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

Homework 6 Due Monday, 9-28-20

A. **IPC**

p.174 12, 35

B. **P-threads**

Note: The program `try_pthreads.c` is part of HW5.

1. Write a modified version of your program `try_pthreads.c`, called `try_pthreads2.c`, that performs the following additional operations, using a pthreads mutex lock.
   - Before printing the initial message, define an `int` variable `val`, initialized at 10.
   - The *pthread* should *add 1 to* the (shared) variable `val` in a thread-safe way (use a pthread mutex lock).
   - The `main()` should *negate* the variable `val` in a thread-safe way (use the same pthread lock as used for the pthread). This negation should take place sometime after the thread is created and before the join operation.
   - After printing the message that the pthread has finished and just before exiting, the `main()` should print the value of `val`, followed by a newline.

   Compile and run your program `try_pthreads2.c` to test it.

   Notes:

   - Use the example provided in class (`pthreads.c`) to determine the library calls for using a pthread mutex lock to insure that `val` is modified in a thread-safe way. Be sure to define and initialize the lock variable, and use the same lock variable for both `main()` and the pthread. Use the man page to check about headers, etc.
2. Make multiple test runs of your program `try_pthreads2.c` in order to answer the following questions.
   a) Is the final value of `val` consistent among all the runs, or does it vary? If that value varies, can you explain that?
   b) Do you see any evidence of a race condition in the output from your multiple runs of `try_pthreads2.c`? If so, can you explain what went wrong?
3. Write a modified version of your program `try_pthreads2.c`, called `try_pthreads3.c`, in which the `main()` *adds 2* to the variable `val` (instead of negating `val`). Verify that the program consistently returns the value 13.
4. Write a modified version of your program `try_pthreads3.c`, called `try_pthreads4.c`, in which `no locks are used` to guarantee thread safety (neither for `main()` nor for the pthread). Make several runs of this program. Does this program consistently return the value 13? If not, explain, why.
5. Create a git `commit` containing your work on this segment.
   ```
   % git add try_pthreads2.c try_pthreads3.c try_pthreads4.c
   % git commit -m "HW6 B5 complete:  try_pthreads2.c try_pthreads3.c try_pthreads4.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 `commits`, using the `commit` messages to indicate your progress.

C. Review - process diagrams

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

   a) `cat file1 file2 | diff - file3 > diff.out`

2. Make a copy of the program `mopen.c` in and modify `mopen.c` to recognize the command string `"close"` that tests the system call `close()`.

   - Follow the same strategy as you did for `dup()` in a previous homework.
   - Be sure to call that system call properly, e.g., check its return value for a possible error.
   - Test your program by closing an open file descriptor then attempting to `read` or `write` to that file descriptor. An error in that `read()` or `write()` call should be detected and reported
   - Also test closing a file descriptor that isn't open. An error should result and be reported.

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

   ```
       % git add mopen.c
       % git commit -m "HW6 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 `commits`, using the `commit` messages to indicate your progress.

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 HW6: 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: `try_pthreads2.c try_pthreads3.c try_pthreads4.c mopen.c`

To submit by-hand parts, you can use the page `https://www.stolaf.edu/people/rab/os/asgt/hw6+.html`