Exercises-Paths

In this exercise set, you will practice using paths. You will go to a directory in the filesystem, list it recursively, then write commands to perform certain tasks, only changing directory as instructed. This is practice in using listings and in using relative paths with a directory component (i.e., to refer to data that is near but not in your current directory)

Some of you will find it very difficult to work with a recursive listing. If you need to draw a tree from the listing, go ahead. It may also help to use the **`tree -F`** command to show the structure graphically. You should, however, be able to figure this out from a recursive listing, as that is what you will be tested on.

It is helpful to open two windows to hills for this exercise. Display the tree structure in one and use the other to do the exercise. This will save you from having to scroll back to the recursive listing constantly.

Part One - Accessing data in an existing structure using a recursive listing

Description:

Go to the existing structure **`/pub/cs/gboyd/cs160a/paths/jmoraz`** on hills. List the area recursively. Then execute commands to do the following (In the listing you will notice that each piece of data has a unique name. The instructions below tell you to access the piece of data using its name. You will have to figure out the path from the listing.)

To begin, you should be in the directory **`jmoraz`**. Stay in that directory until you are told to change directory.

1. List the **`asmt02`** directory recursively.
2. Display the contents of the file **`cc`**.
3. Display the contents of the file **`family.jpg`**. (Normally you wouldn't display a JPEG on a terminal, but this is really just a text file)
4. Change directory to the **`public_html`** directory. Stay here until you are told to change directory.
5. List the **`asmt02`** directory (not recursively this time)
6. Display the contents of the **`me.jpg`** file. (Again, this is really a text file, not a JPEG)
7. Display the contents of the **`index.html`** file.
8. List the directory **`lib`**.
9. Change directory to the **`bin`** directory.
10. List the directory **`script_files`**
11. Display the contents of the **`me.jpg`** file
12. Unfortunately, the recursive listing you used at the beginning may not have been quite accurate. The structure contained a *hidden file*. Can you find it?

As you can see, using relative paths takes some practice. Repeat this exercise until you have it cold, and until you can do it with only the recursive listing.

Check your work in the *Answers* section before you continue.

Part Two - Creating a directory structure and populating it with files

Preparation

To begin, create a directory named **`classes`** in your home directory on hills and connect to it. Then copy a set of files to it using the command below (note the period. It is very important.)

```
cp /pub/cs/gboyd/cs160a/paths/classes/*  .
```

Procedure:

Using the files in **`classes`**, create a structure to match the recursive listing on the next page. Begin by creating the empty directories in their correct orientation. Then move (using **`mv`**) the files into the correct directories. Compare the final results to the listing to verify it.

This document was produced with free software: OpenOffice.org on linux.

One possible set of commands is in the Answers section.

You need to get the structure created correctly to do the next part, so work on this part until it is correct. If you get stuck and want to start over, see the end of the Answers section for instructions on removing the entire structure. Then start again.

```
[gboyd@unixguy4 classes]$ ls -RF
.:
c++/  english/  java/  physics/

./c++:
notes/     progs/

./c++/notes:
c++_lecturenotes.odt  compilation

./c++/progs:
a.out*     hello.cpp  test.cpp

./english:
asmts/     notes/

./english/asmts:
composition1.odt

./english/notes:
stylenotes.odt

./java:
notes/     progs/

./java/notes:
java-syntax

./java/progs:
classes/  HelloWorld.java

./java/progs/classes:
HelloWorld.class

./physics:
notes/   grades

./physics/notes:
physics_labnotes.odt
```

Part Three - Reorganizing the resulting structure

Preparation

Before you start this part, you must have completed the structure from Part Two correctly. If it is not exact, this part will not work. Check it once more and verify that you have done it correctly.

You may want to do this part several times, and it would be a pain to have to recreate the starting tree

This document was produced with free software: OpenOffice.org on linux.

from Part Two each time, so we will make a copy of the entire tree first. Go to one directory above your **classes** directory (this should be your home directory), and execute the following command:

```
cp -r classes classes.sv
```

You now have a copy of the entire **classes** tree named **classes.sv** At the end of the answers for this part you will see instructions about how to delete your **classes** tree and recreate it from this saved copy to start over.

Then connect to the **classes** directory to start.

Procedure

Each of the following describes a change you should make to the tree. Some items have a suggested change. Complete the change to the tree. If there is a suggested change, analyze it and indicate whether it is correct. If it is not correct, explain what is wrong and fix it!

Make sure you are in the **classes** directory when you start. All of the instructions refer to data from that directory. For the first run of this set of commands, stay in the **classes** directory. Further instructions will appear at the end of this exercise.

1. You need the **english** directory to be capitalized: **English**. Fix it.
2. Your grades in physics have been declining. You do not want someone who sees your data to read your grades, so you want to make the **grades** file hidden, keeping it in the physics directory. Here is a suggested command: **mv physics/grades .grades**
   Does the command function correctly? If not, explain, and correct the command.
3. You want to start a new assignment in C++ based on your **hello.cpp** program. Make a copy of **hello.cpp** and name it **prog1.cpp**
4. You have found that having your java class files in a different directory than the source files causes problems. Move the class file to the same directory as the java program and delete the separate directory. Here is a suggested set of commands. Do the commands function correctly? If not, explain, and correct them:

   ```
   mv java/progs/classes/HelloWorld.class java/progs/HelloWorld.class
   rmdir java/progs/classes
   ```

5. The command in 4. was long. Without changing directory from **classes**, can you shorten the command?
6. The name of your physics notes file is too long. Rename it to just **labnotes.odt**
7. You can never remember how to compile your C++ programs when you are working on them, and the notes on compilation are not in the same directory! Move the **compilation** notes to the same directory as your C++ programs.

You should repeat this part twice - once when staying in your **classes** directory, and once when you change directory so as to make your commands shorter. In the second run, try to move to the next starting directory in a single command. (The Answers for this part has answers for each run)

Answers

Part One

1. **ls -R -F cs160a/asmt02**
2. **cat cs160a/asmt02/usr/bin/CC**
3. **cat public_html/images/family.jpg**
4. **cd public_html**
5. **ls -F ../cs160a/asmt02**
6. **cat images/me.jpg**

This document was produced with free software: OpenOffice.org on linux.

7. **cat index.html**
8. **ls ../cs160a/asmt02/usr/lib**
9. **cd ../cs160a/asmt02/usr/bin**
10. **ls -F ../../../asmt01/script_files (the directory is empty)**
11. **cat ../../../../public_html/images/me.jpg**
12. go back to the **jmoraz** directory and add the **-a** option to the recursive listing. You will see a single hidden file **cs160a/asmt01/.script1**

Part Two

Start by creating the directories. There are 12 of them, so this can be tedious. First, let's do it using brute force:

```
mkdir c++ english java physics
mkdir c++/notes c++/progs
mkdir english/asmts english/notes
mkdir java/notes java/progs java/progs/classes
mkdir physics/notes
```

You can make use of **mkdir -p** to shorten this by just specifying the longest paths

```
mkdir -p c++/notes c++/progs
mkdir -p english/asmts english/notes
mkdir -p java/notes java/progs/classes
mkdir -p physics/notes
```

You can add wildcards and the brace operator (both of which we learn about next week) to shorten it a lot:

```
mkdir -p c++/progs english/asmts java/progs/classes {c++,java,physics,english}/notes
```

Once the directories are made, we must move the files into the correct location. Again, this is tedious, but we can make use of the ability of mv to copy multiple files into a target directory:

mv file1 file2 .... dir

```
mv c++_lecturenotes.odt compilation c++/notes
mv a.out hello.cpp test.cpp c++/progs
mv composition1.odt english/asmts
mv stylenotes.odt english/notes
mv java-syntax java/notes
mv HelloWorld.java java/progs
mv HelloWorld.class java/progs/classes
mv grades physics
mv physics_labnotes.odt physics/notes
```

If you need to start over, you must use a *recursive remove*. This is one of the most dangerous commands on Linux. A mistake when using recursive remove can be very costly - remember, there is no recycling bin to retrieve deleted files from! You should NEVER use the -r (recursive) option with rm unless you are sure you are correct!If you need to start over, go to one level *above* your **classes** directory and use the command

```
rm -r classes
```


This document was produced with free software: OpenOffice.org on linux.

Part Three

We assume you have made your backup copy of **classes** named **classes.sv** so you can restart this part if you need to.

1. **mv english English** (this is just a rename operation. easy)
2. The command **mv physics/grades .grades** has a problem. Each relative path on the commandline is independent. This means the path **.grades** is in the current directory! We want the **.grades** file to remain in the **physics** directory, so we must use the command
   mv physics/grades physics/.grades
3. **cp c++/progs/hello.cpp c++/progs/prog1.cpp**
4. The command is okay.
5. You can shorten the first command by not specifying the name of the file in the target:
   ```
   mv java/progs/classes/HelloWorld.class java/progs
   rmdir java/progs/classes
   ```
6. **mv physics/notes/physics_labnotes.odt physics/notes/labnotes.odt** (Note that it is easier to do some of these commands after changing directory appropriately)
7. **mv c++/notes/compilation c++/progs**

You will now remove your classes directory and copy it from the saved version. This requires us to a *recursive remove*. Again, this is one of the most dangerous commands on Linux. A mistake when using recursive remove can be very costly - remember, there is no recycling bin to retrieve deleted files from! You should NEVER use the -r (recursive) option with rm unless you are sure you are correct!

First, **cd** above the **classes** directory. Make sure you have a **classes.sv** directory before you proceed!

```
rm -r classes
cp -r classes.sv classes
```

You now have a fresh copy of **classes** to redo this part of the exercise and using **cd** commands to shorten your commands. Let's see how fast we can do it:

```
cd classes
mv english English
cd physics
mv grades .grades
cd ../c++/progs
cp hello.cpp prog1.cpp
cd ../../java/progs
mv classes/HelloWorld.class .
rmdir classes
cd ../../physics/notes
mv physics_labnotes.odt labnotes.odt
cd ../../c++/notes
mv compilation ../progs
```

If you compare the two sets of command (using **cd** and not), you will probably see that using the **cd** command requires more typing, but it is harder to make a mistake when copying and renaming files! The choice is up to you!

This document was produced with free software: OpenOffice.org on linux.