Absolute and Relative Paths

A path is a unique location to a file or a folder in a file system of an OS. A path to a file is a combination of / and alpha-numeric characters.

Absolute Path-name

The absolute path starts from the directory root (/) and goes up to the actual object (file or directory). It contains the names of all directories that come in the middle of the directory root and the actual object. In this, name of the parent directory is written in the left.

Let's take an example, suppose a user named raman creates a directory named test in his home directory. What will the absolute path of this directory?

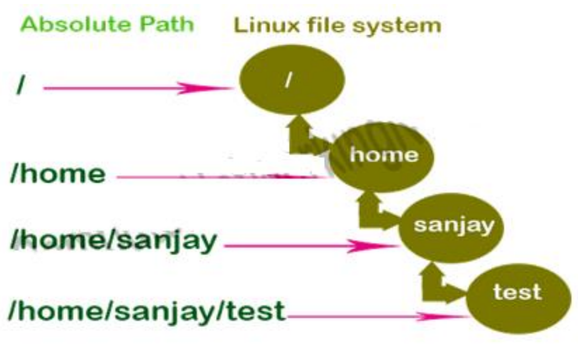
To write the absolute path of this directory, we have to start writing the path from the directory root. The directory root is written as **/** (forward slash).

After **/** (root directory), we have to write the name of the directory in which user's home directory is located. By default, Linux places user's home directory in the directory named **home**. Usually, this directory is created just under the directory root (**/**). If we write the name of the home directory just after the root directory, we get the absolute path of the home directory.

If we write the name of the user's home directory just after the absolute path of the home directory, we get the absolute path of the user's home directory. By default, Linux uses user's username to create user's home directory. In this example, username is Raman, hence the name of its home directory is also Raman.

Following the same track, if we write the name of the directory or file which is created in user's home directory just after the absolute path of the user's home directory, we get the absolute path of that directory or file.

In this example, absolute path of the directory test will be; **/root/home/raman/test**.



Key points

- First forward slash (/) in the absolute path represents the directory root. Besides this, all slashes in the path represent the directory separator.
- Besides the last name, all names in the absolute path belong to directories. Last name can belong to file or directory.
- In the absolute path, directories names are written in their hierarchy order. Parent directory's name is written in the left side.
- Absolute path does not change when we change the current directory.

To know to the absolute path of the current directory, we can use the command pwd

An absolute path is defined as specifying the location of a file or directory from the root directory(/).

To write an absolute path name:

- ➢ Start at the root directory (/) and work down.
- ➢ Write a slash (/) after every directory name (last one is optional)

For Example:

$cat abc.sql

Will work **only** if the fie **“abc.sql”** exists in your current directory. However, if this file is not present in your working directory and is present somewhere else say in /home/kt , then this command will work only if you will use it like shown below:

cat /home/kt/abc.sql

In the above example, if the first character of a pathname is /, the file’s location must be determined with respect to the root. When you have more than one / in a pathname, for each such /, you have to descend one level in the file system like in the above kt is one level below the home, and thus two levels below root.

*An **absolute path** is defined as specifying the location of a file or directory from the root directory(/). In other words, we can say that an absolute path is a complete path from start of actual file system from / directory.*

Relative path

The relative path starts from the current directory and goes up to the actual object. Relative path depends on the current directory. When we change the directory, relative path also changes. Just like the absolute path, the name of the parent directory is written in the left side. Unlike the absolute path, all slashes in the relative path represent the directory separator.

Before we take the examples of the relative path, let’s understand the special meanings of single dot and double dots used in the relative path.

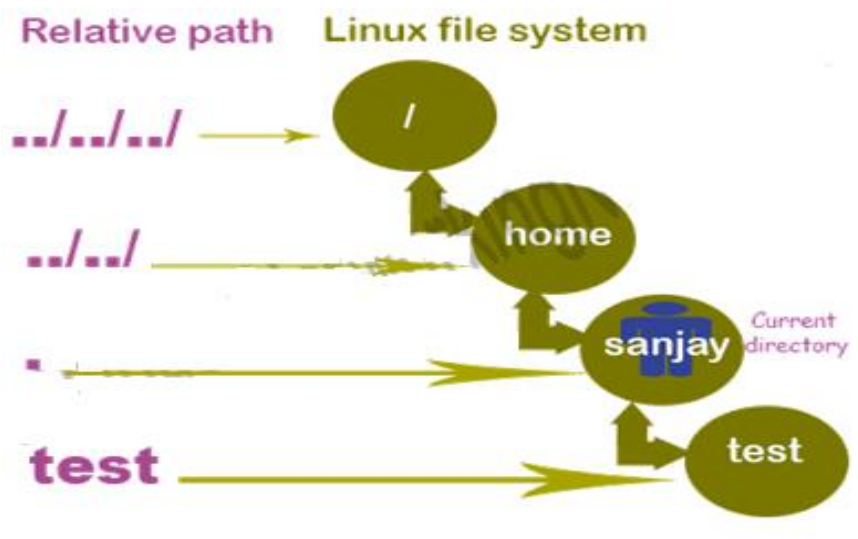
Single dot (.) and double dots (..) in Linux

In Linux, every directory contains two dots; single dot and double dots. When a directory is created, both the single dot and the double dots are also automatically created in it. By default these dots are hidden and do not show in the output of the command ls. To view these dots, we have to use option a with the command ls.

The single dot refers to the directory itself and the double dots refers to its parent directory or the directory that contains it. Shell allows us to access the current directory and the parent directory by using the single dot and the double dots respectively.

Relative path also uses these dots to represent the current directory and the parent directory respectively. With the use of these dots, we can build the relative path of any file or directory from the current directory.

The following figure shows the relative path of the directories used in the previous example.



Let’s take another example. Suppose a user want to access a file that is available two directories above in the hierarchy from his current directory. To access this file, he can use the following relative path.

```
../../file
```

Just like it, if the file is available in the three directories above in the hierarchy, he can use the following path.

```
../../../file
```

Relative path of the file or directory that is below in the hierarchy always starts with a single dot followed by a forward slash as ./. The ./ represent the current directory.

Unlike the parent directory, no symbol (dot or dots) is used for the child directory. If the file or directory is available in the directory that is below in the hierarchy, we must have to use the actual names of child directories in path.

For example, a file named abc is available in the directory named dir1 and the directory dir1 is available in the current directory, relative path of this file will be the following.

```
./dir1/abc
```

To execute a command, Shell uses current directory as the default directory. Because of this, if the target object is available in the child directory of the current directory, we can omit the leading **./** from the relative path. For example, to access the above file we can also use the following path.

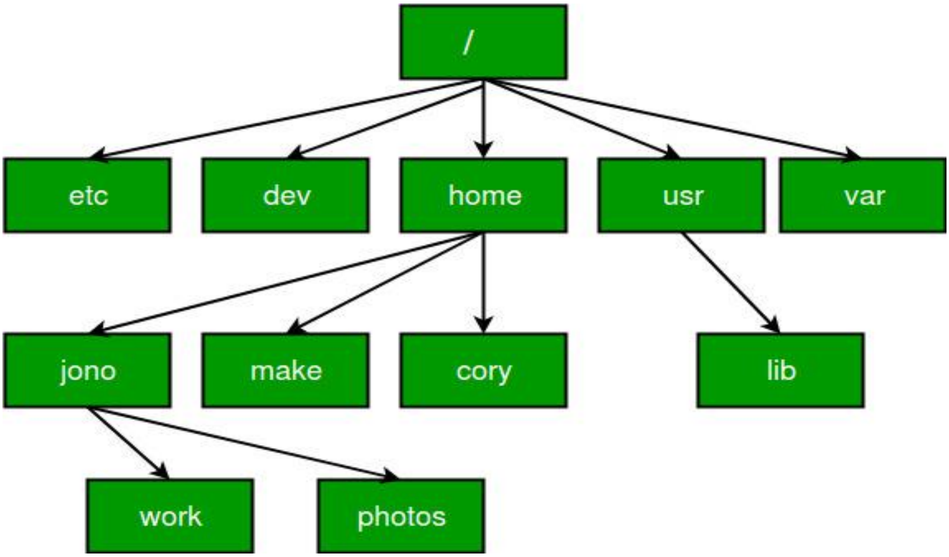
```
dir1/abc
```

In some situations, skipping current directory from the relative path makes it ambiguous. In such situations, we must have to use the full relative name including the current directory. For example, to run a script from the current directory, we must have to use the full relative name. Let's take few more examples to understand the absolute path, relative path and the use of dots in the relative path practically. Access Shell prompt and create a directory named dir1. Create a file named abc in this directory. Also, create a simple script named simple.sh in the current directory. Now run the commands listed in the following table.

Command	Description	Path
cat ./dir1/abc	Print the contents of the file abc.	Use relative path. *Include current directory*
cat dir1/abc	contents of the file	Use relative path. *Skip current directory*
cd ./dir1	Change current directory to dir1	Use relative path.
cd ..	Change current directory to the parent directory	Use relative path.
cd /home/raman/dir1	Change current directory to dir1	Use absolute path.
cp ./dir1/abc .	Copy the file abc in current directory	Use relative path.
./simple.sh	Run script from current directory	Use relative path

The relative path is defined as the path related to the present working directly(pwd). It starts at your current directory and **never starts with a /** .

To be more specific let's take a look on the below figure in which if we are looking for photos then absolute path for it will be provided as */home/jono/photos* **but assuming that we are already present in jono directory then the relative path for the same can be written as simple** *photos*.



Using . and .. in Relative Path-names

UNIX offers a shortcut in the **relative pathname**– that uses either the current or parent directory as a reference and specifies the path relative to it. A relative path name uses one of these cryptic symbols:

.(a single dot) - this represents the current directory.

..(two dots) - this represents the parent directory.

Now, what this actually means is that if we are currently in directory /home/kt/abc and now you can use **..** as an argument to **cd** to move to the parent directory /home/kt as :

```
$pwd

/home/kt/abc
```

$cd ..   ***moves one level up***

$pwd

/home/kt

NOTE: Now / when used with.. has a different meaning;instead of moving down a level,it moves one level up:

 $pwd

/home/kt/abc   ***moves two level up***

$cd ../..

$pwd

/home

Examples of Absolute and Relative Path

Suppose you are currently located in home/kt and you want to change your directory to home/kt/abc. Let's see both the absolute and relative path concepts to do this:

1. **Changing directory with relative path concept :**
2. $pwd
3. /home/kt
4. $cd abc
5. $pwd
6. /home/kt/abc
7. **Changing directory with absolute path concept:**
8. $pwd

9. /home/kt
10. $cd /home/kt/abc
11. $pwd
12. /home/kt/abc

Linux file system is built from files and directories. Files are used to store the data. Directories are used to organize the files systematically. The directory root (/) is the main directory in Linux. All directories and files are created and managed under this directory. The location of a file or directory from this directory is known as the path of that file or directory.