Assignment 2:
Invoice Scripts

Dr. William Kreahling

March 10, 2021

Objectives

Practice assigning values to shell variables, read values into shell variables, invoke shell scripts from within a shell script and check the exit status, use the AWK, read, grep, test, cut, basename, and wc Linux utilities, use command substitution, and redirection, piping.

Instructions

Your assignment is to write at least two shell scripts that are used to setup simple text invoice files.

1. create.sh
2. valid.sh

You may create additional shell scripts as needed to abstract out common operations. While you are allowed to invoke other shell scripts that you have written and the Linux utilities specified above, you are not allowed to invoke other executables (i.e. you cannot write your solution in a conventional programming language like C++ or Java). If you need to store information into a temporary file, then you can use the file *tmp.txt*. Be sure to delete the temp file before you exit your script.

1. **create.sh**: Accepts two command line arguments: a *file type*, either "-i" or "-o", followed by a *file name* (no extension). The script's purpose is to create a new file corresponding to the type specified. This script calls the `valid` script to make sure the file that was created is valid. This script will create files that contain header information only. In this assignment we will not actually populate our invoice with all the data.

   The header for a invoice consists of the following:

   (a) The first line contains the word "customer" immediately followed by a colon, followed by n number of field names that will comprise a customer's name.

   (b) The next second line contains the word "address" immediately followed by a colon followed by an address in the form NNN STREET NAME, CITY, ST. States must be 2 characters long, NC is the only valid state of in-state invoices and NC is not legal for the state field in out-of-state invoices.

   (c) The third line contains the word 'categories' immediately followed by a colon followed by n number of item categories from which a person wants items.

   (d) The fourth line is the word 'items' immediately followed by a colon followed by a positive number indicating the amount of items in each category that a person wishes to buy. The number of items and the number of categories must match.

   (e) There may or may not be empty lines between each of the four header lines.

   Users may not use the key words that start each line for a category name, (e.g., `category:produce items` is an illegal line.

On success your script will create a file name with the extension '.iso' for in-state invoices, and '.oso' for out of state invoices.

When a user is creating an invoice, the script will prompt the user, line by line to enter a name, street address, city, and state (users are not prompted for a state for in-state invoices), categories, and item numbers.

- Print a usage message and exit if the incorrect number of command line arguments is passed.
- Your script should check for invalid command line arguments.
- You can not create a file with a specific name if another file of that name already exists.
- This script must call valid.sh
- If the file was deemed *invalid* there should not be a file created when the script completes.
- Upon successful completion print out a success message and correct exit status and the first two lines of the newly created file.
- A user must enter at least 1 category name, if not prompt them again until they do.

Sample runs of the `create.sh` script:

```
>create.sh -o
usage: create.sh -i|-o filename


>create.sh -o order
Please enter customer name > Harry Potter
Please enter street address > 4 Privet Drive
Please enter city > Little Whinging
Please enter state > Surrey
ERROR: Surrey is not the correct length for a state abbreviation: (6)




> create.sh -i shopping
Please enter customer name > John Doe
Please enter street address > 4567 Maple Drive
Please enter city > Sylva
Please enter the fields that comprise the order > produce clothes candy toiletries
Please enter the number of "produce" items you want to purchase > 4
Please enter the number of "clothes" items you want to purchase > 6
Please enter the number of "candy" items you want to purchase > 2
Please enter the number of "toiletries" items you want to purchase > 4

"shopping.iso" has been created for
customer:John Doe
address:4567 Maple Drive, Sylva, NC



>create.sh -o shopping
Please enter customer name > Jane Doe
Please enter street address > 111 Now Street
Please enter city > Los Angeles
Please enter state > CA
Please enter the fields that comprise the order > Food Crystal
Please enter the number of "Food" items you want to purchase > 4
Please enter the number of "Crystal" items you want to purchase > 2
calling valid with shopping.oso

"shopping.oso" has been created for
customer:Jane Doe
```

```
address:111 Now Street, Los Angeles, CA

> create.sh -o shopping
ERROR: shopping.oso already exists
```

2. **valid.sh**: This script evaluates files for validity.

   This script accepts one command line arguments, the name of the file (with the extension). This script checks for the following errors in file composition:

   - correct permissions
   - correct file extensions
   - correct headers on the first four lines. Keep in mind blank lines may be present between each header line.
   - in state invoices have NC as he state.

   Below are some examples where *shopping.foom* does not exist. *broken.oso* is missing the item row. *broken2.oso* does not have the same number of items values as their are categories. *broken2.isi* exists, but is not a valid file extension.

```
> valid.sh shopping.foom
ERROR: shopping.foom is not accessible

> valid.sh broken.oso
ERROR: Missing header line
Last header line: categories:Food,Crystals


> valid.sh broken2.oso
ERROR: invalid item quantities: 2 categories but 1 items


> valid.sh broken2.isi
ERROR: ".isi" is not a valid file extension
```

Details

1. Where appropriate your scripts should call other scripts. Do not duplicate work performed by one script in another scripts.
2. All scripts should print usage messages if an incorrect number of command line arguments are supplied.
3. Command line arguments must be passed into your scripts in the order they are listed in this handout. Thus everyone's script will accept the arguments in the same order and *should* work with one another.
4. All scripts should exit and print error messages if the files we need to create/read/modify do not have correct permission to allow us the access needed. If a file does not have write permissions but we are not writing to it, that should **not** be reported or cause the script to fail.
5. Use correct style and indentation, comments, name, etc.
6. **Your output should match mine as closely as possible!**
7. You must name your scripts with the names supplied in this document.
8. You should use multiple error code in your script. You must list and describe error code semantics in your comments for each script.
9. The tests shown in this document do **not** cover all errors that could occur.

10. You must submit a text document named `tests.txt` with a list of tests that you performed on your scripts. A step by step list of commands executed and output produced is desirable.

11. **You are encouraged to work in teams of two**.

Submission

Submit your shell scripts by midnight, Wednesday March 24th using the handin program on agora (assignment number 2).