found`

- `D`: delete an existing item's inventory record

(a) Prompt the user for an item number:
`Enter an item number:`

(b) Delete the specified item's file from the **`data`** directory.

(c) Update **`data/queries.log`** by adding the following line:
`[date] DELETED: item_num - item_name - simple_name - cur_qty` where:

i. `date` is the formatted output from the shell's **`` `date "+[%Y-%M-%d %H:%m:%S]"` ``** command,

ii. `item_num` represents the item's internal identification number (e.g. 8299),

iii. `item_name` represents the item's internal identification string (e.g. `btl_water`), and

iv. `simple_name` represents the item's short-form description (e.g. `Bottled Water (24/pk)`).

v. `cur_qty` represents the item's currently-stocked quantity (e.g. 12).

(d) If the item number **does not** already exist, abandon the "Delete" operation and print the following example error message, then continue with the script.
For example: `ERROR: item 8299 not found`

- `T`: calculate total value of current inventory items

(a) Prompt the user for an item number:
`Enter an item number:`

(b) Calculate the total value currently in inventory for this specific item (e.g. unit price $\times$ current quantity).

(c) Print the following message with the item's number, simple name, and total value:
`item_num - simple_name - $(unit_price×cur_qty) total`

(d) If the given item number **does not exist**, abandon the "Total" operation and print the following example error message, then continue with the script.
For example: `ERROR: item 8299 not found`

- If an invalid character is entered at the main menu, print the following error message and continue with the script:
`ERROR: invalid option`

3. After an action is completed, display the menu again. This should go on indefinitely until `CTRL-D` or the end of a file is reached.

Assignment Data

An initial data set can be found in `/usr/local/courses/ssilvestro/cs3423/Spring20/assign1`. Copy this to your own assignment's directory when you are prepared to begin initial testing. You should not, under any circumstances, rely solely on this data as the only data set with which to test the functionality of your script.

Script Files

Your program should consist of six `bash` files:

- **`assign1.bash`** - the main file which is to be initially invoked
- **`create.bash`** - logic for the create option
- **`read.bash`** - logic for the read option
- **`update.bash`** - logic for the update option
- **`delete.bash`** - logic for the delete option
- **`total.bash`** - logic for the calculating total value in inventory of a given item

Verifying Your Program

Your program must ***at least*** function correctly with the input provided in **`a1Input.txt`**. It is extremely important to note that this is ***NOT*** the only input your script will be tested with. Thus, think carefully of corner cases and work diligently to test your scripts with large scale input and debug them accordingly, if necessary.

1. Verify that your assignment folder has a **`data`** directory with the initial data set.
2. Execute your script and *redirect* **`a1Input.txt`** into it. You should **not** be copying or typing the contents of **`a1Input.txt`** into your terminal. File redirection **must** work.

   For example, the following must work: **`./assign1.bash < a1Input.txt`**
3. Verify that both the printed output and data files are as expected.

Submission

Turn your assignment in via Blackboard. Your zip file, named **`a1-a`*`bc123`*`.zip`**, should *only* contain your six `bash` files.