Lab 2 - Bash Shell Script

Description

1. Online Judge: https://nasa.chummydns.com
2. Total: 100 points.

General Goals

- Learn how to write a shell script.
- File hash vaildation.
- Parse JSON and CSV files.
- Create multiple users at once.

Precautions

- You can submit multiple judge requests. However, OJ will cool down for several minutes after each judge.
- **BACKUP** or **SNAPSHOT** your server before judging **EVERY TIME**.
- Make sure everything is fine after reboot.

Tasks & Requirements

General

- All **help** and **warning messages** should be outputted to `stdout` and the **error message** should be outputted to `stderr`.
- The help message function:

```
usage() {
  echo -n -e "\nUsage: sahw2.sh {--sha256 hashes ... | --md5 hashes ...}
-i files ...\n\n--sha256: SHA256 hashes to validate input files.\n--md5:
MD5 hashes to validate input files.\n-i: Input files.\n"
}
```

- Shell script provide the following options:

```
$ ./sahw2.sh -h

Usage: sahw2.sh {--sha256 hashes ... | --md5 hashes ...} -i files ...

--sha256: SHA256 hashes to validate input files.
--md5: MD5 hashes to validate input files.
-i: Input files.
```

```
$
```

Provide `-h` option to show the **help message** with all available options and exit with zero status code.

- Invalid arguments should be rejected with a non-zero status code, with the **error message** (`"Error: Invalid arguments."`) and **help message**.

```
$ ./sahw2.sh --meow
Error: Invalid arguments.

Usage: sahw2.sh {--sha256 hashes ... | --md5 hashes ...} -i files ...

--sha256: SHA256 hashes to validate input files.
--md5: MD5 hashes to validate input files.
-i: Input files.
$
```

- The number of hash strings should match the number of files provided. If not, the shell script should be rejected with non-zero status code and the **error message** (`"Error: Invalid values."`).

```
$ ./sahw2.sh --md5 4a4be40c96ac6314e91d93f38043a634 -i data1.csv data2.csv
Error: Invalid values.
$
```

- If the shell script received two types of hashed value it should be rejected with non-zero status code and the **error message** (`"Error: Only one type of hash function is allowed."`).

```
$ ./sahw2.sh --md5 4a4be40c96ac6314e91d93f38043a634 --sha256
77af778b51abd4a3c51c5ddd97204a9c3ae614ebccb75a606c3b6865aed6744e -i
data1.csv data2.csv
Error: Only one type of hash function is allowed.
$
```

Hash Validation

- Provide a checksum validation to all files.
- If the (`"Error: Invalid checksum."`).

```
$ ./sahw2.sh --md5 00000000000000000000000000000000 -i data.csv
Error: Invalid checksum.
$
```

An example provided a wrong md5 checksum.

- If the script received multiple checksums. Return and exit with the previous error message if one of these hashes is invalid.

Parsing JSON & CSV

- Script will automatically recognize whether the files are JSON or CSV.
    - **DO NOT use the filename extension.**
- JSON input files specification:

```
[
  {
    "username": "admin_cat",
    "password": "cat001",
    "shell": "/bin/sh",
    "groups": ["wheel", "operator"]
  },
  {
    "username": "meow_2",
    "password": "cat002",
    "shell": "/bin/tcsh",
    "groups": []
  },
  {
    "username":"meow_3",
    "password":"cat003",
    "shell":"/bin/csh",
    "groups": []
  }
]
```

- CSV input files specification:

```
username,password,shell,groups
admin_cat,cat001,/bin/sh, wheel operator
meow_2,cat002,/bin/tcsh,
meow_3,cat003,/bin/csh,
```

- Print out all username in the files with the format: "**`This script will create the following user(s): ... Do you want to continue? [y/n]:`**".

```
$ ./sahw2.sh --md5 cfb536a88d21475557afc0adc19e7db4 -i data.csv
This script will create the following user(s): admin_cat meow_2 meow_3 Do
you want to continue? [y/n]:
```

- If the user presses "**n**" or **Enter**, the script will exit with zero status code.
- Invalid file format or contents should be rejected with a non-zero status code and the **error message** (`"Error: Invalid file format."`).

```
$ cat data.csv
<!DOCTYPE html><html><body><p>My First Heading</p></body></html>

$ ./sahw2.sh --md5 52d4bbbb68c2c3dc5601dcbe49153bbc -i data.csv
Error: Invalid file format.
$
```

In this example providing a HTML file should be rejected.

Create Users

- Using the information provided by the files to create **users and groups**.
    - e.g. CSV content: **`saadmin, doyouwanttobuildasnowman, /bin/sh, wheel operator`**
      Your shell script should create a user called `saadmin` with the shell /bin/sh and the user should be within groups `wheel` and `operator` (If groups do not exist please create it) and the password of this user is `doyouwanttobuildasnowman`.

```
$ id saadmin
uid=<uid>(saadmin) gid=<gid>(saadmin)
groups=<gid>(saadmin),<gid>(wheel),<gid>(operator)
$ getent passwd saadmin
saadmin:x:<uid>:<gid>:saadmin:/home/saadmin:/bin/sh
$
```

- Skip users if they are already created or exist in the OS and print the **warning message** (`"Warning: user root already exists."`).

```
$ cat data.csv
username,password,shell,groups
root,toor,/bin/tcsh,wheel operator
$ ./sahw2.sh --md5 539c5df87099968c0b1b27cc3f5807fd -i data.csv
Warning: user root already exists.
$
```

Limitation

- **The following are prohibited.**
    - Use any interpreters, compilers or programming languages (e.g. Python, NodeJS, C/C++ etc.).
    - Use any networking tools (e.g. `wget curl` etc.), or connect to the internet.
    - Use Obfuscator to obfuscate your shell script.
- All system builtin tools are available.
- Only `/usr/local/bin/bash` shell is allowed, the script should start with the shebang `#!/usr/local/bin/bash`.
- We will not test the JSON/CSV file with empty fields in the judge.

Available Packages

- All Linux built-in tools
- `bash`
- `jq`
- `cut`
- `awk`

Grading

Tasks	Dependencies	Score
General (40%)		
Provide `-h` option to show the help message.		14
Invalid arguments should be rejected with a non-zero status code, with the error message and help message.		14
The number of hash strings should match the number of files provided.		6
Shell script received two types of hashed value should be rejected with non-zero status code and the error message.		6
Hash Validation (15%)		
checksum doesn't match the input hash, return with a non-zero status code and the error message. (single file)		10
checksum doesn't match the input hashes, return with a non-zero status code and the error message. (multi file)		5
Parsing Files (25%)		
Print out all username in the files.		10
Press "n" or Enter, the script will exit with zero status code.		10
Invalid file format should be rejected with a non-zero status code and the error message.		5
Create Users (20%)		
Skip users if they are already created or exist in the OS and print the warning message.		10
Users' information is correct.		5
Users can login.		5
Total		**100**

Hints

Example of `jq`

- Extract username in the JSON file.

```
$ cat data.json
[{"username":"admin_cat","password":"cat001","shell":"/bin/sh","groups":["
wheel","operator"]},{"username":"meow_2","password":"cat002","shell":"/bin
/tcsh","groups":[]},{"username":"meow_3","password":"cat003","shell":"/bin
/csh","groups":[]}]

$ cat data.json | jq -r ".[] | .username"
admin_cat
meow_2
meow_3
$
```

Using `jq` to extract usernames.

- The `join()` function in `jq` is useful for concat strings.

Hash Checksum

- Online tools helps you verify shell script's functionality
  - MD5 File Checksum Online (emn178.github.io)
  - SHA256 File Checksum Online (emn178.github.io)