Homework4

CST 334: Operating Systems
Dr. Glenn Bruns

Homework - Week 4

Important note! All homework is to be done on your own.

1. **Reading**. Read OSTEP chapter 8 (MLFQ) and answer the following questions by editing hw4.txt.

    1.1. What assumptions does MLFQ make about jobs? a) the running time of jobs is known when they start, b) jobs start at the same time, c) once a job starts, it will get the CPU until it's complete, d) none of these

    1.2. Which kind of job tends to get highest priority with MLFQ? a) CPU-bound processes, b) mixed processes, c) I/O bound processes,.

    1.3. What is the problem with performing priority boosts too frequently? a) CPU-bound jobs get higher priority than desired, b) I/O-bound jobs get higher priority than desired.

    1.4. In Fig. 8.2 of OSTEP, how is the priority of the job changing over time? a) it's increasing, b) it's decreasing, c) it's staying the same

    1.5. In Fig. 8.3 of OSTEP, how many jobs are shown? a) 1, b) 2, c) 5

    1.6. In Fig. 8.4 of OSTEP, the gray job is: a) I/O intensive, b) CPU intensive, c) neither.

    1.7. In MFLQ, the length of time-slices on higher priority queues tends to be a) shorter than on lower-priority queues, b) longer than on lower-priority queues.

    1.8. On the left of Fig. 8.5 of MLFQ, the black job a) has finished, b) is waiting to be scheduled.

    1.9. (Y/N) Do priority boosts help with the problem of jobs that seek to "game the system"?

    1.10. (T/F) If MLFQ determines a job is I/O intensive, the job will be treated as I/O intensive until it terminates.

2. **msh**. We will continue extending our ‘msh’ shell. First, try running ‘cd’ in your msh and see what you get. Probably your current directory did not change. Think about why this is so, then extend your msh in the two following ways:

    2.1. Add a “built-in” command ‘cd’ in your msh shell. This command should work in two ways, just like in bash: 1) if no argument is supplied, change to the user’s home directory, and 2) if an argument is supplied, change to the directory specified by the argument. Hint: to get the user’s home directory, your code will have to read the value of the HOME environment variable. Investigate the ‘getenv’ library function and ‘chdir’ system call. Here is the sort of output your new msh should produce:

```
$ ./msh
  msh> ls
  backup  Makefile  msh  msh.c  msh-hw-2.c  README.txt  temp
  msh> pwd
  /home/CLASSES/brunsglenn/ctests/msh
  msh> cd backup
  msh> pwd
  /home/CLASSES/brunsglenn/ctests/msh/backup
  msh> cd ..
  msh> pwd
  /home/CLASSES/brunsglenn/ctests/msh
  msh> cd baz
  msh: cd: No such file or directory
  msh> cd
  msh> pwd
  /home/CLASSES/brunsglenn
  msh> exit
  $
```

Some system calls set the error number variable 'errno'. See the previous homework on how to use function strerror() to get an error message from the errno value. I used this in my code for errors related to changing directory.

2.2. Allow msh to read input from a file rather than from standard input. If msh is invoked from bash with a command-line argument, then msh should attempt to open the file specified by the command-line argument, and use that file as input. Your msh should terminate when the end of the file is reached (note that ctrl-d means “end-of-file” in bash). However, msh should still be able to be invoked without a command-line argument (as shown in the previous part of this exercise).

Hint: you are probably already using function fgets to read input from the user. Notice that the last argument of fgets is a “stream”, of type FILE *. When you call fgets you are probably supplying stdin as the input, which tells fgets to get input from the keyboard. Look at the man pages for functions fgets and fopen.

Here is an example of how msh should work with a filename on the command line:

```
$ ls
backup  Makefile  msh  msh.c  msh-hw-2.c  README.txt  temp
$ cat temp
ls
$ ./msh temp
backup  Makefile  msh  msh.c  msh-hw-2.c  README.txt  temp
$
```

Note that no prompts are produced when msh is run from a script.

Testing your code: On mlc104, the directory `/home/CLASSES/brunsglenn/cst334/hw/hw4` contains four test files test1.sh, …, test4.sh. Copy these and the Makefile to the directory where you developed your file msh.c. Each test should give exit status 0, like this:

```
$ ./test1.sh
$ echo $?
0
```

You need to run test1.sh first, as it will compile your code and produce binary file 'msh' that is used by the other tests. The directory also contains a Makefile. If you enter the command 'make', the target 'tests' in Makefile will run, causing each test to run. If you enter the command 'make clean', temporary files created by testing will be deleted.

Make sure to test your code for reasonable behavior beyond the unit tests I provide.

3. **Peerwise.** Spend about an hour on Peerwise. Create a question related to some in chapters 6-8 of OSTEP , or on something from lecture. Also, answer, rate, and comment on other students' questions.

Submitting. Submit your homework on iLearn as two files: an edited hw4.txt for part 1, and an msh.c file for part 2. Sorry to do this, but a 20 point penalty will be applied for submitting zip files or other files in the wrong format.

Grading. Part 1 is worth 30 points: 3 points per question. Part 2 is worth 70 points: 10 points for each of 6 test cases, and 10 points for tidy code. The test cases I run may differ a little from the test cases I have supplied you with. The peerwise work of Part 3 is graded separately.

Please use Slack for questions to clarify what is being asked. Please do not post anything on Slack that will reveal something about your answers.