Homework 5

CST 334: Operating Systems
Dr. Glenn Bruns

Homework - Week 5

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

1. **Reading.** Read OSTEP chapters 13 and 14 and answer the following questions by editing hw5.txt:

1.1. What kind of memory allocation occurs on the blue line of code below?

```
int my_fun() {
  char *s = (char *) malloc(10);
  …
}
```

a. stack allocation

b. heap allocation

c. both stack and heap allocation

d. neither stack nor heap allocation

1.1. Which is the correct way to allocate memory to make a copy of a string foo? (This code shows only the memory allocation, not the copying.)

a. `char *s_copy = (char *) malloc(foo);`

b. `char *s_copy = (char *) malloc(strlen(foo));`

c. `char *s_copy = (char *) malloc(strlen(foo) + 1);`

1.1. The line of code below can be fixed by changing one thing? What is it?

int *xp = (int *)malloc(1);

a. The argument passed to malloc()

b. The (int *) in front of malloc

c. The int * at the beginning of the line

1.1. On mlc104, what is the value returned by sizeof(int)?

1.2. On mlc104, what is the value returned by sizeof(float)?

2. **Bash**. This week you will create two awk scripts.

2.1. `get-problem.awk` This script finds and output a problem in a homework file.

Look at the file student-code.sql to see what a homework file looks like. Each problem is described on a problem line that starts with something like "-- 4. ". The student response is on the lines following the problem. Given an input value, like 4, the script should return the student response to the problem. Please note that a student response ends with either: a new problem line, the end of the file, or a line that begins with three or more hyphens.

2.2. `xform-c.awk` This script transforms a csv file of scores.

An example input file is scores.csv. The purpose of the script is to transform the input file in the following ways: 1) in the header line, "prob_desc" should be changed to "comments". In the non-header lines, 2) "Participant " should be added in front of "Identifier" value, 3) prob_id values in the second field should be renamed from 1-5 to 3_a, 3_b, …, 3_e, and 4) if the input score value is 1, the transformed output value should be 0, and otherwise if the input score value is 0, the transformed output values should be 10, 5, 5, 5, 5, respectively for problem ids 1 through 5.

Your solution should not depend on the ordering of rows in the input file, except the header row will always come first.

On mlc104, the directory `/home/CLASSES/brunsglenn/cst334/hw/hw5` contains a file named hw5.tar that is a tar archive of the files you need to test your code. Copy hw5.tar to your own directory and extract the files in it (see file "tar - extracting files from an archive" in the "bash 2: permissions" lecture of week 2). After you extract the files you'll see a directory for get-problem.awk and another one for xform-c.awk. Those directories contain Makefiles and test scripts. Also, the files get-problem.awk and xform-c.awk are in the directories, but you need to edit those files. Do not edit the other files!

Note: you will need "shebang line" `#!/usr/bin/awk -f` at the top of your awk scripts.

3. **msh**. We will continue extending our 'msh' shell. This week we will add only one small feature: redirection. This feature will allow commands to use files for input and output, instead of standard input and standard output. To direct a command's output to a file, the syntax "> outfile" is used. To read a command's input from a file, the syntax "< infile" is used.

Your extended version of msh should do what last week's msh could, plus should be able to handle commands like the following:

```
$ ./msh
  msh> ls -l > temp.txt
  msh> sort < temp.txt > temp-sorted.txt
```

The result of these commands should be that the sorted output of "ls -l" is in file temp-sorted.txt.

Your shell builtins (like 'cd' and 'help') do not have to handle redirection. We will discuss how to implement redirection in class. Only one new Linux command is needed: **dup2**. You will use dup2 for both input and output redirection. You may want to look at the slides on dup2 that are included with the homework assignment. The basic idea is that if you see redirection on the command line, you open the file or files, and then use dup2. Think: do you want to do this in the child process that gets forked, or by the parent process?

Also, you need to handle the ">" and "<" characters that appear on the command line. Think about whether or not you put them into an array of tokens.

Testing your code: On mlc104, the directory `/home/CLASSES/brunsglenn/cst334/hw/hw5` contains two test files test1.sh and test2.sh. Copy these and the Makefile to the directory where you develop 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.

4. **Peerwise.** Spend 30-60 minutes on Peerwise. Consider creating a question related to chapters 13 or 14 of OSTEP , or something from lecture. Also, make a meaningful comment on another student's question, rate another student's question, and answer some questions.

Submitting. Submit your homework on iLearn as 4 files: your edited hw5.txt for part 1, files get-problem.awk and xform-c.awk for part 2 and msh.c for part 3. 20% penalty on homework for submitting files of the wrong type, like zip or tar files.

Grading. Part 1 of the homework is worth 20 points (4 pts/question), part 2 is worth 40 points (20 pts for each script), and part 3 is worth 40 points.

For the msh code, 10 points will be given for each of 3 test cases, and 10 points will be given for neatly formatted and appropriately commented code. No “tidy code” points if your code isn’t close to being right. Your code should pass tests from previous weeks, plus any new tests related to redirection.