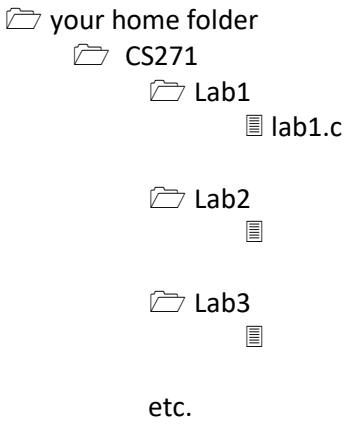
CS 271
Lab Assignment # 2

1. Log in to your Computer Science Linux account.

2. In the terminal window (Activities --> search for Terminal), change working directory to your home folder.

3. Organize your directories (folders) and files so that you have something like this:



Make your life easier... don't put spaces in the names of folders or files.

Here are some Linux commands that may be helpful:

Command	**What it does**
`cd` *`directory_name`*	changes working directory to *directory_name*
`cd ..`	changes working directory to one level up
`cd home`	changes working directory to your home directory
`pwd`	displays the working directory
`mkdir` *`directory_name`*	makes a new directory
`mv` *`source destination`*	moves (or renames) the source to the destination The source may be a file or a directory.

4. In the terminal window: Navigate to the lab 1 folder and remove the object files (files that end with .o)

   Suggested commands:

cd	(press Enter...takes you to your Home folder)
cd CS271/Lab1	(press Enter...takes you to the CS271 lab 1 folder, assuming that you named the folders correctly.)

   *** from here on I'm going to assume that you know to press Enter after a command ***

rm *.o	Caution: rm deletes things, there is no undo, and no "are you sure?" message

   Note: If you put a space in a filename or folder name, you'll need to surround it with quotes when you type it in a Linux command. For example: cd "lab 1"

Style Requirements

- When you write an inline comment, place it on a separate line before the program statements that perform the task. Use only the single line comment syntax. (Yes, there is a penalty in the rubric for using multiline comments.)

- Include header comments in the following format. This header format is expected on all programs you submit this semester.

  ```
  // CS 271 - lab assignment #
  //program_name
  // purpose of the program
  // written by yourname
  // date written
  ```

Download and Run the Sample Program

5. Download the files named fileInputExample.c and sampleData.txt from Canvas and save them in your lab 2 folder.

6. Open your editor (Kate, I hope) and open fileInputExample.c.

7. Read the code carefully. Pay particular attention to the variables that are declared, the way the data file is opened, and the fscanf statements used to read data from the file.

8. Open the data file sampleData.txt. You should see 4 lines of text. Each line has 3 fields: first name, last name, and age. There is one space between fields.

9. Go back to the tab with the program. Make sure that you understand how the program gets input from the file and stores it in the variables for first, last, and age.

10. In the terminal area of Kate, change your working directory to the lab 2 folder.

11. Compile fileInputExample.c.

12. Run the executable. You should see this output:

```
Average age of the 4 people in the study is 66.8.
```

If you don't see that, ask for help right away.

13. Close sampleData.txt. Close fileInputExample.c.

Create a C Program Named cereal.c

14. In the terminal area of Kate, copy the file named fileInputExample.c to a new file named cereal.c.

15. Download the file data2.txt from Canvas and save it in the lab 2 folder. Change the name of the file data2.txt to cereal.txt.

16. Open the file cereal.c in the editor area of Kate.

    Add header comments.

    Change the main function so that it will solve the following problem.

    Problem Description:

    A group of college students is looking for a healthier breakfast. They found a dataset that contains the nutrition information for 43 cereals.

    They want you to write a program to search the dataset for cereals that meet certain criteria. For this assignment, we'll limit the criteria to calories per serving, grams of sugar, and grams of protein.

    For example, they want to be able to run the program and look for cereals that have 3 grams of protein or more; less than 6 grams of sugar; and less than 200 calories per serving.

    The program will get input from two sources.

    Keyboard Input
    a) input the minimum number of grams of protein
    b) input the maximum number of calories
    c) input the maximum number of grams of sugar

    File Input:
    data2.txt is a text file containing cereal data. The data is organized as follows. Each line in the file represents the data for one cereal. The field values are separated by one space.

Field #	Description	Data Type
1	Cereal name, camel case, no internal spaces	string
2	Company, one character (G for General Mills, K for Kelloggs, Q for Quaker Oats)	string
3	Calories per serving	int
4	Protein (grams per serving)	int
5	Fat (grams per serving)	int
6	Sodium (mg per serving)	int
7	Fiber (grams per serving)	float
8	Total Carbs (grams per serving)	float
9	Sugar (grams per serving)	int

The file input will terminate when the end of file is reached. (See below for specific C language loop syntax for getting input from a file using an "eof" loop.)

Processing: Find all cereals that meet the user's criteria. Count the number of cereals that match the criteria. (**All** criteria must be met for the cereal to be considered a match.)

Output: Output information for the cereals that match the criteria. Do not store the cereal information in an array. Print the information for a matching cereal immediately.

Align the output as shown below. (Some sample output is shown on the next page. The numbers shown are not the actual values. They're just to show how the output is to be formatted.)

Hint: The longest cereal name is 23 characters and the longest company name is 11. Make sure you leave enough space.

DO NOT USE the tab character \t in your printf statements.

Example Output:

```
Here are the cereals that match your criteria:

Kelloggs       NutriGrainWheat          90 calories
                 - protein               3 grams
                 - sugar                 2 grams

Quaker         HoneyGrahamOhs          120 calories
                 - protein               1 gram
                 - sugar                11 grams

... output will continue for the rest of the cereals that match ...

10 cereals match your criteria.
```

***You may use char arrays for cereal name and for company. Other than that, no additional arrays. There is a penalty in the rubric.

***Do not write functions other than main. There is a penalty in the rubric.

17. Save the program. Compile

```
gcc cereal.c -o cereal
```

18. Debug if needed and recompile. Run and test the program.

```
./cereal
```

Submit cereal.c on Canvas.

The program you submit must compile with the gcc compiler
on the CS computers in SH 118 or 118B.
Any program that contains syntax errors will receive a grade of zero.