Assignment 2, due September 17

The objectives of this assignment are

- to introduce you to some simple UNIX commands,
- to give you practice working with directories and the file hierarchy, and
- to give you practice using your UNIX account.

It should not take long to do this assignment. Everything that you need to know is contained in Lessons 1, 2, 3, and 4 or in this document. Most of it has been covered in class already.

Background

When you type a command in UNIX, by default it displays its output on the screen, which is called the ***terminal window***. For example, if I enter the `date` command, it will display today's date and time in my terminal window, as shown here:

```
$ date
Sun Sep 3 19:10:52 EDT 2017
```

To be more accurate, commands send their output to what is called the ***standard output stream***, which is by default the terminal window. You can think of the standard output stream as a stream of characters flowing out of the command and into the terminal window. In UNIX, you can change where the standard output stream goes by ***redirecting*** it.

An analogy might help. Suppose the output produced by a command is like the cars of a train traveling down a track to a terminal station. Each character is like a single car on the train. When you run the command, it sends its data on the cars of this train down the track. By default the train goes to the terminal station. But we can divert the train by rerouting it with a switch onto another track to a different location, perhaps to a train yard or to a siding. This is what output redirection is.

For example, you can send the output of a command to a file or send it to another command, which can use it as its input. To redirect the output of a command into a new file, you use the ***output-redirection operator***, ">", followed by the file's name. This is the *greater-than* symbol found on the keyboard above the *period*. For example, suppose that there is no file currently named `now` in my home directory. If I type the command

```
$ date  > ~/now
```

then a file named `now` will be created in my home directory and it will contain the output of the `date` command.

In general, if you type any command followed by "> *`filename`*" where *`filename`* is the name of a ***new*** file, *`filename`* will be created and will contain the output of that command. For example,

```
$ who > current_users
```

will create a file named `current_users` containing the output of the `who` command, which is information about who is currently logged on to the computer. If you try to send the output to a file that already exists, it will not overwrite it. It will instead issue an ***error message***. For example, if I type the two commands

```
$ who > current_users
$ who > current_users
-bash: current_users: cannot overwrite existing file
```

then the second `who` command prints the above error message to your terminal.

If you want to add something to the end of a file that already exists, then you need to use the `append-redirection operator`, "»", (two greater-than symbols with no space in between). We illustrate this next, but first we introduce another command named `echo`.

The `echo` command is a very useful command. It displays whatever argument you give it on the standard output stream. In other words, it echoes what you give it. For example,

```
$ echo hello
hello
$ echo goodbye
goodbye
$ echo This seems like a useless command.
This seems like a useless command.
$ echo "These quotes will not be displayed."
These quotes will not be displayed.
```

But we can use the `echo` command in a myriad number of ways. For example, suppose that the file named `current_users` does not exist in my home directory. I can issue the following three commands:

```
$ echo "The time is now" > ~/current_users
$ date  >> ~/current_users
$ who   >>  ~/current_users
```

which writes the phrase "`The time is now`" on the first line, the current date on the second line, and appends the output of `who` to the remaining lines as needed. (By the way, if you want the date to appear on the same line as "The time is now", then you can read the man page for the `echo` command to discover what option to give it to suppress its printing a new line.)

Do you see that `echo` together with output redirection can be used to create files?

The `head` command, by default, displays the first 10 lines of its file argument. For example, if I have a file named numbers whose first 15 lines are

```
1
2
3
4
5
6
7
8
9
10
11
```

```
12
13
14
15
```

then typing `head numbers` displays lines 1 to 10:

```
$head numbers
1
2
3
4
5
6
7
8
9
10
```

You can change how many lines it displays like this:

```
$ head -5 somefile
```

which will display the first 5 lines instead. In general,

```
$ head -n somefile
```

displays the first n lines of the file where n is an actual number.

Removing Files and Directories

If you make a mistake and need to delete a file or a directory, this is what you need to know:

To delete a single file named `foo`, type

```
rm foo
```

This command might ask you if you are sure you want to delete it. Answer 'y' for yes or 'n' for no.

To delete an empty directory named `dir`, type

```
rmdir dir
```

If `dir` is not empty, it will not delete it. You must empty it first. To delete a directory and all of its contents, you can type

```
rm -r dir
```

but it is risky business, so use it with caution! You can also type

```
\rm -rf dir
```

and it will definitely remove it without any hope of recovering it. Be careful!

For example, if you decided you want to replace the contents of the file `current_users`, then just delete the file and try again:

```
$ rm current_users
rm: remove regular file ‘current_users’? y
$ date > current_users
$ who >> current_users
```

Your Assignment's Tasks

1. Login to `eniac`, and then `ssh` to any `cslab` host. For the sake of precision, I will assume you logged into `cslab1`. You will be "in" your home directory when you login.
2. (20%) Create a directory named `your-username-hwk2` using the `mkdir` command, where you replace "your-username" by your actual username. For example, I would create `sweiss-hwk2`. Read the man page for how to use the `mkdir` command if you need help.
3. (20%) Type the command `hostname` redirecting its output to a file in this directory named `host_machine`.
4. (20%) Type the `date` command redirecting its output into a file in this directory named `current_time`.
5. (40%) The directory `/data/biocs/b/student.accounts/cs132/data/open_datasets` contains a directory named `usa_babynames`. Inside this directory are many files whose names are of the form `yob1880.csv`, `yob1881.csv`, `yob1882.csv`, ..., `yob2017.csv`. These files contain the names of babies born in the United States and who were given social security numbers in the given year. ("yob" stands for "year of birth".) In each file, the most common baby name given that year is the first name in the file, after which they appear in decreasing order of frequency. Each line in the file looks something like this:

   ```
   Mary,F,1209
   ```

   meaning that there was a name Mary given to 1209 different female babies. Your job is to create a file named `most_popular_names` inside your `your-username-hwk2` directory, containing the most popular baby names for a selected set of years. The years for which you must do this are the years assigned to you in the file contained in the `usa_babynames` directory named `years_to_include.csv`.This file has a row for each student in the class, and in each row is a set of six years. If your row, for example, contains the year 1902, then your file must have the first line of file `yob1902.csv`.
6. When you have finished, if you did it all correctly, you will have a directory with three files in it: `host_machine`, `current_time`, and `most_popular_names`. Your final task is submitting your work. This is described below.

That is all you have to do in this assignment. It should not be hard. The objective is to get you thinking and working. Make sure you name your files exactly as I specified above.

Grading Rubric

This homework is graded on a 100 point scale. Each of steps 2 through 5 is worth the percent indicated above.

Submitting the Homework

This assignment is due by the end of the day (i.e. 11:59PM, EST) on Monday, September 17. (I give a grace period of six hours after that, so it is okay to submit it by 6:00 AM of the following day.)

There is a directory in the CSci Department network whose full path name is `/data/biocs/b/student.accounts/cs132/hwks` You must put it in that directory.

To submit your project, you must follow the instructions below exactly! Do not deviate from these instructions.

To be precise:

1. Login using `ssh` to `eniac.cs.hunter.cuny.edu` with your valid username and password, and then `ssh` into any `cslab` host. Do not forget this step. You will not be able to run the `submithwk` command on `eniac`.
2. Make sure that you see the `your-username-hwk2` directory that you created in the assignment by typing `ls` and verifying that it appears in the output list. If you do not see it, then either you are in the wrong working directory or you misplaced it somehow.
3. Run the command

   ```
   zip -r your-username-hwk2.zip your-username-hwk2
   ```

   This will create the file `your-username-hwk2.zip.` The `zip` command is a special command that compresses the files in the directory and creates a new file that can later be extracted by the `unzip` command. So it will create a "zip file" named `your-username-hwk2.zip` containing your `your-username-hwk2` directory and the three files it contains. For example, I would run

   ```
   zip -r sweiss-hwk2.zip sweiss-hwk2
   ```

4. Run the command

   ```
   /data/biocs/b/student.accounts/cs132/bin/submithwk 2  your-username-hwk2.zip
   ```

   Do exactly this. Do not mistype it. The command will create a copy of the file `your-username-hwk2.zip` in the directory

   `/data/biocs/b/student.accounts/cs132/hwks/hwk2`

   It will be named `hwk2_`*`username`*, where *`username`* is your username on the network. You will not be able to read this file, nor will anyone else except for me. If you decide to make any changes and resubmit, just do all the steps again and it will replace the old file with the new one. I will be able to unzip the file, extracting whatever files you created. Do not try to put your file into this directory in any other way - you will be unable to do this.

Although these instructions may seem complicated, they simplify the way you submit your work and the way I can retrieve it. If you make mistakes, just start over. If things don't seem to work out, post a question on Piazza with the details included.