Working with Directories

Directories are simply containers for files and other directories. They provide a tree like structure for organizing the system. Directories can be accessed by their name and they can also be accessed using a couple of shortcuts. Linux uses the symbols `.` and `..` to represent directories. Think of `.` as "this directory" and `..` and "the parent directory."

Symbol	Description
.	This directory.
..	The parent directory.
/	Directory separator. Directories end in a forward slash and this is often assumed.

The directory separator is optional for the last subdirectory in a path or command. For example, the following commands work identically.

```
$ cd /var/tmp
$ cd /var/tmp/
```

Using the shortcuts can make navigating easier. For example, type `cd..` to go to the directory just above your current directory.

```
$ pwd
/home/bob
$ cd tpsreports
$ pwd
/home/bob/tpsreports
$ cd ..
$ pwd
/home/bob
$ cd ..
$ pwd
/home
$ cd .
$ pwd
/home
```

The `cd .` command did not take you anywhere. Remember that `.` is "this directory" and `..` is "the parent directory." Another shortcut for navigating directories is `cd -`. This command takes you to the previous directory. The environment variable that represents your previous working directory is `OLDPWD`. So, `cd -` and `cd $OLDPWD` are equivalent.

```
$ pwd
/home/bob
$ cd /var/tmp
$ pwd
/var/tmp
$ echo $OLDPWD
/home/bob
$ cd -
/home/bob
$
```

How would you execute a command that is in your current directory? Assume your current directory is your home directory. By default your home directory is not in your `$PATH`. Here is how to do that.

```
$ ./program
```

Why does that work? Well, `.` represents "this directory", `/` is the directory separator, and `program` is the program to execute. You can always use the full path to be explicit. Here are two ways to execute `program`.

```
$ pwd
/home/bob
$ ./program
$ /home/bob/program
```

Creating and Removing Directories

The `mkdir` command is used to create directories and the `rmdir` command removes them.

`mkdir [-p] directory` - Create a directory. Use the `-p` (parents) option to create intermediate directories.

`rmdir [-p] directory` - Remove a directory. Use the `-p` (parents) option to remove all the specified directories. `rmdir` only removes empty directories. To remove directories and their contents, use `rm`.

`rm -rf directory` - Recursively removes the directory and all files and directories in that directory structure. *Use with caution.* There is no "trash" container to quickly restore your file from when using the command line. When you delete something, it is gone.

```
$ mkdir newdir
$ mkdir newdir/product/reviews
mkdir: Failed to make directory "newdir/product/reviews"; No such file or
directory
$ mkdir -p newdir/product/reviews
$ rmdir newdir
rmdir: directory "newdir": Directory not empty
$ rm -rf newdir
$ ls newdir
ls: newdir: No such file or directory
$ pwd
/home/bob
$ cd ..
$ pwd
/home
```