Project - 2
CSE 4/521 – Introduction to Operating Systems
Due: August 12th @11:59pm, 2018

Objective:
Implement correct User Program Execution functionality in PintOS

Submission Deadline: There are 6 soft deadlines and one hard deadline for Project 2.

- The hard deadline is **August 12th, 2018**.
- Every team has a buffer time of 5 days to work around the hard deadline for Project 2 and Project 1 combined.
- If a team misses the soft deadline, that team would have to meet me in person.
- A total of 6 soft deadlines and 1 hard deadline exists for this project.
- Please note the 3 types of submissions: 1)in recitation, 2)in-class and 3)autograder.

Grading: (Your `make grade` score + 5% of your score for design doc) would be your final grade for Project 2.
Please note the academic integrity policy at: http://academicintegrity.buffalo.edu/policies

QUICK LINKS:
PintOS reference website – https://web.stanford.edu/class/cs140/projects/pintos/pintos_3.html
Autograder - https://autograder.cse.buffalo.edu/

Bird's eye description of User Program Execution in PintOS:
(Extensive description given in Stanford PintOS reference website)

In this project, you will enable user programs to interact with the kernel via system calls. You will be working on the `userprog` directory. No code from project 1 is required for this assignment.

The tests for Project 2 are very extensive. Make sure the user program interface meets the specifications implemented in the test cases. You are free to restructure/rewrite 'kernel code' however you wish.

Section 3.1.1 in PintOS reference website provides a description of all the source files that you may need to use/alter. In Project 2, you will need to just 'interface to' (not alter) the file system code (located in `filesys` directory) because user programs are loaded from the file system.

Use this document only as a reminded of the *design doc* and *code* deadlines and to pace yourself accordingly. Rather than copy-pasting content, I redirect all of you to the PintOS reference link now: https://web.stanford.edu/class/cs140/projects/pintos/pintos_3.html. Feel free to use any of their recommendations/code for UB's version of Project 2[1].

Be mindful that your test cases would begin to PASS only after sub-checkpoint 2 in implemented correctly.

[1] Hopefully we'll have our own version of 'Lloyd TacOS' someday.

Begin: Crux of the project (22th July, 2018)

Sub-checkpoint (1/2): Argument Passing

Design Doc Deadline: **27th July** *Code Deadline* (In recitation): **30th July**

Design Doc template – https://jerryajay.com/wp-content/uploads/2018/07/Project2-design-doc-1.txt

Currently, `process_execute()` does not support passing arguments to new processes. Implement this functionality, by extending `process_execute()` so that instead of simply taking a program file name as its argument, it divides it into words at spaces. The first word is the program name, the second word is the first argument, and so on. That is, `process_execute("grep foo bar")` should run `grep` passing two arguments `foo` and `bar`.

End point: The output of `hex_dump()` should display your parsed input correctly.

Sub-checkpoint (2/2): System Calls

Design Doc Deadline: **3rd Aug** *Code Deadline (Autograder)*: **12th Aug**

Design Doc template – https://jerryajay.com/wp-content/uploads/2018/07/Project2-design-doc-2.txt

Implement user program syscall functionality in PintOS. The recommended order of implementation is:

- *User memory access* (see section 3.1.5 Accessing User Memory). All system calls need to read user memory. Few system calls need to write to user memory. Code Deadline (In recitation): 3rd Aug
- *System call infrastructure* (see section 3.3.4 System Calls). Implement enough code to read the system call number from the user stack and dispatch to a handler based on it.
  Code Deadline (In recitation): 8th Aug
- *The* `exit` *system call*. Every user program that finishes in the normal way calls exit. Even a program that returns from main() calls exit indirectly (see _start() in lib/user/entry.c).
- *The* `write` *system call* for writing to fd 1, the system console. All of our test programs write to the console (the user process version of printf() is implemented this way), so they will all malfunction until write is available.
- *Change* `process_wait()` *to an infinite loop* (one that waits forever). The provided implementation returns immediately, so PintOS will power off before any processes actually get to run. 'exit + write+ process_wait' Code Deadline (In recitation): 10th Aug

You get 2 additional days (till Aug. 12th) to make all your tests pass after you implement the above functionalities. If you implemented the `syscall` functionality right, the make check script would give you the 30 points (as noted in the point distribution below) 'for free' without having to do any coding for the filesys/base.

End point: All syscall tests passed. ***(108+88+1+30=227 points)***

End: Crux of the Project (12th August, 2018)

Appendix:

A. Submission Procedure

Navigate to your pintOS repository folder (i.e. ~/pintos/pa2-foobar/)

1. Run the following commands:
```
cd src; make clean; cd ..; tar -czf pa2.tar.gz src/
```
2. This will clean your src directory, and provide you with an archive file that you will need to submit.
3. Open your browser, and navigate to: https://autograder.cse.buffalo.edu/
4. Sign in using your UB credentials.
5. In course page and choose Project 2.
6. Hit SUBMIT and upload pa2.tar.gz file that you created. Once a member submits, the submission and score will be applied to all group members automatically.
7. Once the job is processed (might take some time), your submission and score will be shown in Project 2 page. When you click on the score, it will show you the full result from the job.

B. Credits

Prof. Tevfik Kosar (University at Buffalo, NY)
Mr. Farshad Ghanei (University at Buffalo, NY)