# HW4 (100 POINTS)

***Before you start:***

1. Download the zip file HW4.zip. This file will usually be saved in the Downloads folder (directory).
2. Move HW4.zip to your home directory by using GUI-drag and drop.
3. Open the zip file. A folder named HW4 will be created in your home directory.
4. Open a terminal session.
5. Change into your HW4 directory.
6. Use vi to make and edit files in directory HW4.
7. Open a second terminal by pressing **CMD+n** and use this terminal to test the Unix code that you write as answers to the questions.

**This HW has 3 parts, HW4-part1, HW4-part2 and HW4-part3, and will have 3 separate submissions on Gradescope.**
**All these parts are due Sunday Feb 18 11:59am**
**If you submit:**

- **HW4-part1 by Wed Feb 14 (11:59pm) you will get 3 extra points**
- **HW4-part2 by Fri Feb 16 (11:59pm) you will get 3 extra points**
- **HW4-part3 by Sat Feb 17 (11:59pm) you will get 2 extra points**

Ubuntu and WSL users please include in a comment line that you use Cygwin or Linux/Ubuntu in each script you submit - Write it down below the class header, like this:

```
#class header
#Ubuntu
```

# HW4-part1 (36)

**Follow these instructions for this HW part**
**In each script, include only the class header – no Q&A format.**

**Example:**
**#mprocop2:02/15/2024:filename**

**Syntax errors in a script will result in a 0 for the entire script.**
**If you're unsure how to fix errors, comment out your code for partial credit.**

*General Grading: Give all points for correct script*
*0.5 for class header*
*All points off for shell error messages*
*Partial points if you comment out a line with errors*

**1 (11)** Start this exercise from the HW4 directory.

Write a bash script called **Ex4-hidden.bash.** The script must be made within the HW4 directory. **DO NOT USE absolute paths in this script.**

- The script should contain **one for loop** that does the following:
  **loops** over all the hidden files ending with .bash that are within the scripts directory **runs** each of those files (which are bash scripts), **redirects** stderror to the null device, and **appends** the stdout to a file called *stdout.txt*.

  Do not hardcode the hidden files in the for loop but use metacharacters.

  The content of the *stdout.txt* should be the following: but the order of the lines could be different.

```
this is the bob file
this is the pop1 file
this is the program script
this is script1
this is script2
```

- (0.5) Add one line of code, so that if you run the script multiple times, the content of the *stdout.txt* will not change. Be careful where you put this line.

**2** (11) Make a script called **Ex4-cities.bash.** The script must be made within the HW4 directory. **DO NOT USE absolute paths in this script.** In the script do the following:

- (2) Create a directory called *cities* in your current directory.
- (8) Use **only one for loop,** that does the following:
    - Create the following directories within directory *cities:*
      *NY*
      *DC*
      *LA*
      *SF*
    - In each directory (NY, DC, LA, SF) create a file, i.e,
      the file within directory *NY* should be called *file_NY.txt*
      the file within directory *DC* should be called *file_DC.txt* and so on.
    - In each file write the city abbreviation in lowercase letters, i.e., in *file_NY.txt* write *ny* in the *file_DC.txt* write *dc* and so on.
- (0.5) After the for loop, view the content of one of the generated files. Choose a file you like (any file in one of the cities subdirectory is okay).

**3** (14) Make a script called **Ex4-letters.bash.** The script must be made within the HW4 directory.

The script should contain a for loop, which iterates over letters provided from the command line. The input arguments should be a list of letters separated by one space.

For example:

```
a c d e g
```

Inside the for loop, each letter should be appended to a file named letters.txt. In addition, within the for loop a count of the number of iterations (or letters) should be performed.

Do not use the wc command to count.

After the loop has finished, print to screen **the total number of letters** contained in file letters.txt such as:

```
The file letters.txt contains 5 letters.
```

Do not hardcode the number, but use the count variable resulting from the count, which should contain the total number of letters.

At the end of your script, write in a comment lien the bash code you would use to run the script.

**Upload the following file to Gradescope HW4-part1:**
**Ex4-hidden.bash**
**Ex4-cities.bash**
**Ex4-letters.bash**

# HW4-part2 (37)

**Follow these instructions for this HW part**

- **Include only the class header – no Q&A format.**

**Example:**
**#mprocop2:02/15/2024:filename**

**Syntax errors in a script will result in a 0 for the entire script. If you're unsure how to fix the error, comment out your code for partial credits.**

*General Grading: Give all points for correct script*
*0.5 for class header*
*All points off for shell error messages*
*Partial points if you comment out a line with errors*

**1.** (14) Make a script called **Ex4-guess-music.bash**. The script will take arguments from the command line.

- When you run the script, the user will provide the name of a music genre (i.e., jazz, pop, classical, trap, drill, etc.) from the command line. Store the music genre entered by the user in a variable called *guess*
- Store a music genre that you like in a variable called *music1*
- Store a music genre you do not like in a variable called *music2*

- **Use an if-elif-else statement** to implement the following:
  if the music entered by the user (and stored in variable *guess*) matches your favorite music stored in variable *music1*, print to screen:

  ```
  You got my favorite music genre
  ```

  if the music genre entered by the user (and stored in variable *guess*) matches the music genre you do not like, stored in variable *music2*, print to screen:

  ```
  You got my not favorite music genre
  ```

  otherwise, print to screen these two sentences in separate lines:

  ```
  Sorry, you did not pick my music genres.
  Run the script again and enter a different music genre.
  ```

- In a comment line, write the code the user should use to run the script.

**2.** (10) Make a script **Ex4-numbers.bash**. The script should generate 2 files: *even.out* and *odd.out*.

The file *even.out* should contain even numbers and *odd.out* odd numbers. Each number should be on a separate line.

Make one for loop over this sequence of numbers 1,2,3,4,5,6, …, 40. Use metacharacters to generate the sequence of numbers. Inside the for loop use an **if-else statement** to generate those two files.

Run the script and make sure it generates the correct files.

**3.** (13) Make a script called **Ex4-mean-while.bash**, and in it make a while loop to calculate the average of numbers from 1 to 50.
To perform this calculation, within the while loop you should sum all integer numbers from 1 to 50 and count the number of iterations. After the loop has finished, calculate the average by dividing the total sum by the total count. The script should print to screen ONLY the results by using the two variables that store respectively the final sum and the final count:

```
The sum is 1275, the count is 50, and the average is 25
```

If you use a for loop, you will lose 8 points.

**Upload the following file to Gradescope HW4-part2:**
**Ex4-guess-music.bash**
**Ex4-numbers.bash**
**Ex4-mean-while.bash**

# HW4-part3 (27)

**Follow usual instruction**

- **Include the class header**
- **Format of Q&A pairs**

**Example:**

```
#mprocop2:02/15/2024:filename

#Q1
cat file

#Q2
ls -lt file
```

*General Grading: Give all points for correct script*
*0.5 for class header*
*All points off for shell error messages*
*Partial points if you comment out a line with errors*

Make a script called **Ex4-symbols.bash.** The script must be made within the HW4 directory. In this script you will use the provided file **dow_jones_2011.dat**, which contains weekly stock data for Dow Jones Index during 2011.
The file contains the following fields:

1st field: yearly quarter (1 for Jan-Mar; 2 for Apr-Jun).
2nd field: the stock symbol (below is a list of some stock symbols).

| **company** | **stock symbol** |
|---|---|
| Bank of America | BAC |
| Intel | INTC |
| IBM | IBM |
| Johnson & Johnson | JNJ |
| Pfizer | PFE |

3rd field: date, i.e., the last business day of the work week (usually Friday)
4th field: the price of the stock at the beginning of the week
5th field: the highest price of the stock during the week
6th field: the lowest price of the stock during the week
7th field: the price of the stock at the end of the week

Q1. (5) Make one variable called *symbols* and store in it the following string of symbols:

```
BAC HD IBM INTC JNJ MSFT PFE WMT XOM
```

These are stock symbols separated by one space.

Q2. (7) Use one for loop over the variable stock symbols by using variable **symbols**, and for each symbol generate a file. The repeated task of the for loop is to extract all the lines of **dow_jones_2011.dat** matching a stock symbol and save those lines into a file, i.e.,
all the lines of the dow_jones_2011.dat containing IBM should be stored in a file called *IBM.dat*, all the lines containing JNJ in a file called *JNJ.dat* and so on. Do not hardcode the symbols and use only one for loop. Do not use if statements.

Q3. (15) Use one for loop over the values of the stock symbols by using variable *symbols* and for each stock symbol use a pipeline of commands to do the following:

- extract the maximum price at the end of the week ($7^{th}$ field) and corresponding date ($3^{rd}$)
- print the stock symbol, the date and maximum value.

You can use the files generated in the previous question (in this case do not hardcode the files) or just use **dow_jones_2011.dat.**

Format the output to obtain:

```
BAC    1/14/2011   15.2
HD     2/18/2011   38.5
IBM    4/29/2011  170.6
INTC   5/13/2011   23.4
JNJ    5/27/2011   66.8
MSFT    1/7/2011   28.6
PFE    4/29/2011   21.0
WMT    1/28/2011   56.7
XOM    4/29/2011   88.0
```

Set the width to align the fields. The number of spaces between fields should be at least two, and the fields should be aligned as reported above.
Do not hardcode the stock symbols and use only one for loop. Do not use if statements.

**Upload the following file to Gradescope HW4-part3:**
**Ex4-symbols.bash**