# Homework 1

File Processing (Term I/2017–18)

*built on 2017/09/26 at 13:43:51* ***due:*** *Fri, Oct 6th @ 11:59pm*

**Be sure to read this problem set thoroughly, especially the sections related to collaboration and the hand-in procedure.**

## Collaboration

We interpret collaboration very liberally. You may work with other students. However, each student ***must*** write up and hand in his or her assignment separately. Let us repeat: You need to write your own code. You must not look at or copy someone else's code. You need to write up answers to written problems individually. The fact that you can recreate the solution from memory will be taken as proof that you actually understood it, and you may actually be interviewed about your answers.

*Be sure to indicate who you have worked with (refer to the hand-in instructions).*

## Hand-in Instructions

To submit this assignment, please follow the steps below:

1. Make sure your scripts run and work correctly on Hamachi. We will run the grading script on the server.
2. Zip up all the scripts and name it `a1.zip`

```
> zip a1.zip cpuinfo.sh backup.sh restore.sh happy_countries.sh awesome.sh git.txt
```

3. Find out the MD5 hash of your zip file. You will need to submit this code on Canvas. We use it for keeping track of your submission time. You may resubmit your work but the MD5 hash has to match.

```
> md5sum a1.zip
```

4. Copy the zip file to the directory `/subm/u5712345` where `u5712345` is your student ID.

```
> cp a1.zip /subm/u5712345
```

5. Log on to Canvas, go to assignment 1 submission page and enter the MD5 hash.

## Task 1: CPU Info (10 points)

Write a bash script called `cpuinfo.sh` that lists out the CPU information on the current machine.

**Expected output**

```
> ./cpuinfo.sh
 Intel(R) Core(TM)2 Quad CPU    Q9550  @ 2.83GHz
 Intel(R) Core(TM)2 Quad CPU    Q9550  @ 2.83GHz
 Intel(R) Core(TM)2 Quad CPU    Q9550  @ 2.83GHz
 Intel(R) Core(TM)2 Quad CPU    Q9550  @ 2.83GHz
```

The number of lines in the output must match with the number of CPU cores on the machine.

**Hints**

1. You can find the CPU details in the system file `/proc/cpuinfo`.
2. Look for `model name` in the file.
3. Use `cut` to format the output.

## Task 2: Backup (10 points)

We all agree that backing up data is important but it is often negected. In this task, you will write a script called `backup.sh` that will take an argument, a path to perform backup.

Let's assume you want to backup a directory call `my_work` located under your home directory i.e. `~/my_work`. Running the backup script:

```
> ./backup.sh ~/my_work
```

should perform the following tasks in order:

1. Create a directory:

   `/subm/u12345/backups/[DIR NAME]_YYYY-MM-DD_HH:MM:SS`

   where `YYYY-MM-DD_HH:MM:SS` is the current timestamp. For example, if you run the script at exactly 2pm on Jan, 6 2017, then the backup directory should be:

   `/subm/u12345/backups/my_work_2017-01-06_14:00:00`

2. Copy (recursively) all files and directories in `~/my_work` to the backup directory you just created in the previous step.

3. Print to the terminal:

   ```
   Backup ~/my_work completed successfully.
   ```

**Hint:** You might want to check out `date` command.

## Task 3: Restore (10 points)

Backups won't be very useful if we cannot recover the files. Since you already have a backup script, now write a script called `restore.sh` that takes in directory name e.g. `my_work` for restoring your backups.

Running the script as shown below:

```
> ./restore.sh my_work
```

should perform the following tasks in order:

1. Create (if not already) a directory called `recovered` in the current directory.
2. Remove (if exist) `./recovered/my_work`.
3. Copy over `my_work` from our backup repository `/subm/u12345/backups` to `./recovered/my_work`. If multiple backups exist, choose the newest one. (Hint: `man sort`)
4. Print to the terminal:

   ```
   my_work has been restored to ./recovered/my_work.
   ```

## Task 4: Happy Countries (10 points)

Write a bash script called `happy_countries.sh` that lists out the names of the countries reported by 2017 World Happiness Report ranked by the happiness.

Your script will retrieve the information directly from the following wikipedia page, https://en.wikipedia.org/wiki/World_Happiness_Report. However, parsing the data directly from HTML is a headache. Luckily, you can request the page in an alternative format (raw wiki format) by using (https://en.wikipedia.org/wiki/World_Happiness_Report?action=raw). Notice the suffix `?action=raw`.

Below is expected output of your script.

```
Norway
Denmark
Iceland
Switzerland
Finland
Netherlands
...
Togo
Rwanda
Syria
Tanzania
Burundi
Central African Republic
```

There are total of 155 countries in the list. For this task, you may only use the following tools: `curl`, `sed`, `awk`, `grep`, `tr`, `cut`, `sort`, `head`, `tail`.

## Task 5: Your Awesome Scripts (10 points)

Design and write a shell script called, `awesome.sh` to complete ONE of the following tasks below:

- List files in a given directory whose size is larger than $K$ bytes
- Find all sub-directories containing more than $K$ files
- Other script of your choice. However, the script should take at least 2 command-line arguments and check with me first if it is okay.

Your awesome script MUST give an error message when the number of arguments is wrong and support `--help` option to print out help message.

## Task 6: Git (10 points)

Now you will learn to use git repository on `github.com` to open source your awesome shell script (from previous task). Follow the steps below to complete this task. Do not skip steps.

1. Obtain a account on `github.com`
2. Create a public repository for your script.
3. Create a commit that adds your script to the repository.
4. In another commit, add `README.md` to describe what your script does and how to use. Read about how to use Markdown here: `https://guides.github.com/features/mastering-markdown/`
5. In another commit, add `LICENSE` to include the license information of your repository. Since your script is original, you can use the following license template.

```
MIT License

Copyright (c) 2017 <YOUR NAME>

Permission is hereby granted, free of charge, to any person obtaining
a copy of this software and associated documentation files (the "Software"),
to deal in the Software without restriction, including without limitation the
rights to use, copy, modify, merge, publish, distribute, sublicense, and/or
sell copies of the Software, and to permit persons to whom the Software is
furnished to do so, subject to the following conditions:

The above copyright notice and this permission notice shall be included in all
copies or substantial portions of the Software.

THE SOFTWARE IS PROVIDED "AS IS", WITHOUT WARRANTY OF ANY KIND, EXPRESS OR
IMPLIED, INCLUDING BUT NOT LIMITED TO THE WARRANTIES OF MERCHANTABILITY,
FITNESS FOR A PARTICULAR PURPOSE AND NONINFRINGEMENT. IN NO EVENT SHALL THE
AUTHORS OR COPYRIGHT HOLDERS BE LIABLE FOR ANY CLAIM, DAMAGES OR OTHER LIABILITY,
WHETHER IN AN ACTION OF CONTRACT, TORT OR OTHERWISE, ARISING FROM, OUT OF OR IN
CONNECTION WITH THE SOFTWARE OR THE USE OR OTHER DEALINGS IN THE SOFTWARE.
```

You can read more about different licenses here:
`https://opensource.guide/legal/#which-open-source-license-is-appropriate-for-my-project`.

6. Create a remote branch called `filepro-2017`. In this branch, add an arbitary file to your repository.

7. Create a tag "1.0" on the master branch and push it github. Read more about tagging here: `https://git-scm.com/book/en/v2/Git-Basics-Tagging`

To summarize, in your github repo, you must have at least **3 commits, 2 branches (master and filepro-2017) and 1 tag ("1.0")**.

For this task, you will have to submit a file called `git.txt` that contains the url to your repository on Github.