FILE PATHS; BISECTION AND NEWTON'S METHODS FOR FINDING ROOTS

VIVEK MOHAN MALLICK

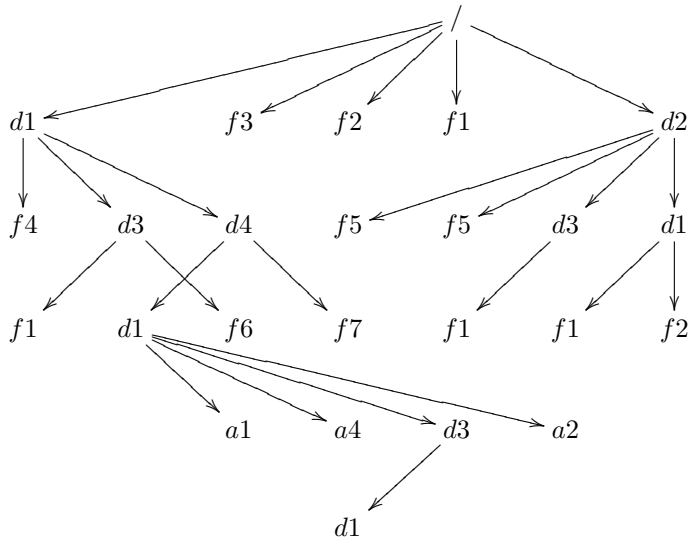
1. File paths

Most of the operating systems which we use on PCs stores all our data in a filesystem. Though there are a lot of different types of filesystems : FAT, NTFS, ext2, ext3, ext4, reiserfs etc. However for the end users like us the structure looks the same. The only difference is how the data is stored in the disk. In this lecture we'll just be considering the structure of a filesystem and how to access/write particular files from both the operating system and python.

All filesystems store files in the form of a tree. There is a root folder or directory. The words "folder" and "directory" are synonyms. In this lecture I shall use the word "folder" as it means less typing for me.

The root folder can contain files or other (sub)folders. Each (sub)folder can further contain files or ((sub)sub)folders and so on. And soon we'd be tired of writing 'sub's and so we'll write folder for $(\text{sub})^n$folders.

We have a picture like



where / denotes the root folder, rest are all file names except for the ones starting with d which are directories. Note that filenames and directory names can be anything subject to a few operating system dependent rules. An arrow of the form $d? \to f$ denotes that that the file or folder f is contained in the directory $d?$.

Date: April 1, 2014.

1.1. **Unix like systems : Linux, Sun, BSD and Mac OS.** In all these Unix like systems, there is one filesystem tree. If one attaches another media, like a CD-ROM, usb stick or an external hard drive (or if you are ancient, a floppy disk), it gets *mounted* as a part of the filesystem of the original operating system under `/cdrom/`, `/media/` or `/mnt/` depending upon the configuration.

1.1.1. *Absolute paths.* Absolute paths are the path to the file or folder you want to access starting from the root. Look at the figure above. Suppose we want to write the absolute path to the file $a4$ in line 5 of the above figure. The folders you have to traverse to go from / to $a4$ are $/ \to d1 \to d4 \to d1 \to a4$. The path is then written as `/d1/d4/d1/a4`. Similarly convince yourself that the absolute path to the third file named $f1$ in the fourth line is `/d2/d1/f1`. Also note that the second file named $f1$ in the fourth line has path `/d2/d3/f1` is different from both the first and third $f1$. What is the path to the first $f1$ on the fourth line?

Our example has only 4 levels. But in reality file structures can get pretty complicated and writing absolute paths can be difficult. To make life easier we can also access files by their *relative path names*.

1.1.2. *Relative paths.* When we work on a shell or a computer we usually start at some folder called the present working directory. You can see what your present working directory is by typing (without the $)

```
$ pwd
```

on the terminal. One can change the present working directory using the **c**hange **d**irectory `cd` command. For the sake of example, suppose you are on directory $d4$ in line 3 of the above picture, and you want to change to the directory $d1$ in the next line. You do:

```
$ pwd
/d1/d4
$ cd d1
$ pwd
/d1/d4/d1
```

To go up one level, you type

```
$ pwd
/d1/d4/d1
$ cd ..
$ pwd
/d1/d4
```

Here `d1` and `..` were relative pathnames. One can think of a relative path to be the dirctory formed by joining the output of pwd and the relative path with a `/` in between. Try the following on a terminal and try to explain what happens.

```
$ cd ~
$ pwd
```

```
<Make a note of the output suppose it is /home/u20130001>
$ mkdir test
$ cd /home/u20130001/test/../test
$ pwd
```

Here `/home/u20130001` in line 4 should be replaced by the output of the first `pwd` command. What is the output of the last `pwd` command? After doing this experiment you can remove the `test` folder by typing

```
$ cd ~
$ rmdir test
```

1.2. **Windows.** In Windows, each disk has its own root. For example, the main hard disk, is usually `C:`, with the usb sticks having different letters, like `F:`, `G:` etc. Rest of the file names are very similar to unix except that the forward slash `/` is replaced by backward slash \. So an absolute filename will look like `C:\Users\Python\Myfile`.