CS 273 (Operating Systems), Fall 2020, R. Brown

Homework 2 Due Monday, 8-31-20

A. Directory and file operations

Create a directory for your work in this homework assignment. Then make that your current working directory, so your work on this assignment will be created in that directory.

```
% mkdir ~/OS/hw2
% cd OS/hw2
```

B. C programming

1. Write a program `prtenv.c` in C that prints all strings in the environment, one per line. Use the third argument `envp` of `main()`, which is an array of strings, i.e., an array of pointers to `char`. (See the `reverse.c` example of Lab 1 for an example of a third argument `envp` in the header of `main()`.)

   You can use array indexing `envp[i]` to access the individual environment strings; or, you can optionally use pointers to access this array. Note that *the last environment string in the array* `envp[ ]` *is followed by a null pointer*, i.e., the value 0 (cast as a pointer to `void`, which is the generic pointer type in C).

2. Create a git `commit` containing your work on this segment.

   ```
   % git add prtenv.c
   % git commit -m "HW2 B complete:  prtenv.c"
   ```

   Note. If your work on this segment is not yet complete, indicate the status of your work so far in the `commit` message. As you complete more of this work, create additional `commit`s, using the `commit` messages to indicate your progress.

C. Programming with UNIX system calls

1. Make a copy of the program `~rab/os/egs/mopen.c` in your own directory, then compile and run that program to get familiar with its operation. As demonstrated in class, the program allows you to open a file multiple times, and use the file descriptors returned from those open calls to read and write to that file.

2. Modify the program `mopen.c` to recognize a command string `"dup"` that tests the system call `dup()`. This involves modifying the final "while" loop portion of `main()` by adding a case for `"dup"`, and modifying the prompts near the beginning of that loop. Report the return value from `dup`, so a user can use that returned file descriptor to request in `read` or `write` calls, as a test.

   Be sure to check for an error return from the system call `dup()`. You can test this feature by attempting to `dup()` an inactive file descriptor.

3. Create a git `commit` containing your work on this segment.

   ```
   % git add mopen.c
   % git commit -m "HW2 C complete:  mopen.c"
   ```

   Note. If your work on this segment is not yet complete, indicate the status of your work so far in the `commit` message. As you complete more of this work, create additional `commit`s, using the `commit` messages to indicate your progress.

D. **C programming**

Complete and submit Lab 2 (`https://www.stolaf.edu/people/rab/os/pub/lab2.pdf`).

E. **Linux system calls**

1. Describe the use of Linux system calls for the following shell input lines, using process diagrams as discussed in class.

   a) `ls -als`

   b) `ls -als &`

   To submit the diagrams in this part, follow the instructions in this google doc:
   `https://www.stolaf.edu/people/rab/os/asgt/hw2e.html`

F. **Submission**

To submit the electronic portion of this homework:

1. Make sure you are somewhere within your working directory ∼`/OS`, and that you have performed all the commits indicated above.

2. Use
   ```
   % git commit --amend
   ```
   to update your most recent commit message to *add* the following:
   ```
   submit HW2: complete
   ```
   Modify that added string if you have any clarifications about this submission (e.g., `submit HW21: parts A-C and D2`). You can use `git commit --amend` again later if you want to indicate an update.

3. Finally, `pull/push` your committed code to stogit.
   ```
   % git pull origin master
   % git push origin master
   ```
   Note: Always pull before you push.

The commands above should submit these files:

Files: `prtenv.c mopen.c`