Homework 2: Python and more C refresher

Due: Monday January 20, 2014 23:59:00 (11:59 PM) Pacific USA time zone

Points on this assignment: 175 points with 10 bonus points available.

Work submitted late will be penalized as described in the course syllabus. You must submit your work twice for this and all other homework assignments in this class. Ecampus wants to archive your work through Blackboard and EECS needs you to submit through TEACH to be graded. If you do not submit your assignment through TEACH, it cannot be graded (and you will be disappointed with your grade). Make sure you submit your work through TEACH. Submit your work for this assignment as a *single tar.bzip file* through TEACH. The same single tar.bzip file should also be submitted through Blackboard.

Place all of the files you produce for this assignment in a single directory called `Homework2`. Remember to not put spaces in filenames or directory names.

In **all** your source files (including the Makefile), you need to have 4 things at the top of every file as comments:

1. Your name
2. Your email address (ONID or engineering)
3. The class name and section (this is CS311-400)
4. The assignment number (this is homework #2).

1. **30 points**. When you are ready to submit your files for this assignment, make sure you submit a *single tar.bzip file*. Review homework #1, problem #1 if you need a refresher on how to do this. If your file is not a single tar.bzip file, you cannot receive points on this assignment.
2. **15 points**. Write a Python script that creates the following directories (under your home directory or a subdirectory of your home directory) for a given `term` and `course`:

   - `assignments`
   - `examples`
   - `exams`
   - `lecture_notes`
   - `submissions`

   Also create a symbolic link to `/usr/local/classes/eecs/<term>/<class>/src/README` called `'README'`, and a link to `/usr/local/classes/eecs/<term>/<class>/src` called `'src_class'`. The `src_class` is in **your** directory.

It is your responsibility to ensure that you don't attempt to create a directory that already exists (it should produce a warning message, not an error from Python). Make use of the Python modules `os`, `sys`, and `getopt`, with both short and long form options for term and directory.

You should be able to pass the <class> on the on the command line with a `-c` switch. You should be able to pass <term> on the command line with a `-t` switch. So, running your script should look like this:

```
python Problem2.py -t winter2013 -c cs311-400

python Problem2.py --term winter2013 --class cs311-400
```

3. **20 points**. Using the `urllib2` module in Python, write a Python script equivalent to the command line utility `wget`. Only basic usage need be supported for this assignment: this is a tool that can be used to retrieve an arbitrary file based on URL and save it into a named file.

   You only need to worry about two command line arguments, the URL from which the file is to be downloaded and the name of the file into which the content are to be saved.

   ```
   python Problem3.py www.python.org pythonHomePage.html

   python Problem3.py google.com googleHomePage.html
   ```

4. **25 points**. Find and print (in decimal) the 2400th prime number, using only Python (Remember the last part of the Homework #1?). Use a command line argument to determine which prime number you will print. When your program is done, I should be able to print other prime numbers using it.

   ```
   python Problem4.py 2400
   ```

5. **20 points**. Use the Python subprocess.Popen() call to connect to the Unix/Linux command "`who`". Print the results from running `who` through the Popen. Don't worry about processing command line arguments, just plain `who`. Make sure you use subprocess.Popen() (not os.popen*()). The output from your Python script (Problem5.py) should look just like the output from `who`. **For extra credit (10 points)**, process the following command line arguments for `who, -bdlprTtu`. The extra credit output from your script (Problem5.py) should look just like the output from running `who` with those command line options. You can look at the man page for `who` to see what the options do.

   In addition to the pages in the standard online tutorial on Python Popen(), here are a couple additional resources:

SaltyCrane Blog – this is a good one

PyMOTW – subprocess

6. **20 points**. Using a Python script (Problem6.py), print the greatest product of five consecutive digits in the 1000-digit number.

```
53697817977846174064955149290862569321978468622482
83216637048440319989000889524345065854122758866688 1
16427171479924442928230863465674813919123162824586
17866458359124566529476545682848912883142607690042
65727333001053367881220235421809751254540594752243
52584907711670556013604839586446706324415722155397
07198403850962455444362981230987879927244284909188
84580156166097919133875499200524063689912560717606
05886116467109405077541002256983155200055935729725
24219022671055626321111109370544217506941658960408
12540698747158523863050715693290963295227443043557
66896648950445244523161731856403098711121722383113
62229893423380308135336276614282806444486645238749
30358907296290491560440772390713810515859307960866
70172427121883998797908792274921901699720888093776
71636269561882670428252483600823257530420752963450
73167176531330624919225119674426574742355349194934
83972241375657056057490261407972968652414535100474
96983520312774506326239578318016984801869478851843
85861560789112949495459501737958331952853208805511
```

The first product is (5 * 3 * 6 * 9 * 7). The second product is (3 * 6 * 9 * 7 * 8)…

7. **25 Points.** Write a C program that uses `getopt()`. Your C code should accept 3 different command line options: -h, -t, and –f. Your code should return results as shown in the table below:

Switch	Meaning	Example
`-h`	Give the base hostname, not fully qualified hostname.	`Hostname = orange2`
`-t`	Show the current time	`The current local time: Sun Apr 7 23:34:09 2013`
`-f Problem7.c`	Show the size in bytes of the file given as additional parameter.	`Size of file 'Problem7.c' is 762 bytes`

The parameter for the `-f` command line option should be able to be any file in the file system.

This problem is about getting accustomed to using `getopt()`, not about making the system calls. You'll get a lot more practice making the system calls in following assignments. So, here is some supplemental information about making the system calls necessary for this problem.

Include the following header files in your C code.

```
#include <stdio.h>
#include <getopt.h>
#include <sys/utsname.h>
#include <time.h>
#include <sys/stat.h>
```

Define the following variables near the top of `main()`.

```
char c;
struct utsname uname_pointer;
time_t time_raw_format;
struct stat s;
```

Print the hostname using the following commands:

```
uname(&uname_pointer);
printf("Hostname = %s \n", uname_pointer.nodename);
```

Print the current time using the following commands:

```
time(&time_raw_format);
printf("The current local time: %s",
ctime(&time_raw_format));
```

Print the file size using the following commands:

```
if (stat(optarg, &s) == 0) {
    printf("size of file '%s' is %d bytes\n"
         , optarg, (int) s.st_size);
} else {
    printf("file '%s' not found\n", optarg);
}
```

8. **20 Points**. Save the following C code as a file on your system. Name the file Problem8.c. This will be a lot easier on a Linux system (`eos-class` or CS311 VM)

```
#include <stdio.h>
#include <stdlib.h>

#define NUM_INTS 100

void func(int *ptr)
{
  int i = 0;
```

```
  for (i = 0; i <= NUM_INTS; i++) {
      ptr[i] = i;
  }
}

int main (int argc, char *argv[])
{
  int *intArray = NULL;

  func(intArray);
  return 0;
}
```

Compile the C program using the following `gcc` command (the `-g` option is **very** important):

```
gcc -g -o Problem8 Problem8.c
```

Once you have compiled the program, load the program into the gdb debugger by using the following command (you will need to be at a shell prompt in the same directory a you used to compile the C code):

```
gdb Problem8
```

The prompt you see should indicate that you are within the gdb debugger. You should see `(gdb)` as the prompt.

From the gdb prompt, issue the following commands.

```
run
bt
print ptr
print i
```

After you issue the `run` command, you should see a response that the program has had a "Segmentation fault." The `bt` command will show you the "back trace" of the faulted program, you see some addresses, function names, and line numbers. The 2 `print` commands will show you the value of 2 variables at the time the code faulted.

Copy and paste the text between the `run` command and after the second `print` command. Paste the text into a file (`Problem8.txt`) and submit it as part of this homework assignment.

Once done, you can enter `quit` at the gdb prompt and then `y` to the "Exit anyway" question.

You have now experienced running and using the GBU debugger gdb. You have used the most basic commands to show you where a program has faulted and it will show you the line number where the code faulted. This is important information that you'll need to make use of during the term.

Things to include with the assignment (in a single tar.bzip file):

1. Python source code for problems 2-6
2. C source code for problems 7 and 8.
3. A text file from Problem 8.
4. Remember to put your name in every file you submit.