Programming Project #1
CS4352 Operating Systems

Due Date: 3/2, 11:59 p.m., via Blackboard.
Late submissions are accepted till 3/9, 11:59 p.m., with 10% penalty each day.

In this programming project, you are asked to try shell, shell programming, and system programming (in C programming language) on Linux platform. This assignment consists of two parts. This assignment is designed to assist you for better understanding of operating systems including Unix/Linux shell and system programming, as well as enhancing programming skills and experiencing programming on a Unix-like environment.

Part 1. Shell Experiences and Programming

You are asked to try Unix/Linux shell in this part of project.

Requirements:

1. Run the *displayinfo.sh* and *greeting.sh* scripts provided and report the result (i.e. using shell redirection to produce output files and then submit output files).
2. Made two modifications to the *greeting.sh* script provided: 1) the script only shows a greeting message when it's morning, afternoon, or evening; please make a code modification to show a greeting message "Hello [your login name]!" when it's other than morning, afternoon, or evening; 2) please output the greeting message to a file called "greetingmsg", and if this file already exists, then deletes the file first, then output the message to this file.
3. Run the modified *greeting.sh* script on the Oak server and report the result.

Part 2. System Programming and Process Management (in C programming language)

You are asked to develop basic system programming including process creation and termination on a Linux platform in this part of programming project. You are asked to create multiple children processes to work under one parent process. In theory, children processes can do their own work separately or cooperatively to accomplish a task. In this assignment, these children processes simply print out a "hello" message together with their PIDs (process IDs) and exit. You should use *fork()* and *wait()* system calls to implement this program.

Requirements:

1. Take the number of children processes as an argument when the parent process creates children processes. This argument should be passed through a command line argument.
2. The parent process creates children processes and should print out an error message if creation fails. The parent process should also wait for all children processes to finish and then exit.
3. Each child process should print out a hello message together its PID and then exit.

4. Test with 2, 4, and 8 children processes.
5. Next, instead of creating multiple children processes of a parent process, you are asked to create a chain of processes, i.e., the parent process will create one child and wait for it to finish, while the child creates its own child and wait for it as well, and so on. The last child created should print out the message and exit immediately so that its ancestors can finish too. Test your program with 2, 4, and 8 children processes again.
6. Develop a Makefile, similar to the one we discuss in lecture to automate the compilation process.
7. Develop a shell script to automate the test process, i.e., to test with 2, 4, and 8 children processes for both versions of program automatically with this script.

Hints:

- System calls *getpid()* and *getppid()* return a process ID and the parent process ID, respectively.

Expected Submission:

You should submit a single tarball/zipped file through the Blackboard containing the following, and please name your submission file starting with LastName_FirstName_Project#1.

- Source codes for part 1 and part 2.
- Output files for the result of test cases for all parts.

How to report your results:

- You can simply use bash redirection ('>') to produce log files like what we showed in class, and then submit the log files (plain text files)

How to submit your codes and results

- First, on the Oak server, you can create a tar ball to include all your source codes, logs, and other files, like one of below commands
    - `tar cvf mycs4352project.tar *`
    - `tar cvfz mycs4352project.tgz *`
- Then you can use "scp" or any SFTP client, e.g., FileZilla to download the tar/zipped file to your local machine, after that you can submit to Blackboard just like you submit your other homework solutions
    - E.g., you can download "pscp.exe" from the Putty website, then run a command like below to copy/download "mycs4352.tar" file to your local machine (you'll need to replace the path name "home/TTU/yonchen/cs4352/public_gitlab_repo/mycs4352.tar" with your own file name
    - `pscp.exe yonchen@oak.cs.ttu.edu:~/cs4352/public_gitlab_repo/mycs4352.tar .`
    - `pscp.exe yonchen@oak.cs.ttu.edu:/home/TTU/yonchen/cs4352/public_gitlab_repo/mycs4352.tar .`

Grading Criteria:

Please note that, if needed, we may request an in-person, 5-10 mins quick demo from you.

Part 1	**Percentage %**	**Criteria**
40%	50%	Correct shell programming modification, modified script can be run
	50%	Correctness of result
Part 2	**Percentage %**	**Criteria**
60%	10%	Inline comments to briefly describe your code
	30%	Correct use of system calls/library procedures of creating processes, check process ID
	20%	Correctness of result. Source code can be compiled and executed.
	20%	Successfully carry out specified test cases
	10%	Correctness and features of Makefile (minimum features of automating compilation and cleaning up)
	10%	Correct shell script to automate tests

Source Code Samples

Please checkout source code samples with executing the following command on the Oak server:

```
git clone https://github.com/githubyongchen/OS.git
```

Reference Materials:

- Linux Shell scripting:
  http://www.freeos.com/guides/lsst/

- Linux system programming:

  Book: *Linux System Programming*

  Online:
  Tutorial for Beginners, http://www.ee.surrey.ac.uk/Teaching/Unix/
  Advanced Linux Programming, http://www.advancedlinuxprogramming.com/alp-folder/advanced-linux-programming.pdf

  Stack Overflow, http://stackoverflow.com/
  www.unix.com
  The Geek Stuff, http://www.thegeekstuff.com/

- Makefile**:**

- http://www.gnu.org/software/make/manual/make.pdf or
- http://www.gnu.org/software/make/manual/html_node/index.html