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

Homework 7 Due Monday, 10-5-20

A. **Scheduling**

p.174 43, 45, 49, 58

B. **FILE I/O**

1. Copy the example program ~rab/os/egs/FILE.c to your homework directory for this assignment, and compile and test the program.

   Notes:

   - This program reads the lines from a pre-existing file, and prints each of those lines to the standard output.
   - Before running the program, create a file with at least one line of input named `myfile.txt`, using *emacs* or another method.
   - The function `fopen()` opens a file for I/O operations. In `FILE.c`, `fopen()` is used to open `myfile.txt` for input (`"r"` or *read* access). Other options are to open the file for output (`"w"`, for *write*) and for appending (`"a"`). See the manual page (section 3, for libraries) for `fopen()` for complete information.
   - Observe that the return value from `fopen()` has type `FILE*`. A successful call to the standard library function `fopen()` allocates memory for a `FILE` object and returns a pointer to that object.
   - Thereafter, we can use that returned `FILE*` value instead of `stdin` as an argument for `getline()`, in order to read a line from that file (instead of reading from standard input).
   - Other library functions that use `FILE*` arguments include the following (See manual pages for more information).
     - `feof()`, which returns a non-zero value if its `FILE*` argument has an end-of-file position;
     - `fputs()`, which performs output of a (null-terminated) string on a `FILE*` object, such as the standard output `stdout`; and
     - `fclose()`, for closing a connection to a file via a `FILE*` value.

2. Create a C program `copyfile.c` that prompts for and reads one line of standard input, parses it into a `name` data structure as in Lab 2, and copies a file named in the first token into a new file named in the second token.

   Suggested steps:

   - Start with a copy of the final version of the program `echoline.c` you wrote for Lab 2, which reads a line of input, parses it into a `name` structure, then prints those tokens (in a different order). *Test this program...* Modify the prompt to request two filenames.
   - Then, replace the code that prints the tokens by code that attempts to open and read the lines a file named by the first token. See ~rab/os/egs/FILE.c for example code that opens a file with error checking, then prints the lines from that file onto standard output.

- Finally, attempt to open a (new) file whose name is the second token, but for output instead of input. Be sure to check for any error and use perror to print any error that arises. If there is no error, print the output to that (new) file instead of `stdout`. At the end, be sure to `fclose()` the second file.

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

```
% git add copyfile.c
% git commit -m "HW7 B complete:  copyfile.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 Swapping

p.255 4

D 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 HW7: 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: `copyfile.c`

To submit by-hand parts, you can use the page `https://www.stolaf.edu/people/rab/os/asgt/hw7+.html`