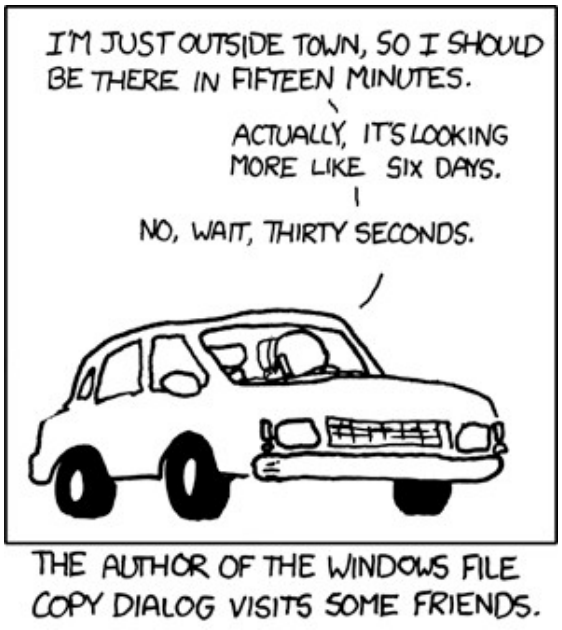
CS 218 – Assignment #11, Part B

Purpose: Become more familiar with input/output buffering concepts.
Points: 50 (Description 10 pts, Write-up 40 pts)

Assignment:

We will update the assignment #11 code to change the buffer size from 1,000,000 to 2. This update is done in the provided **makefile**. Execute the original program from assignment #11 A (buffer size of 1,000,000) and the modified assignment #11 B (buffer size of 2). We will use the Unix **`time`**[1] command to obtain the execution times. Additionally, we will use the Unix **`diff`**[2] command which compares two files and reports any differences.

To simplify this process, the provided **`makefile`** will read the original **`a11procs.asm`** file, edit the buffer size, and create two executables; once with the large buffer and one with the small buffer. You will need to copy the provided main from part A into the part B directory. *Do not alter the original buffer size definition line* (as the **makefile** will do it). A provided bash script will execute the program multiple times with each executable file. The timing results are placed in a file **`a11times.txt`**.



Source: xkcd.com/612

- Summarize your results for assignment #11 A and B. The write up (in **`write_up.txt`**) should 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 two '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 two '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[3] should be calcualted as follows:

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

  - Explaination of the results. Specifically, explain the impact of the buffer size on the execution speed of the program. Explaination should not exceed 200 words. (40 pts) *Note, any explainations exceeding 200 words will not be graded and scored as a 0.*

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

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 ./imageCvtLG -br coast1.bmp tmp1.bmp
ed-vm%
ed-vm% time ./imageCvtSM -br coast1.bmp tmp2.bmp
ed-vm%
```

To automate this process, a bash script is provided that will execute the assignment #11 multiple times for the 'large' buffer size, for the 'small' buffer size and place the results in a file. You can download the bash script, set the permission, and execute as follows:

```
ed-vm$ chmod +x a11timer
ed-vm$ ./a11timer imageCvtLG imageCvtSM
```

Where **`imageCvtLG`** is the assignment #11 A executable (with the large buffer) and **`imageCvtSM`** is the assignment #11 B executable (with the small buffer) as created by the provided makefile.

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.

Submission:

- All source files must assemble and execute on Ubuntu and assemble with **`yasm`**.
- Submit source files
  - Submit a copy of the program source file via the on-line submission.
  - Note, only the original, unmodified functions file (**`a11procs.asm`**) will be submitted.
  - Timer script output (**`a11btimes.txt`**).
  - Assignment #11B write up (**`write_up.txt`**).
- Once you submit, the system will score the project and provide feedback.
  - If you do not get full score, you can (and should) correct and resubmit.
  - You can re-submit an unlimited number of times before the due date/time.
- Late submissions will be accepted for a period of 24 hours after the due date/time for any given assignment. 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: <4-digit-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 10%.

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)	90%	Program must meet the functional requirements as outlined in the assignment. Must be submitted on time for full score.