Due by classtime January 28th

Exercise:

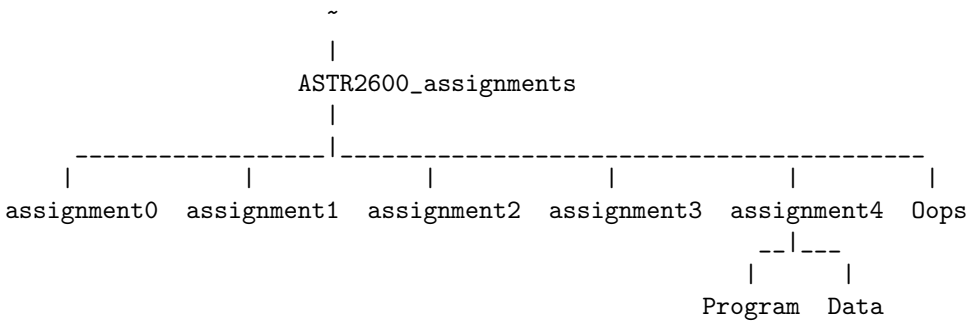
Ex 1.0: UNIX file & directory management

This has you create a bunch of directories in a given tree structure, create a file, copy it around and do some deletions. You'll need to use `cd` a lot and `pwd` to verify that you're in the right directory. I have you verify the results frequently by using `ls`.

I show you exactly how to do many of the steps, but after a while I stop showing all the details because by then I think you should know how to do it without me showing you. For each line that tells you to do something, you should read the next couple lines of instruction before doing it. Those lines may show you how or just give a little further explanation. (Don't be in a hurry to follow the command "Cut off your foot" before reading the next line, which says "of rope".) Remember, UNIX does care about upper & lower case letters.

Since this is UNIX and not IDL, there is no journal file record of this. However, there is a `history` file. If you're already familiar with UNIX, scan these steps to make sure you're familiar with them all, and then make sure you set up the directory structure as described at the end.

You will use the UNIX `mkdir` command to create the following directory tree structure in your home directory.



So from your home directory, do these commands to change the current directory to the `ASTR2600_assignments` directory you created as part of the git setup.

First, use `ls` to check and make sure the directory is there:

```
$ ls
```

Then, assuming it is, change to that directory:

```
$ cd ASTR2600_assignments
```

Then create the assignment subdirectories in this directory:

```
mkdir assignment0
mkdir assignment1
mkdir assignment2
mkdir assignment3
mkdir assignment4
```

(This goes faster if you use the up-arrow to retrieve the previous command and then edit the assignment number.)

Use `ls` to verify that they are there:

```
$ ls
```

(You should see the 5 directories listed.)

Then change into the `assignment4` directory:
`$ cd assignment4`

Create the Program and Data directories in there. (You should know how now.) Use `ls` to verify that they are there. Return up to the ASTR directory:
`$ cd ..`

Verify that you are in your `ASTR2600_assignments` directory:
`$ pwd`

Create the `Oops` directory under the `ASTR2600_assignments` directory. Use the gedit text editor to create a text file called "`notes.txt`" in your `ASTR2600_assignments/assignment1` directory. The ampersand (&) means "run in the background" so that you can continue to use the command line after running the editor.

```
$ cd assignment1
$ gedit notes.txt &
```

Put a couple (meaningless) lines of text in it and exit the editor. Use `ls` to see that it is in that directory. Use `cat` to print the file to the screen

```
$ cat notes.txt
```

Use `cp` (3 times) to copy this file to your `ASTR2600_assignments/assignment0`, `assignment2` and `Oops` directories You are in the `assignment1` directory so the destination is up to the parent (`..`) and then down into the destination directory, e.g.

```
$ cp notes.txt ../assignment0
```

Use the up arrow to retrieve this command and then edit the destination path for the other directories. Use `ls` to verify that there are copies of this file in each directory. You can do this from the current `assignment1` directory by using the parent (..) in the path, but at this point you might want to `cd` up to the `ASTR2600_assignments` directory so that you don't have to do that each time:

```
$ cd ..
$ ls assignment0
```

Use `cat` to print out the version in the `Oops` directory and see that it is the same.

```
$ cat Oops/notes.txt
```

Use `mv` to rename the original version of the file in the `assignment1` directory to `junk.txt`

```
$ cd assignment1
$ mv notes.txt junk.txt
```

Verify with `ls`.

Use `rm` to remove the file from the `assignment2` directory.

```
$ cd ../assignment2
$ rm notes.txt
```

Verify with `ls`.

Attempt to use `rmdir` to remove the `Oops` directory (It should not work)

```
$ cd ..
$ rmdir Oops
```

`rmdir` will only remove empty directories. Verify that it didn't work with `ls`

Use `rm` to remove the copy of notes.txt from the `Oops` directory.
`$ rm Oops/notes.txt`

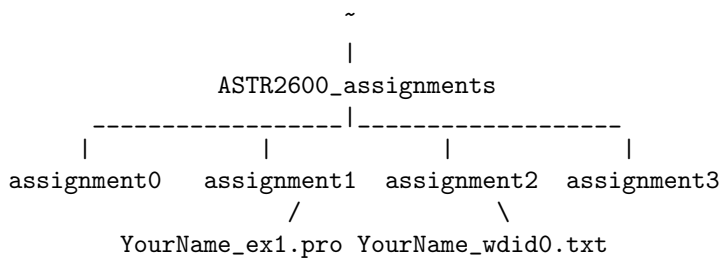
Verify with `ls`.
Use `rmdir` to remove the `Oops` directory. (Now it works because `Oops` is empty)
Use `ls` to verify.
Use `rmdir` to try and remove the `assignment4` directory. (It should not work)
Use `rm -r` to remove the `assignment4` directory.

```
rm -r assignment4
```

The `-r` stands for "recursive" and it means that `rm` should be applied to each file & directory under the one you're trying to remove and any subdirectories under that, and any under those, etc.

Verify that `assignment4` (and so also `Program` and `Data`) directories are gone.

OK, that's enough playing around with basic "files in directories" stuff. Now we'll do something useful: Use `mv` to move your assignment 0 files (`YourName_ex0.pro` and `YourName_wdid0.txt`) from your home directory (`~`) to your `assignment0` directory. Remove the copy of `notes.txt` from the `assignment0` directory. Use `ls` to verify that your directory structure now looks like this:



From the `ASTR2600_assignments` directory, use `ls -l` to check the permissions on all your `assignment#` directories. If necessary, change the permissions on all those directories so that only you, the "user" can read or execute them, i.e., take away r, w and x permissions from others (o) and group (g):

```
chmod o-rwx assignment*
chmod g-rwx assignment*
```

Use `ls -l` to check the permission of your files in your `assignment0` directory. If necessary, change the permission so that "others" can read the file, e.g.

```
chmod o+r YourName_ex0.pro
```

For the rest of the semester, you should always do your assignments from inside your corresponding assignment directory. This means that starting with assignment 4, you'll need to create new directories to hold those assignments.

You can't add a directory to `git` unless it has a file in it. Once you create the first file in a directory for an assignment, though, you can add it by doing, for example:

```
$ git add assignment1/
```

Then, you need to commit and push:

```
$ git commit
$ git push
```

IDL 1-dimensional Arrays and Basic Plotting

(As with all subsequent assignments, first `cd` into the appropriate directory for the assignment. In this case, your `assignment1` directory.) Get into IDL and type each courier font line exactly as it appears in the exercises. You do not have to type the "comment" consisting of the semi-colon and all text after it. Pay attention to the output. It's useful to type in your own comment labeling the exercise parts, e.g.,

```
; Exercise 1.8
```

Open a journal file in your `assignment1` directory called "`YourName_ex1.pro`"

Do the following exercises from the book:

- Exercise 1.8: Basic array definition arithmetic
- Exercise 1.9: Basic plotting
- Exercise 1.10: Automatic "coordinate array" generation (fill array with index values)
- Exercise 1.11: Planck Blackbody curve

Close the journal file with:

```
journal    ; you can close your exercise journal file now.
```

Turn in by `git add`ing the journal file of the exercise: `YourName_ex1.pro`

As always, if you decided to break the exercise into multiple journal files, they all need separate filenames (including the exercise part, e.g. `YourName_ex1_1.11.pro` and, of course, you need to turn in all files.

What does it do?

Using `gedit`, answer Whuduzitdo 1.8 through 1.10 in a file called `YourName_wdid1.txt`.

Reminder of how to start gedit:
At the bash `$` prompt (not the IDL `IDL>` prompt) type

```
$ gedit YourName_wdid1.txt &
```

When you are done, you can click the Save button to save the file. You can then exit `gedit` by clicking on the X in the upper right corner or select Quit under the File menu.

Turn in the text file containing the Whuduzitdo: `YourName_wdid1.txt`

Homework 1.0 Planck Blackbody equation

Graded Homework: Due by midnight, January 30th

Begin a new journal file called `YourName_hw1.0.pro`

Answer the questions in the homework by writing comments that will then show up in your journal file, e.g.,

```
; 1.0.0:  The overall area under the curve does sumpin' sumpin' as temperature
;   increases
```

Close the journal file.

Turn in a journal file of the homework:
`YourName_hw1.0.pro`