CSCI 204 – Introduction to Computer Science II

Lab 3 – Recursion

1 Objectives

In this lab, you will implement recursive directory listing on UNIX file system and handle exceptions.

2 Getting Started

Begin by opening a terminal window and creating a directory for this lab.

```
cd ~/
cd csci204/
cd labs
mkdir lab03
cd lab03
```

Then copy all files, including the skeleton program `dirs.py`, from the course Linux directory using the following command.

```
cp ~csci204/2017-fall/student/labs/lab03/* .
```

The a few text files other than the Python program in the directory can help you practice the diff command as described below.

Linux command `diff`

Linux provides many useful commands for programmers. The command `diff` is one of such commands. The diff command compares two text files and prints out the differences between the two. The diff command can be used in many ways. For example, you can use the command to see if the output of your program is same, or very similar to the required output. Please read this blog post to understand the general meaning of diff output.

https://unix.stackexchange.com/questions/81998/understanding-of-diff-output

After reading the post, try the diff command among the four text files you just copied. For example

```
diff file1.txt file2.txt
diff file1.txt file3.txt
diff file2.txt file4.txt
```

Make sure you can explain the output of these comparisons.

3 Recursively Listing Files

Recursive algorithms are natural for solutions to problems with hierarchical structure. An example problem is listing all the files in a directory and all of its subdirectories. Since the UNIX file system is hierarchical, we should immediately think of using a recursive approach.

For this part of the lab, you will write a Python program to list all of the files in a directory and, recursively, in all of its subdirectories. You can see this in action with the following command.

```
ls -R ~csci204/public_html/2016-fall/testpages/
```

If you have maintained a working directory for this course, you can try to list it under your home directory by

```
ls –R ~/csci204
```

For this part of the lab, you will write a Python program to list all of the files in a directory and, recursively, in all of its subdirectories, in a similar fashion by the `ls -R` command.

Read the program contained within `dirs.py` to get an idea what is involved in the program. Then try the following commands with the program using Linux Terminal. (If you are running your program from Idle, read the comments at the end of the program and revise it properly. Note that Idle may not work with paths that contain tilde)

```
python dirs.py dirs.py
python dirs.py ../
python dirs.py ~/csci204
```

Observe the behavior of the program as it stands before you make any modifications. `dirs.py` is a Python program that lists the names of the files and sub-directories in a directory, but does *not* recursively list the files under any subdirectories. You'll notice that it checks to see if there are any command-line arguments passed to it. If no argument is given, the program prints a usage message, asking the user to supply an argument.

You can also run the Linux command to list the directories recursively. Try the following command:

```
ls –R ~/csci204
ls –R ../
```

You are to modify the `dirs.py` class to *recursively* print the names of all the files in all subdirectories. Here are some details.

1. You must use a recursive solution.
2. File names should be printed one per line.
3. Just *before* any recursive calls, print out "-- Entering *name*" where the program should fill in the directory name. [The method `os.path.basename()` will extract the directory name from the path argument. Note that if the path ends with a '/' character, this method will return the empty string, so it is worth putting in a check to remove the trailing '/'.]
4. Just *after* any recursive calls, print out "-- Leaving *name*" where the program should fill in the directory name.
5. Python provides a few useful methods and functions in the `os` module and the `os.path` module that you need to use (read the relevant Python online documents for details)
    a. import the `os` and `os.path` modules. Since you do not know what parts of them you want, use a `*` to get all the parts

    ```
    from os import *
    from os.path import *
    ```

    b. The `os` module has a function `listdir()` that returns a list of the content in the directory. The list contains both files and subdirectories. Read this link to understand how it works: `http://www.tutorialspoint.com/python/os_listdir.htm`

c. While a directory contains a list of files or subdirectories, a file doesn't have any directories. Thus you can't use the function `listdir()` to generate a list if the parameter to the function is a file itself. To check if an object is a file or a directory, you can use the function `isfile()` and `isdir()` in the `os.path` module, respectively.

   Not sure how they work? Search for them online! Lots of people use them and there is also official python documentation. Search for `os.path.isfile()`

   TIP: when checking if an object is a file or a directory, the `isfile()` and `isdir()` functions require the parameter to be a full, absolute path. A relative path is something similar to the following:

   `~csci204/2017-spring/`, or `~/message.txt` or `../readme.txt`

   while an absolute path is a complete path from the root directory, such as:

   `/home/accounts/student/s/sam023/hello.txt`

   The problem is that `listdir()` doesn't give you an absolute path (ugh). It gives you the filename or directory name. Good news is that `os.path` has you covered:

   - `os.path.abspath(curr_dir_name)`- will take the name of a directory as input and convert it to the absolute path
   - `os.path.join(absolute_path, file_or_dir_name)` – willintelligently combine an absolute path with your file or directory name

d. To check your program's output, compare your result with that of the following UNIX command:

```
ls -R ~csci204/public_html/2016-fall/testpages/
```

where the `-R` option is used to recursively list subdirectories encountered. Your listing should contain the same folders and files listed (albeit in a different format, or a different order since Python doesn't list files in a fixed order.)

Sample output:

```
--Entering testpages
grading.html
index.html
level2
readme.txt
test2.html
test.html
--Entering level2
page1.html
page2.html
--Leaving level2
--Leaving testpages
```

Run your program using at least two different directories each of which has files and a subdirectory (or subdirectories). You must use `~csci204/public_html/2017-fall/testpages/`

(remember to deal with the trailing '/') as one of the test directories. Make sure your program is well commented. **Save and upload your program to Moodle.**

Get File Statistics When Traversing Directories

If you list files on UNIX using commands such as `ls -l` or `ls -lt`, you will find other statistics about the files are printed, including dates the files are created and the size of the files. Try these commands in your own home directory.

`ls -l ~/` and `ls -lt ~/`

For now, we will just concentrate on two pieces of information, the size of a file and the date when a file is last modified. These are the two middle columns in the above listing. For example, the first file `grading.html` in the `testpages` directory was created on September 7, 2016 with a size of 667 bytes (or 667 characters since this is a text file).

Your task

Revise the program you just finished so that the new program can count the total number of bytes all the files used on the disk, list the maximum and minimum sizes, as well the oldest and newest time of the files created in the directories your program is visiting.

The basic logic of the program is to compute the maximum and minimum sizes, as well as the oldest and newest time stamp of all files when visiting each file. Once you find the size and the date of creation of a file, you should be able to compute the max and min in a collection of those values.

Make a copy of your existing program, name the new program `dirattrib.py` (directory attributes). Modify the program `dirattrib.py` so that it can accomplish the following tasks.

Develop a FileStats class:

1. Since you are going to be creating a series of statistics, create a new `FileStats` Python class with the following data attributes: `maxSize, minSize, oldestTime, newestTime`.
2. You should define three methods within `FileStats`.
    a. The constructor where you define your data attributes
    b. The `printResults(self)` method which prints the statistics in the following format.

```
maximum size of files : 667
minimum size of files : 53
oldest time : wed Sep 7 14:45:55 2016
newest time : Wed Sep 5 14:55:35 2016
```

    c. An `update(self, filename)` function to carry out the task of retrieving the file statistics and updating the values held in your `FileStats` object.

3. To retrieve file statistics, Python's os module provides a function called lstat(). The function lstat() returns an object. Among other pieces of information, the returned object contains the size of the file and the time stamp when the file is last modified. These two data members are called st_size and st_mtime. You can use them to collect the required statistics. Read the relevant Python document to make sure you understand how to access these values.
4. The number of bytes a file takes is an integer, so you can print the maximum size and minimum size directly when the directory traversal is completed. However, the time when a file is last modified is the number of seconds since epoch (January 1, 1970), which is a huge value and in general it won't make sense for a human being to read.

    For example: a time stamp for March 16, 2011 about 1 o'clock in the afternoon would read something like: *1300296729.762571* seconds. The `time` module in Python provides a function called `ctime()` that converts a value in seconds to a human-readable time such as Wed Mar 16 13:32:09 2011. While you would use the time value directly to find minimum and maximum, you must use `ctime()` function to print the final oldest and newest time.

    [Remember to import the `time` package.]

Other Modifications

- Modify the `listDir()` method so that it takes a `FileStats` object as a parameter. We ask that the `listDir()` method recursively call itself, passing your updated `FileStats` object each time.
- Initialize a `FileStats` object in the `main()` method before calling the `listDir()` method. During the creation of your `FileStats` object, you'll be initializing its data attributes. What should those initial values be?

  Here are two hints.

  Python provides a `sys.maxsize` as a reasonable maximum integer value. (Note that there is really no limit on the values of numbers in Python.)

  File size on UNIX system is measured by bytes and the time on UNIX is measured as the number of seconds since January 1, 1970. Both are non-negative values.

The Final Product

Show your program works correctly in at least two directory listings, each of which must have multiple levels of directories. The first must be the directory

```
~csci204/public_html/2017-fall/testpages/
```

The test run should show the following values:

```
maximum size of files : 667
minimum size of files : 53
oldest time : Wed Sep 7 14:45:14 2016
newest time : Wed Sep 7 14:55:58 2016
```

Save and upload to Moodle your newly completed program (`dirattribobj.py`)

Congratulations! You just completed the lab exercises!