CS 218 – Assignment #11, Part B

Purpose: Become more familiar with input/output buffering concepts.
Points: 80 Code (including header/comments) – 40%
Time Script Output – 10%
Write-up – 50%

Assignment:

Use the make[1] utility and the provided ***makefile*** which will create a second version of the **`a11procs.asm`** file with by changing the BUFFSIZE constant from 500,000 to 2 (as seen in the newly created file **`a11procsSM.asm`**). Then, the make will build two executable files; ***benfordLG*** (using large buffer) and ***benfordSM*** (using small buffer).

We will use the Unix **`time`**[2] command to obtain the execution times. Additionally, we will use the Unix **`diff`**[3] command which compares two files and reports any differences.

To simplify this process, a script file is provided to execute the program three (3) times with each executable file. The timing results are placed in a file '**a11Btimes.txt**'.

Source: xkcd.com/612

- In the provided template, **`write_up.txt`**, summarize your results for assignment #11 A and B. Include the following information.
  - Briefly describe your machine (one sentence). Include the machine type (desktop/laptop/mini), processor speed, disk type (ssd, hard-drive, etc.) and memory.
  - Compute the average '***real***' time for the three 'large' buffer size executions (original). Ensure to leave the original two results and add the final averaged result.
  - Compute the average '***real***' time for the three 'small' buffer size executions. Ensure to leave the original two results and add the final averaged result.
  - State which was faster and by how much. Include the time difference and the percentage faster or slower. The percentage change[4] should be calculated as follows:

$$percentChange = \left(\frac{(small\ buffer\ average) - (large\ buffer\ average)}{(large\ buffer\ average)} \right) * 100$$

  - Explain the results. Specifically, explain the impact of the buffer size on the execution speed of the program and address what specifically caused the difference. Explanation should not exceed 200 words.

Note, any explanations exceeding 200 words will not be graded and scored as a 0.

1 For more information, refer to: https://en.wikipedia.org/wiki/Make
2 For more information, refer to: http://en.wikipedia.org/wiki/Time_(Unix)
3 For more information, refer to: http://en.wikipedia.org/wiki/Diff
4 For more information, refer to: http://en.wikipedia.org/wiki/Percent_change

Submission:

- All source files must assemble and execute on Ubuntu and assemble with **`yasm`**.
- Submission three (3) files
  - Submit source file
    - *Note*, only the functions file (**`a11procs.asm`**) will be submitted.
  - Submit only the **`a11Btimes.txt`** file (the system will change the buffer size from 500000)
    - This is the output of the provided timing script (no changes are needed).
  - Submit Write Up file
    - Includes system description, percentage change, and results explanation (per above).
- Once you submit, the system will score the code part of the project.
  - If the code does not work, you can (and should) correct and resubmit.
  - You can re-submit an unlimited number of times before the due date/time (at a maximum rate of 5 submissions per hour).
- Late submissions will be accepted for a period of 24 hours after the due date/time for any given lab. Late submissions will be subject to a ~2% reduction in points per an hour late. If you submit 1 minute - 1 hour late -2%, 1-2 hours late -4%, … , 23-24 hours late -50%. This means after 24 hours late submissions will receive an automatic 0.

Program Header Block

All source files must include your name, section number, assignment, NSHE number, and program description. The required format is as follows:

```
;  Name: <your name>
;  NSHE ID: <your id>
;  Section: <section>
;  Assignment: <assignment number>
;  Description: <short description of program goes here>
```

Failure to include your name in this format will result in a loss of up to 3%.

Scoring Rubric

Scoring will include functionality, code quality, and documentation. Below is a summary of the scoring rubric for this assignment.

Criteria	Weight	Summary
Assemble	-	Failure to assemble will result in a score of 0.
Program Header	3%	Must include header block in the required format (see above).
General Comments	7%	Must include an appropriate level of program documentation.
Program Functionality (and on-time)	30%	Program must meet the functional requirements as outlined in the assignment. Must be submitted on time for full score.
Timing Script Output	10%	Output of timing script is correct.
Write-Up	50%	Write-up includes required section, percentage change is appropriate for the machine

		description, and the explanation is complete.

Assignment #11B Timing Script

The following commands will execute the program and provide timing results (for both the 'large' and 'small' buffers).

```
ed-vm% time ./benfordLG -i a11PartB.txt -o tmpLG.txt
ed-vm% diff tmpLG.bmp mstr.txt
ed-vm%
ed-vm% time ./benfordSM -i a11PartB.txt -o tmpSM.txt
ed-vm% diff tmpSM.bmp mstr.txt
ed-vm%
```

To automate this process, a script file is provided that will execute the assignment #11 three times for the 'large' buffer size and two times for the 'small' buffer size and place the results in a file. After you download the script file, **`a11Btimer`**, set the execute permission as follows:

```
ed-vm$ chmod +x a11Btimer
ed-vm$ ./a11Btimer
```

The script file will look for the executables **`benfordLG`** and **`benfordSM`** as created by the make utility.

You will need to perform the averaging using a calculator. Be careful of the minutes and seconds times when adding the values! It may be easiest to convert all times to seconds.

Unix Time Command

The Unix Time command will provide some details on how long a program or command took to execute. For example, if you have a program **`./someProg`** then in the shell you can type:

```
ed-vm$ time ./someProg
```

The output (shown below) details how long the code took to run:

```
real   1m10.951s
user    0m2.390s
sys     0m1.705s
```

- Real time - Elapsed time from beginning to end of program (or wall clock time)
- CPU time - Divided into User time and System time
  - User time - time used by the program itself and any library subroutines it calls
  - System time - time used by the system calls invoked by the program (directly or indirectly)

At the terminal prompt, you can type **`man time`** to see the manual page for time.