Linux Pathnames and Directories

- Hidden Filenames
  - Startup Files
  - The **.** and **..** Directory Entries
- Pathnames
  - Absolute
  - Tilde **~** in Pathnames
  - Relative
    - In ***current*** directory
    - In a ***subdirectory*** or an ***ancestor*** directory
    - In ***neither*** a subdirectory nor an ancestor directory

- Other Topics
  - Standard Directories
  - `/home` vs. *your* home directory
  - Special Uses of **/**
  - Special Uses of **.**

Hidden Filenames

- A file whose filename begins with a period **.** is a *hidden* or "invisible" file
- *ls* does not display these files – unless you use the **-a** option
- You have already seen one such file → `.forward`
- Two special hidden files **.** and **..** appear in every directory

```
$ ls  -a
.   ..   hw3   work
```

- We'll discuss them shortly

Startup Files

- The following two files are **startup files**:

  `.login` and `.bash_profile`

- Startup file contain Unix commands that are run just before you get the first shell prompt
  - These customize your Unix environment
  - They only work when placed inside your home directory
  - They can set variables which can help you with your work
- We'll talk more about this in a future class

The . and .. Directory Entries

- Every directory has at least two entries: . and ..
- When a new directory is created, these entries will always be there
  - . stands for the current directory
  - .. stands for the parent directory – the directory immediately above your current directory

```
$ pwd
/home/ckelly

$ cd ..

$ pwd
/home
```

The . and .. Directory Entries

. is most often used in two circumstances

- To *run a program* located **in** your current directory
- To *move or copy a file* **to** your current directory

```
$ ls
it244  notes.txt  work  work2

$ cd work

$ ls
bletch.txt  foo.txt

$ cp ../notes.txt  .

$ ls
bletch.txt  foo.txt  notes.txt
```

Pathnames

- Every file has a **pathname**, which is used to access the file
- A pathname has two components
  - The *name* of the file
  - A path to *reach* the file
- A pathname is like an address on a letter:
  - a name and...
  - directions to get there
- The name of the file is always at the ***end*** of a pathname
- What comes ***before*** the filename is the path to the directory that holds the file

Pathnames

- A **path** is *the list of directories* you must go through to get to the file
- When you see a slash, **/** , to the right of a name
- it means the name refers to a directory
- When a path consists of *several* directories a slash, **/** , separates the directory names
- There are two types of pathnames
  - Absolute
  - Relative

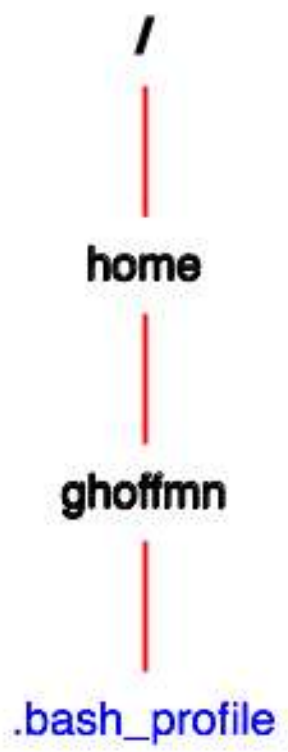
Absolute Pathnames

- The top of the filesystem is called the **root directory**
  - The root directory is represented by a single slash, **/**
  - It can stand alone or appear as the first character before a directory name
- An **absolute path** is a list of directories starting with the root directory and ending with the directory that holds the file
- When you add the filename to the end of an absolute path you have an **absolute pathname**

Absolute Pathnames

- The absolute pathname of my startup file, `.bash_profile` in my home directory is `/home/ckelly/.bash_profile`
- This means that my home directory, `ckelly`, is under the directory named `home` which is under the root directory **/**
- The advantage of an absolute pathname is that it can be used from ***any*** part of the filesystem
  - It *does not depend* on your current directory
- The disadvantage is that absolute pathnames tend to be long
  - It is easy to make a *mistake* typing an absolute pathname
- An absolute pathname **always** begins with either a slash **/** or a tilde, **~**

Absolute Pathnames: Example (User ghoffmn)



Absolute pathname: /home/ghoffmn/.bash_profile

Tilde, ~ , in Pathnames

- There is one form of absolute path that is very short
- This is the tilde character **~**
  - Tilde stands for the absolute address of your home directory
  - This means you can use tilde **~** anywhere you would normally use a path to your home directory

```
$ pwd
/home/ckelly/it244

$ cd ~

$ pwd
/home/ckelly
```

Tilde ~ in Pathnames

- When you put a tilde in front of a Unix username it stands for the home directory of that account

```
$ cd ~ckelly

$ pwd
/home/ckelly
```

Relative Pathnames

- Absolute pathnames are useful because you can use them anywhere
- But they are long and easy to mistype
- It is easier to use **relative pathnames**
- In a relative pathname the path starts from your current directory
- In an absolute pathname the path starts from the root directory, /
- The only difference between an absolute and a relative path is *where they start*

Relative Pathnames

- An address on a letter is like an absolute pathname
- If you address a letter to me at...

```
Chris Kelly
99 Boylston Street
Boston, MA 02116
USA
```

- ...then you can mail it to me anywhere in the world and it will get to me

Relative Pathnames

- If someone asked me the directions to the men's room from the Web Lab I would say

```
Go out the door
Turn left
Take you first door on the right
```

- These directions only work from the Web Lab
- This is similar to a relative path

Relative Pathnames

- ***All*** absolute pathnames start with a slash **/** or a tilde, **~**
- However, ***relative*** pathnames **never do**
- The absence of a slash **/** or a tilde **~** at the start of a pathname means it is a *relative* pathname
- As far as Unix is concerned it makes no difference whether you use and absolute or relative pathname
- Relative pathnames...
    - are more convenient, and as such...
    - are most often used

Relative Pathnames

- There are four types of relative pathnames
  1. When the file is in your **current directory**
  2. When the file is in a **subdirectory** of your current directory
  3. When the file is in a directory that is an **ancestor** of your current directory
  4. When the file is in a directory that is **neither an ancestor or descendant** of your current directory

Relative Pathnames in Your Current Directory

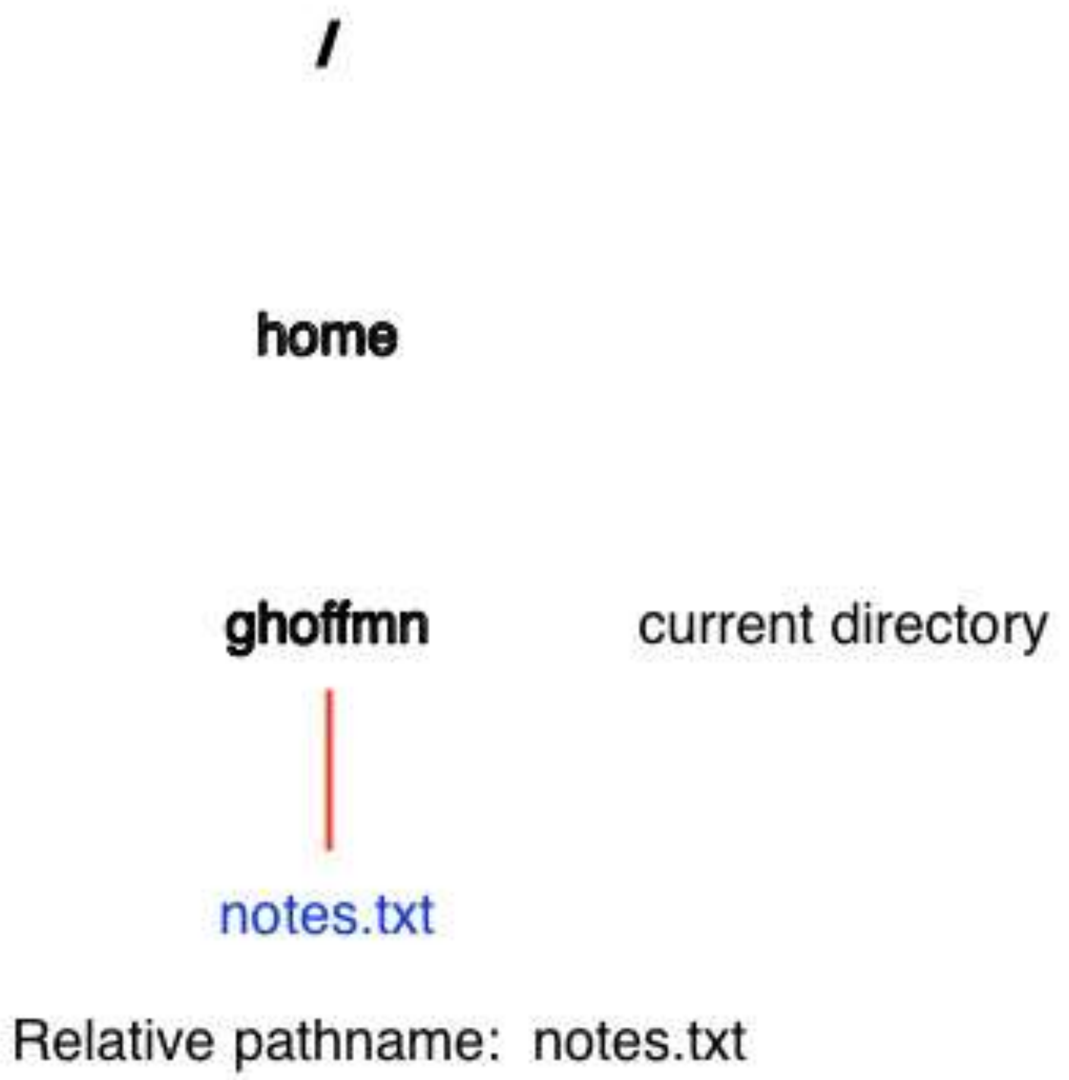
- Using a relative pathname with a file or directory *inside your current directory* is easy
- The relative pathname is simply the ***name*** of the file or directory
- The "path" part of the relative pathname is empty because the file or directory you want is already inside your current directory

```
$ ls
notes.txt  it244  work  work2

$ cat notes.txt
foo
foo
foo to you
```

Relative Pathnames in Your Current Directory

- Notice that all I need to get the file is the file name
- It's the same for *directories*



Relative Pathnames in a Subdirectory

- Things get a little more complicated when you are dealing with a file in a subdirectory
- In this case you must list every directory between your current directory and the file you want

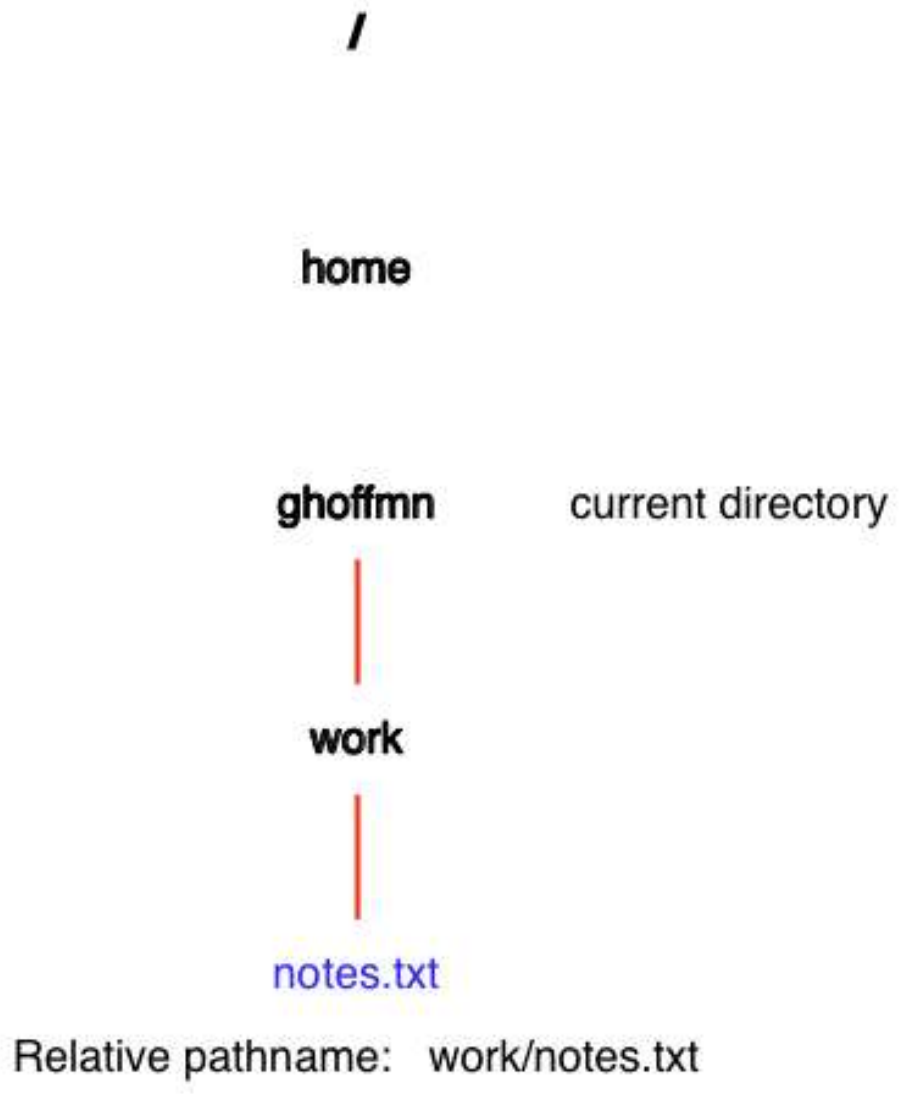
```
$ ls work
bletch.txt  foo.txt  notes.txt

$ cat work/notes.txt
foo
foo
foo to you
```

- You must use a slash **/** to separate the name of each directory from the name of the file or directory that comes after it

Relative Pathnames in a Subdirectory

- Notice that there is **nothing** no slash **/** or tilde **~** before the name of the first directory in the path



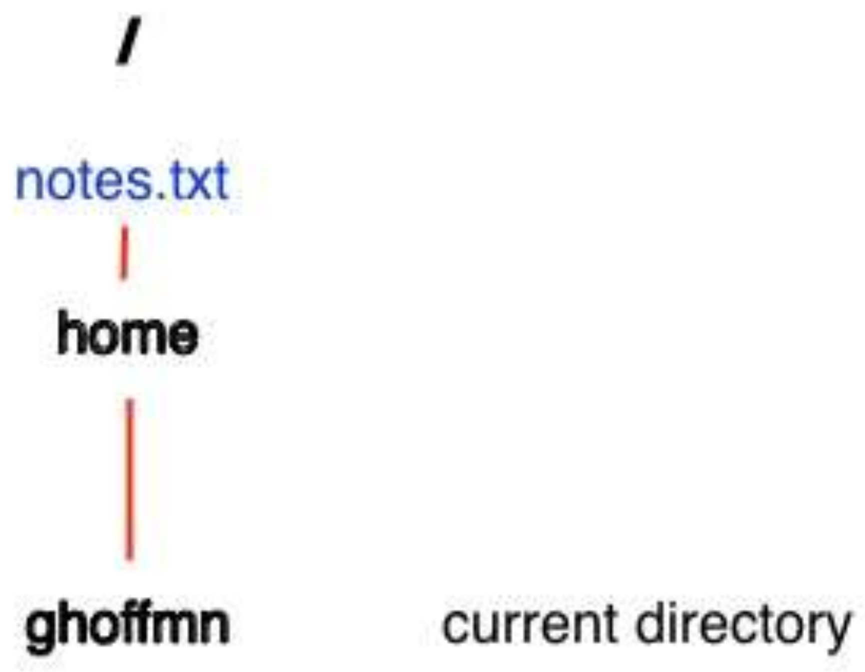
Relative Pathnames above the Current Directory

- When the file or directory is above the current directory you can't list the directory names
- Instead, you have to use the special **..** entry in each directory
- Use one **..** for each directory up the chain of directories in the path with a slash **/** between each **..**

```
$ pwd
/home/it244gh/work

$ ls ../..
it244gh  jharris  mphamman
```

Relative Pathnames above the Current Directory



Relative pathname: ../../notes.txt

Relative Pathnames Neither above Nor below the Current Directory

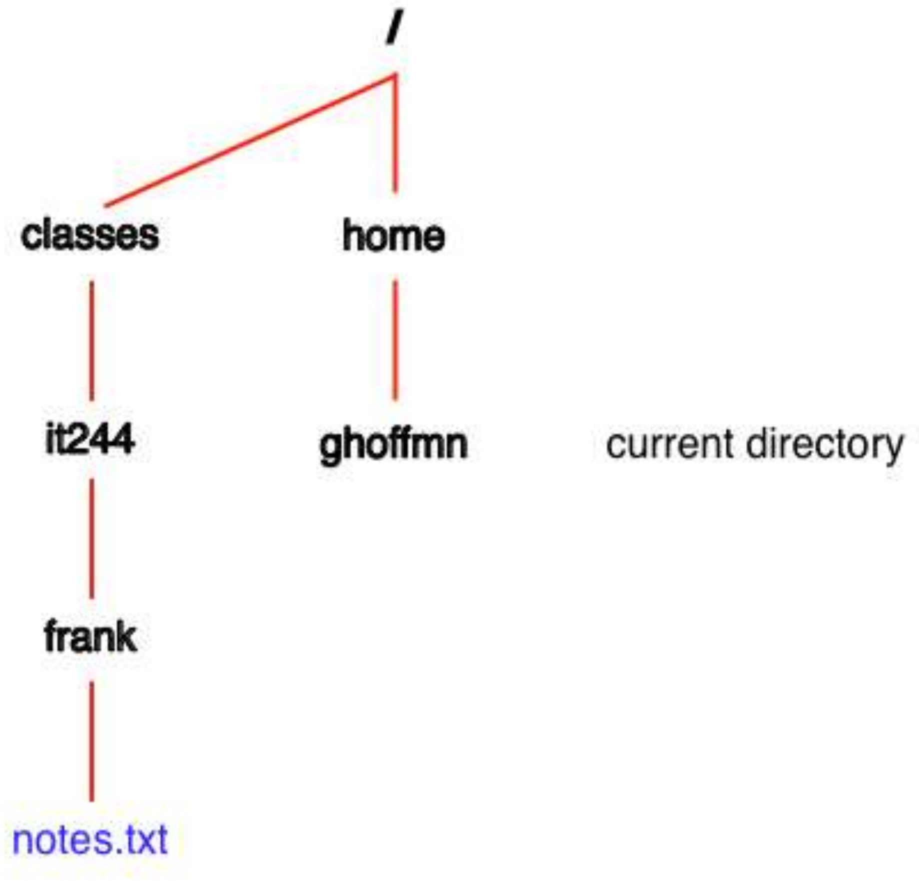
- What if the file is ***neither* above *nor* below**?
- Here you have to...
  - go **up** to a common **ancestor** directory...
  - ...and then go **down** to the directory that holds the file
- The path starts with one or more **..** and keeps going up until you get to a directory that is an ancestor of...
  - **both** your current directory...
  - ...**and** the file you are trying to reach

Relative Pathnames Neither above Nor below the Current Directory

- Once you get to the common ancestor you go down to the directory that holds the file

```
$ ls ../../../courses/it244/f16/ckelly/
alexgri   fatalaty  kiwan   neko92  rangeley  sfarah   sukhi515
cdelaney  GROUP     MAIL    neoalx  rolon     sindel   wenwu10
cs110ck   hsingh    meteos  nle     sanf5456  skhalifa ychen123
```

Relative Pathnames Neither above Nor below the Current Directory



Relative pathname: ../../classes/it244/frank/notes.txt

Standard Directories

- In the early days of Linux each distribution stored its important files in different directories
  - This made it hard to document programs
  - It also made things hard for developers working with different flavors of Linux
- Fortunately, reason has since prevailed
- We now have something called the FHS
  - It stands for ***Linux Filesystem Hierarchy Standard***
  - It has been adopted by most major Linux distributions

Standard Directories

- This standard does not tell where to put different kinds of files
- Instead, for each directory name, it says what you can find in it
- You can see a listing for the FHS on page 96 of the current version of Sobell or page 91 of the previous edition
- Most Linux distributions do not use all these directories
- But when they do, they use them for the type of files specified by the standard

Your Home Directory versus `/home`

- Most Unix accounts have a **home directory** though not all
- One of the Ubuntu commands that creates an account has an option to **not** create a home directory for that account
- On most Unix and Linux systems all home directories are contained in the directory `/home` which is directly under the **root directory** written as `/`
- On the Mac, home directories are contained in `/Users` also directly under the root directory
- Do not confuse `/home` with your home directory

The Two Uses of /

- / is used in two different ways when writing pathnames
- When / appears at the beginning of a pathname it refers to the **root directory** -- the directory at the top of the filesystem
- The root directory is the only directory that is **not** contained in another directory
- Here is an example of this use of /

```
/home
```

- This is the absolute address of `home` which holds the home directories of all users

The Two Uses of /

- When the / is **not** at the beginning of a pathname it is used a separator a character to mark the boundary between a directory and the name of a file or another directory
- Here is an example of this use of /

```
it244/hw
```

- This is the relative pathname of the `hw` directory inside the `it244` directory

The Two Uses of **/**

- A pathname can use the **/** in both ways

```
/home/ckelly
```

- Here, the first **/** means the root while the second is used to separate the `home` directory from **my** home directory, `ckelly`

The Two Special Uses of .

- Another character that has more then one use when writing pathnames is the . character
- When . appears alone, it means the current directory
- So the following command...

```
cp /home/ckelly/test.txt  .
```

- ...copies a file to the **current directory**
- When . appears as the first character in a filename or directory name, the *ls* command *will not* list these directory entries unless you use the `-a` option

The Two Special Uses of .

- These "invisible" files are used to configure Linux and some programs
- Examples are `.forward` and `.plan`
- Unlike **/** , **.** can be used as an ordinary character in a file or directory name

```
$ touch a.b.c.d....

$ ls
a.b.c.d....
```