zzCST 334: Operating Systems
Dr. Glenn Bruns

Homework - Week 1

Important note! All homework is to be done on your own. Do not discuss it with fellow students.

In this and many other homework assignments you will provide answers by editing a text file, which I will grade with a script. This week, you will provide answers to part 1 and 2 of the homework by editing file `hw1.txt`, which is attached to the homework assignment on iLearn.

1. **Reading**. Read Chapter 2 of our OSTEP text and answer the following questions:

   a. In the C program of Figure 2.1, when does the while loop that begins on line 15 terminate? (select one)
      a. never
      b. immediately
      c. after the body of the while loop is executed once

   b. What does it mean to “virtualize a CPU”? (select one)
      a. there is no hardware CPU, only a software CPU
      b. programs can be written as if they didn’t have to share a CPU
      c. the instructions supported by the CPU are virtual

   c. Which of the following are Linux systems calls? This problem is not answered in chapter 2. For a hint, look at the man page for command ‘man’. It's possible that some of the following are both bash commands and system calls. (select all that apply)
      a. cd
      b. fork
      c. ls
      d. rmdir

   d. Figure 2.4 shows the output from two copies of program ‘mem’. What is being illustrated by lines 4 and 5? (select one)
      a. two processes can have the same virtual memory address
      b. two processes can have the same physical address

   e. What is emacs? (select one)
      a. a text-based calculator
      b. a text editor
      c. a Linux system call

2. **Binary and hexadecimal numbers**. Write the following binary numbers as decimal numbers. For example, write 10110 as 22. Please do not use web resources to solve any of these problems.

   a. 10110 _____
   b. 111 _____
   c. 0011 _____

d. 11001 _____

e. 11111111 _____

Write the following decimal numbers in binary. For example, write 13 as 1101.

f. 13 _____

g. 21 _____

h. 16 _____

i. 55 _____

j. 80 _____

Write the following hexadecimal numbers as decimal numbers. For example, write 0x2C as 44 (which is 2*16 + 12*1). The “0x” is often used to indicate hex numbers.

k. 0x2C _____

l. 0x12 _____

m. 0x1D7 _____

n. 0xFF _____

o. 0x7B _____

Write the following decimal numbers as hex. For example, write 18 as 0x12 (which is 1*16 + 2*1). Do not forget to provide the “0x”.

p. 18 _____

q. 5 _____

r. 88 _____

s. 123 _____

t. 100 _____

3. **C**. Write a program bindec.c that will take a binary numbers as input, and output the decimal equivalent. I have supplied starter code.

a. on mlc104, copy bindec-skeleton.c to a directory of your own. For example:

```
$ cp /home/CLASSES/brunsglenn/cst334/hw/hw1/bindec-skeleton.c <yourdir>
```

b. rename bindec-skeleton.c to bindec.c

```
$ mv bindec-skeleton.c bindec.c
```

c. use a text editor (vim or nano) to edit the code. Look for "YOUR CODE HERE". Modify bindec.c so that it converts the input string into a number, and prints the number. Write the code to convert from binary to decimal yourself (don’t use a library call, or code found on the internet).

d. your code should respond to empty input with value 0, and should respond to invalid input with message ‘input must contain only zeros and ones’, and exit with value 1, indicating an error. Your code should be able to handle inputs of at least 20 binary digits.

Some examples of how your program should work:

```
$ ./bindec
> 1001
```

```
9
$ ./bindec
> 11111111
255
$ ./bindec
> hello
input must contain only zeros and ones
$ ./bindec
>                        (no input given, just hit 'enter')
0
```

Testing your code. In the directory `/home/CLASSES/brunsglenn/cst334/hw/hw1` there are some scripts you can use to test your code. The scripts assume your code is named 'bindec.c'. Here's how to run a test:

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

The result of the second command will be 0 if your code is correct, and some other number if your code is not correct. You need to run `test1.sh` first; it will test to see if your code compiles, and will generate a file named `test-code` that the other test scripts will use.

When I test your code I will use these scripts plus a few additional ones.

Copying your code from mlc104. Note: to copy a file from the hosting server to your own machine, using the bash command 'scp'. Here is an example:

```
scp bruns1992@mlc104.csumb.edu:/home/CLASSES/brunsglenn/cst334/hw/hw1/hello.c .
```

This is how I would copy the file; you would replace 'bruns1992' with your own user name on mlc104. Also, you can replace the '.' at the end with any local file name.

On windows machines you can use Putty pscp, which can be downloaded from the putty download page. pscp is used on the command line just like scp. An easier option is WinSCP, a free app. https://winscp.net/eng/index.php

4. **Peerwise**. Register yourself on peerwise and create a question based on Chapter 2 of our text, or on something from lecture not in the text. Also, if possible, make a comment on another student's question, or rate another student's question. Please be thoughtful in your work so that the results are useful for the whole class.

peerwise.cs.auckland.ac.nz

our school is listed as CSU, Monterey Bay

our course IDs are 18631 (section 1) and 18632 (section 2)

5. **Prerequisite quiz**. Take the CST 238 Articulation Verification Test that can be found at https://ilearn.csumb.edu/course/view.php?id=1821. The password is otter2015. You don't need to take the test if you've taken it before and scored at least 75.

You can use Slack to ask for clarification about homework problems, but please don't post answers. You can help other students by providing hints.

Submission. Submit your homework on iLearn as two files: your edited `hw1.txt` file for parts 1 and 2, and your bindec.c file for part 3. Do not submit a zip file -- submit two files!

Grading. The homework is worth 100 points (maximum).

- 20 points for part 1 (4 pts/question)
- 40 points for part 2 (2 pts/question)
- 40 points for part 4

However, your score will be cut in half if you don't take the CST 238 AVT.

Details on grading for the C code: 10 points for compilation, 5 points for each of 4 additional tests, 10 points for tidy and properly formatted code. No points for compilation if code is obviously incorrect.