Assignment 1: Shell Scripting

CS3423 - Systems Programming — Rocky Slavin - UTSA

For this assignment, you will use **bash** create a simple tenant database for an apartment complex. The system will store basic information about apartments and their tenants and allow the user to create, read, update, and delete them.

This assignment requires only the utilities used so far in the lecture notes. **Do not** use sed, awk, or any programming languages or utilities not yet covered in the lecture notes.

Storing Apartment Information

Apartment information will be stored in text files.

1. Files will be stored inside a directory called `data` within the same directory as your script.
2. Each file will be named based on an apartment number, an integer with exactly **three** digits, followed by the extension `.apt` (notice that the text file does not end in `.txt`. Does that need to be accounted for?).
3. An apartment file consists of *exactly* three lines:
   - first_name (string with **no whitespace**) last_name (string with **possible whitespace***)
   - lease_start (string with **no whitespace**) lease_end (string with **no whitespace**)
   - balance (integer)

   * Last names may contain whitespace. You should account for names with multiple tokens (e.g., "Bob My Last Name Is Really Long" $\implies$ last_name = "My Last Name Is Really Long")
4. Example file named `323.apt`

```
Ralph Vaughan Williams
2/13/19 2/13/20
900
```

Script Execution

When the script is run, the following should occur. **All script output should appear exactly as it appears below.**

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 apartment record
R - read an existing apartment record
U - update an existing apartment record
D - delete an existing apartment record
P - record payment for an existing tenant
T - get total apartments
```

2. The user then enters a one-character action (upper or lowercase), leading to one of the following.
    - `C`: an apartment record is created
        (a) From the terminal, read the following one at a time
            i. Apartment number (three digit integer)
            ii. Full name (string possibly containing whitespace)
            iii. Lease start date (string with slashes)
            iv. Lease end date (string with slashes)
        (b) Using the values entered by the user, create a new file in the `data` folder based on the instructions above. **The starting balance should be 900.**
        (c) Update `data/queries.log` by adding the following line:
        `[date] CREATED: apt_number`
        where *date* is the output from the `date` command and *apt_number* is the corresponding apartment number.
        (d) If the apartment number already exists, print the following error and continue with the script. The script should accept all four inputs *before* checking if the record exists.
        `ERROR: apartment already exists`
    - `R`: read an existing apartment's information
        (a) Prompt the user for an apartment number:
        `Enter an apartment number:`
        (b) Search for the specified apartment using the apartment number.
        (c) Print the apartment information in the following format:
        `Apartment Number: apt_number`
        `Tenant Name: last_name, first_name`
        `Lease Start: lease_start`
        `Lease End: lease_end`
        `Current Balance: balance`
        (d) If the apartment is not found, print the following error instead and continue with the script.
        `ERROR: apartment not found`
    - `U`: update an existing apartment record
        (a) Prompt the user for the following one at a time
            i. Apartment number (three digit integer)
            ii. Full name (string possibly containing whitespace)
            iii. Lease start date (string with slashes)

iv. Lease end date (string with slashes)

v. Balance (integer)

(b) Search for the specified apartment using the apartment number.

(c) Update each of the corresponding fields based on the user input. **If the user input is blank for a particular field (except apartment number), keep the original value from the file.**

(d) Update `data/queries.log` by adding the following line:
`[date] UPDATED: apt_number`
where *date* is the output from the `date` command and *apt_number* is the corresponding apartment number.

(e) If the apartment record is not found, print the following error and continue with the script. The script should accept all five inputs *before* checking if the record exists.
`ERROR: apartment not found`

- `D`: delete an existing apartment record

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

(b) Delete the specified apartment's file.

(c) Update `data/queries.log` by adding the following line:
`[date] DELETED: apt_number`
where *date* is the output from the `date` command and *apt_number* is the corresponding apartment number.

(d) Print the following message to stdout with the apartment's number:
`apt_number was successfully deleted.`

(e) If the apartment is not found, print the following error instead and continue with the script.
`ERROR: apartment not found`

- `P`: record payment for an existing apartment record

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

(b) Prompt the user for a payment amount:
`Enter a payment amount:`

(c) Search for the specified apartment using the apartment number.

(d) Update the apartment record by subtracting the payment amount from the apartment's current balance. **Negative balances are allowed.**

(e) Update `data/queries.log` by adding the following line:
`[date] PAID: apt_number - AMOUNT: amount - NEW BALANCE: balance`
where *date* is the output from the `date` command, *apt_number* is the corresponding apartment number, *amount* is the amount paid, and *balance* is the updated balance.

(f) If the apartment record is not found, print the following error and continue with the script. The script should accept the payment amount *before* checking if the record exists.
`ERROR: apartment not found`

- `T`: print the total apartment records

  (a) Print the total number of `.apt` files within the `data` directory:
  `Total apartment records:  total`
  where *total* is the total `.apt` files.

- If an invalid character is entered, print the following error 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/rslavin/cs3423/Spring19/assign1`. Copy this to your own assignment's directory.

Script Files

Your program should consist of seven bash files with the following names (case sensitive):

- `assign1.bash` - the main file which is 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
- `payment.bash` - logic for the payment option
- `total.bash` - logic for the total option

Verifying Your Program

Your program must work with the input provided in `a1Input.txt`. To test it:

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. Redirection must work.
3. Verify that the output and files are as expected.

Submission

Turn your assignment in via Blackboard. Your zip file, named **`abc123.zip`** (with your personal abc123) should contain only your seven bash files.