Homework 1: General

Due: Monday 8 April 2013, 23:339:00 (11:59 PM) Pacific USA time zone

Points on this assignment: 150 (with no extra credit available on this assignment).

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 bzip file* through TEACH. The same single bzip file should also be submitted through BlackBoard.

---

1. When you are ready to submit your files for this assignment, make sure you submit a *single bzip file*. **50 points**
    1. Here is how you can submit your files as a single bzip file.
        1. Put all your files for this assignment into a directory called `Homework1`. The `Homework1` directory can have additional sub-directories, such as one called `Question6`.
        2. Change directory up one level above the `Homework1` directory.
        3. From the directory above `Homework1`, enter the following command. Where <yourname> is replaced with your onid/eecs login name (mine is `chaneyr`).

            ```
            tar cvfj Homework1-<yourname>.tar.bzip Homework1
            ```

            In my case, I would create a single bzip file named

            ```
            Homework1-chaneyr.tar.bzip
            ```

    2. If your file is not a single bzip file, you cannot receive points on this assignment.
    3. Don't create some other format archive file and just rename it Blaa.tar.bzip. If your files for this (and other assignments) cannot be extracted by `tar` on a Linux system, the assignment cannot be graded (and you will be disappointed with your grade).
    4. Once you have your single bzip file, go to the TEACH site and submit it. Make sure your submission goes all the way through.
2. What are Revision Control Systems (sometimes called Version Control Systems)? Why are they useful? Name 3 examples of revision control systems. **10 points**.

A note for simple RCS: there are many revision control systems available from many different vendors or open source projects. RCS has been around since 1982. While newer revision control systems have more features to support group projects with big network-based repositories, it remains difficult to beat RCS for simplicity for small projects in a single directory. Knowing 2 simple RCS commands is just about all you'll need to be efficient with RCS: `ci` and `rcsdiff`. RCS is installed on `os-class` and the CS311 VM. If you choose to use RCS and read one of the many tutorials on the web, don't get wrapped up in `rcsmerge`, tagging branches and such. You'd probably really only need "`ci -l`" and (maybe) `rcsdiff` for this class.

3. Describe the difference between redirection and piping on Unix. **10 points**.
4. Give a `find` command that will run the `file` command on every regular file (not directories!) within the current file system sub-tree (pwd and down). Run this from within your home directory. HINT: look at the man pages for both `find` and `file`. The man page for `find` is long, but has some really good examples in it, near the end of the man page. The man page for `file` tells you about "magic tests" and "magic numbers." Man pages are good. **10 points**.
5. What is Unix command `make`, and how is it useful? **10 points**.
6. Compile the 4 C source code programs located in the attachment on BlackBoard for this assignment (it will be labeled Homework1-Question6.tar.bzip. Once you have downloaded the file, you'll need to extract the contents into a location where you can modify the files. I suggest you create a directory structure like this:

   ```
   ~/src/Homework1/Question6
   ```

   You will need to create the directory/directories first ("`mkdir ~/src ~/src/Homework1 ~/src/Homework1/Question6`" is a nice command to try). Move the tar.bzip file into the Homework1 directory and extract files with this command:

   ```
   tar xvfj Homework1-Question6.tar.bzip
   ```

   The contents will extract into a directory called Question6. You can edit the files from within there.

   Each of the (small) programs has some syntax errors. Correct all the syntax errors so that when you compile using "`gcc -Wall`" you see no warnings or errors. Do not just comment out all the code. After you correct the syntax errors, the programs should run and request input (helloworld does not request input) and we will check for that. You may want to look over question 7, below, before going too far on this portion of the assignment. **20 points**.

   The objective on this question is to refresh your feel for seeing compiler warnings and errors; then finding and correcting them in the source file. When warnings and errors are generated by the gcc compiler, the messages indicate line number where (or near where) the flaw was detected. This is a very important skill to have for this class.

7. Take the small programs for which you corrected the syntax errors (question 4 above) and create a single `Makefile` that will compile each of them. There is a starter `Makefile` in the same directory where you found the C source. You should have a target for each of the 4 programs. You can leave both the C source and the Makefile in a single directory. **10 points**.
8. The *Sieve of Eratosthenes* identifies all prime numbers up to a given number n as follows:
    1. Write down the numbers 1, 2, 3, ..., n. We will eliminate composites by marking them. Initially all numbers are unmarked.
    2. Mark the number 1 as special (it is neither prime nor composite).
    3. Set k = 1. Until k exceeds or equals the square root of n do:
        1. Find the first number in the list greater than k that has not been identified as composite. (The very first number so found is 2.) Call it m. Mark the numbers

           `2 x m, 3 x m, 4 x m, ...`

           as composite. (Thus in the first run we mark all even numbers greater than 2. In the second run we mark all multiples of 3 greater than 3.)

        2. m is a prime number. Put it on your list.
        3. Set k = m and repeat.
    4. Put the remaining unmarked numbers in the sequence on your list of prime numbers.

    Write a Python function (on `os-class` or a CS311 VM), which returns the number of primes less than a given value, as well as the values of all the primes. Your Python function should take a parameter indicating the value for which smaller prime numbers are to be generated. 3**0 points**.

Things to include with the assignment (in a single bzipped file):

1. A write up for questions 2-5. You can use MS Word, LibreOffice Writer, or just plain text files for the write-ups. Use reasonable formatting. A single 2000 character line of text is not reasonable formatting.
2. All C source and the Makefile for questions 6 and 7.
3. Python source code for the solution to question 8 (all files).
4. A file describing how you ran any tests.
5. See Question 1 in this assignment for how you can bundle your files into a single bzip file for submission through TEACH.

Please combine all of the above files into a single bzip file for submission through the TEACH web page.