libdevdocs Documentation

Release 0.0.1

Francis Kayiwa

April 14, 2014

This document (which was initially from a workshop) is aimed at helping librarians get comfortable with a UNIX system. We will target and use the Ubuntu Operating System for the purposes of this tutorial. In addition to an introduction to UNIX the document will have a brief introduction to the Python programming language. We always welcome pull requests. :-)

CHAPTER 1

Installation

In this chapter you will learn how to install a Virtual Machine (with Ubuntu operating system) and a Python interpreter on your Machine. The motivation for using a virtual machine is mostly for uniformity in the workshop. If you are attempting this workshop outside the classroom, this chapter (or at least the virtual machine part) may be optional. However keep in mind the rest of the workshop assumes you are using Ubuntu 12.04 LTS.

1.1 On Windows

1.1.1 Downloading VMWare Player and Ubuntu

To get started, you'll need to download VMWare Player. You will need administrator privileges on your machine to install this!

1. Start by going to www.vmware.com/download/player
    (a) The page will redirect - scroll down to VMWare Player and choose "Download Product."
    (a) Look for "VMWare Player and VMWare Player Plus for Windows" and choose "Download."
    (b) Leave yourself some time, as it takes a while to download.
2. When the .exe file is done downloading, double-click it wherever you've saved it (probably your Downloads folder).
    (a) A pop-up box will open that says "Do you want to run this file?" Click "Run."
    (a) Accept the user agreement.
    (b) A new window will pop up with an install sequence - you should be able to accept any defaults and go through the sequence quickly.
2. If VMWare Player doesn't open automatically, go into your Programs menu and open it.
3. You will now see a new window - at the top should be "Welcome to VMWare Player," and the first option beneath that should be "Create a New Virtual Machine."

You can pause now in the process and move on to:

1.1.2 Downloading Ubuntu

1. Start by going to http://www.ubuntu.com/download/desktop and choose Ubuntu 12.04 LTS.
    (a) Choose Ubuntu 12.04 LTS (there will be an option to select your operating system above the orange download button).

(a) **This will take you to a page asking you to donate to Ubuntu. If you don't want to donate, you can scroll to the bo**

i. Your file should start downloading. Again, this can take a while to download, so leave yourself some time.

3. That's it! You're done.

1.1.3 Installing Ubuntu on your Virtual Machine

Now we're starting to get to the fun part!

1. Go back to your VMWare Player window that says "Create a new Virtual Machine."
2. A second window should pop up which says "Welcome to the New Virtual Machine Wizard." It asks you how you will install the operating system.
   (a) Choose the option "Installer disc image file (iso):"
   (a) Browse to where you saved your Ubuntu file (probably your Downloads folder) and choose that file ("ubuntu-12.04.4-desktop-amd64.iso")
   (b) Click "Next"
2. You will move on to "Easy Install Information"
   (a) Set your name as it will display under Full name
   (b) Set a name for your username
   (c) Set a name for your password
   (d) Click "Next"
3. Give your virtual machine a name (maybe something like "PythonClass").
4. Accept the default location and click "Next."
5. It will ask you about the disk capacity - maximum size and whether or not to split the virtual disk into multiple files. Accept the defaults and click "Next."
6. Click "Finish" - the virtual machine should start up once you do.

Important Note: We are going to be using the command line to do this next part. If your Ubuntu VM has defaulted to opening in the command line (you will see a black screen with a white blinking cursor waiting for your commands), proceed right away. If, on the other hand, your Ubuntu VM has defaulted to a normal-looking desktop interface, type CTRL-ALT-F1 to switch to the command line. To use your VM with command line, just click in the window. To get your mouse back so you can do anything else, type CTRL-ALT.

1.1.4 Installing ssh-server

Now we're really getting somewhere.

1. In your command-line interface, type: `sudo apt-get install openssh-server`
2. Type in your password.
3. After a little thinking, your machine should tell you how much space the installation will take, and ask you if you want to proceed. Type: y
4. Hit the 'Enter' key.

1.1.5 Getting ready with Python

To double-check that Python is installed on your machine (it should come with your installation of Ubuntu), type: python –version You should get back something like: Python 2.7.3

If Python is not installed:

1. Type: `sudo apt-get install python2.7`
2. It will ask for your password - type that in.
3. It should tell you how much disc space nano will take, and ask if you're sure you want to install (Y/n). Type: y
4. Hit the 'Enter' key.
5. You should be done! You can check that it installed properly by typing: python –version

1.1.6 Installing the nano text editor

Python is already installed on your machine, but your machine will still need a way to write Python programs, and help figuring out what to *do* with them. Luckily, your ubuntu VM should also come with nano, which is a text editor that can write Python. To double-check that Python is installed on your machine, type: nano –version You should get back something like: GNU nano version 2.2.6 (compiled 20:32:12, Dec 3 2010) ...

If nano is not installed:

1. Type: `sudo apt-get install nano`
2. It will ask for your password - type that in.
3. It should tell you how much disc space nano will take, and ask if you're sure you want to install (Y/n). Type: y
4. Hit the 'Enter' key.
5. You should be done! You can check that it installed properly by typing: nano –version

Congratulations! You are now officially ready to roll!

1.2 On Mac OSX

1.2.1 Downloading VMWare Fusion and Ubuntu

To get started, you'll need to download VMWare Fusion. You will need administrator privileges on your machine to install this!

1. **Start by going to www.vmware.com/products/fusion and choose "Download Free Trial" - you'll get 30 days of free access.**
    (a) Leave yourself some time, as it takes a while to download.
2. **When the file is done downloading, click on the .dmg file.**
    (a) A pop-up box will open that says "Double-click to Install" - do it!
    (a) Accept the user agreement.
    (b) Since you won't have a product key, choose the free 30-day trial option.
    (c) Put in your email address to continue (it doesn't look like there's a way around this, unfortunately).
    (d) This finishes the VMWare Fusion download and install process.

(e) You'll see a new box pop up called "New Virtual Machine."

You can pause now in the process and move on to:

1.2.2 Downloading Ubuntu

1. Start by going to http://www.ubuntu.com/download/desktop and choose Ubuntu 12.04 LTS.
    (a) Choose Ubuntu 12.04 LTS (there will be an option to select your operating system above the orange download button).
    (a) **This will take you to a page asking you to donate to Ubuntu. If you don't want to donate, you can scroll to the bo**
        i. Your file should start downloading. Again, this can take a while to download, so leave yourself some time.
3. That's it! You're done.

1.2.3 Installing Ubuntu on your Virtual Machine

We're starting to get to the fun part!

1. Go back to your VMWare Fusion window that says "Create a Virtual Machine."
2. The window should say "Select the Installation Method."
3. Make sure the "Install form disc or image" option is selected, and then click "Continue."
4. You should come to a new window which says, "Choose an operating system installation disc or image:" over a box which will be blank.
    (a) Choose the button toward the bottom, "Use another disc or disc image."
    (a) In the window that pops up, browse to wherever you saved the Ubuntu file you just downloaded (for me, it's my downloads folder).
    (b) Find the file "ubuntu-12.04.4-desktop-amd64.iso" and choose "Open."
    (c) Ubuntu should now be listed as an option and selected in the window under "Choose an operating system installation disc or image."
    (d) Choose "Continue."
    (e) Choose "Easy Install."
        i. Set your name as it will display.
        ii. Set a name for your username (Account Name).
        iii. Set a password.
        iv. Make sure the drop-down menu at the bottom says "The virtual machine can: Read & Write."
    (f) Click "Continue."
    (g) Double-check your settings and click "Finish."

Important Note: We are going to be using the command line to do this next part. If your Ubuntu VM has defaulted to opening in the command line (you will see a black screen with a white blinking cursor waiting for your commands), proceed right away. If, on the other hand, your Ubuntu VM has defaulted to a normal-looking desktop interface, click

on the 'Dash Home' button at the top left of your screen and search for Terminal. Open Terminal and then proceed with the directions.

1.2.4 Installing ssh-server

Now we're really getting somewhere.

1. In your command-line interface, type: `sudo apt-get install openssh-server`
2. Type in your password.
3. After a little thinking, your machine should tell you how much space the installation will take, and ask you if you want to proceed. Type: y
4. Hit the 'Enter' key.

1.2.5 Getting ready with Python

To double-check that Python is installed on your machine (it should come with your installation of Ubuntu), type: python –version You should get back something like: Python 2.7.3

If Python is not installed:

1. Type: `sudo apt-get install python2.7`
2. It will ask for your password - type that in.
3. It should tell you how much disc space nano will take, and ask if you're sure you want to install (Y/n). Type: y
4. Hit the 'Enter' key.
5. You should be done! You can check that it installed properly by typing: python –version

1.2.6 Installing the nano text editor

Python is already installed on your machine, but your machine will still need a way to write Python programs, and help figuring out what to *do* with them. Luckily, your ubuntu VM should also come with nano, which is a text editor that can write Python. To double-check that Python is installed on your machine, type: nano –version You should get back something like: GNU nano version 2.2.6 (compiled 20:32:12, Dec 3 2010) ...

If nano is not installed:

1. Type: `sudo apt-get install nano`
2. It will ask for your password - type that in.
3. It should tell you how much disc space nano will take, and ask if you're sure you want to install (Y/n). Type: y
4. Hit the 'Enter' key.
5. You should be done! You can check that it installed properly by typing: nano –version

Congratulations! You are now officially ready to roll!

CHAPTER 2

Post Installation

In the next section we will focus on connecting to your new Virtual Machine from the host computer. If your host computer is the Windows you will need to install the Putty application. If your host computer is a Mac you will can use the Terminal Application.

The Terminal Applications is found in the Utilities Directory under the Applications directory of all Mac OSX versions. Launch the Terminal application by double clicking and skip the next section.

If you don't already have it putty can be installed by downloading the software from *the Putty Homepage <http://www.chiark.greenend.org.uk/~sgtatham/putty/>_*

1. Download and run the current setup executable.
2. The default installation location is recommended
3. The Start Menu folder is simply the name of the sub directory under which your PuTTY shortcuts will be located. The default value should be fine.

Launch the Putty application and follow the steps in the next section.

CHAPTER 3

Ubuntu Introduction

We now have a Virtual Machine set up. In this section we will work on connecting to the Virtual Machine from the host computer.

The steps for this require us to log in and work from the Virtual Machine one "last" time. So if your Virtual Machine is not running please follow the platform specific steps below to start the Virtual Machine.

3.1 On Windows

Start the VMWare Player Application and double click on the Ubuntu Virtual Machine you created in the last section.

3.2 On Mac OSX

Start the VMWare Fusion Application and double click the Ubuntu Virtual Machine you created in the last section.

3.2.1 Get IP address of Virtual Machine

We will log into the virtual machine by using the username created in the previous section. This will be followed by launching Terminal. The way to do this is to click the Dash at the top left of your Desktop. Search for Terminal and start the application.

From within the Terminal type `ifconfig`

```
eth0    Link encap:Ethernet  HWaddr 00:26:bb:53:8d:90
        inet addr:192.168.0.14  Bcast:192.168.0.255  Mask:255.255.255.0
```

From the results we are interested in the numbers after inet addr in the eth0 device. This is the IP address of your Virtual Machine. Write this down for future use. In the fake example above the IP address of our Virtual Machine is 192.168.0.14.

3.2.2 Connecting from Windows

For the rest of the tutorial these will be the steps used to connect to your Virtual Machine. Enter the IP address in the step above and click on the Open. On your first connection you will get a pop up key warning you about the fingerprint of the server you are about to connect to. Accept the keys and enter your username and password in the window that follows.

3.2.3 Connecting from Mac OSX

For the rest of the tutorial these will be the steps used to connect to your Virtual Machine. Type the following in your Terminal.app (this was the last step in the last segment) `ssh -l username IP Address` So if your username was jondoe on my fake Virtual Machine above the steps would be `ssh -l jondoe 192.168.0.14` On your first connection you will get a warning about the fingerprint of the server you are about to connect to. Accept the keys and enter your password.

3.3 Administration Basics

On your first log in it is possible that you will see a message resembling the one I saw while creating this tutorial.

```
88 packages can be updated.
80 updates are security updates
```

It is extremely important to heed this advice. Even though you are running a virtual machine (incidentally this is the reason for lagging behind) it is important to get into good administration habits early. The way to fix this is to type the following.

```
sudo apt-get update         # Fetches the list of available updates from the Ubuntu project
sudo apt-get upgrade        # Upgrades your Ubuntu virtual machine
```

In the next segment we will get more comfortable with general computation from the commandline. To logout of your virtual machine type `exit` . You can then shutdown your VMWare Player Application or VM Fusion application.

CHAPTER 4

Command Line

You may be asking yourself about the virtue using the Commandline. In many way the Graphic User Interface is friendlier. For many users it feels familiar and to other Operating Systems. The reason for learning the commandline is primarily a way to introduce you to what administering a server that could be in a remote location where you have no physical access. Your Ubuntu virtual machine is a good place to get practice. This segment is driven towards some repetitive exercises with a goal to make you comfortable on your Ubuntu virtual machine.

4.1 Navigation

4.1.1 Location

We will start with the `pwd` which stands for Present Working Directory. Many of the commands in UNIX are often an abbreviation of a word of words describing what they do.

```
username@hostname:~$ pwd
/home/username
username@hostname:~$
```

The command `ls` which is short for list will produce the following on your termimal

```
username@hostname:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
```

Before we proceed much futher it is important to remember that all UNIX systems are designed with a built-in manual to explain what the commands do. The way to find out what it does is to prepend the command with `man` In the examples above it would be `man pwd` and `man ls`. Go ahead and type those and read what the manual says about them. As you may have see the `ls` command can take arguments while the `pwd` does not. Below is an outline of the use of `ls`.

```
ls [options][location]
```

The square brackets above are optional. Here is an example with the echo of the results shown.

```
username@hostname:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
username@hostname:~$ ls -l
total 48
drwxr-xr-x  3 fkayiwa fkayiwa 4096 Mar  7 14:09 Desktop
drwxr-xr-x  2 fkayiwa fkayiwa 4096 Dec  9 17:32 Documents
drwxr-xr-x  2 fkayiwa fkayiwa 4096 Dec  9 17:32 Downloads
-rw-r--r--  1 fkayiwa fkayiwa 8445 Dec  9 17:21 examples.desktop
```

```
drwxr-xr-x  2 fkayiwa fkayiwa 4096 Dec  9 17:32 Music
drwxr-xr-x  2 fkayiwa fkayiwa 4096 Dec  9 17:32 Pictures
drwxr-xr-x  2 fkayiwa fkayiwa 4096 Dec  9 17:32 Public
drwxr-xr-x  2 fkayiwa fkayiwa 4096 Dec  9 17:32 Templates
drwxr-xr-x  2 fkayiwa fkayiwa 4096 Dec  9 17:32 Videos
```

So what did we just do?

- The `ls` will list the contents of our current directory.
- The `ls -l` adds the optional switch (-l) to do the long listing. A long listing has the following:
  - First character indicates whether it is a normal file (-) or directory (d)
  - Next 9 characters are permissions for the file or directory (we'll learn more about them in a later segment)
  - The next file is the number of blocks.
  - The next field is the owner of the file or directory (fkayiwa/root in the examples above).
  - The next field is the group the file or directory belongs to (fkayiwa).
  - Following this is the file size.
  - Next up is the file modification time.
  - The actual name of the file or directory.

Here is another example.

```
username@hostname:~$ls -l /var
total 52
drwxrws---  5 archivematica archivematica 4096 Mar  6 11:38 archivematica
drwxr-xr-x  2 root          root          4096 Apr  1 07:35 backups
drwxr-xr-x 19 root          root          4096 Mar  6 11:10 cache
drwxrwsrwt  2 root          whoopsie      4096 Dec 26 07:35 crash
drwxr-xr-x  2 root          root          4096 Aug 20  2013 games
drwxr-xr-x 68 root          root          4096 Mar  6 11:23 lib
drwxrwsr-x  2 root          staff         4096 Dec 11 15:04 local
lrwxrwxrwx  1 root          root             9 Mar 10 13:40 lock -> /run/lock
drwxr-xr-x 24 root          root          4096 Apr  3 07:55 log
drwxrwsr-x  2 root          mail          4096 Mar 31 07:58 mail
drwxr-xr-x  2 root          root          4096 Aug 20  2013 opt
lrwxrwxrwx  1 root          root             4 Mar 10 13:40 run -> /run
drwxr-xr-x 10 root          root          4096 Mar  6 11:23 spool
drwxrwxrwt  2 root          root          4096 Apr 19  2012 tmp
drwxr-xr-x  2 root          root          4096 Dec 11 11:59 www
```

- We ran `ls` with a command line argument (/var). When we do this it tells `ls` not to list our current directory but instead to list that directories contents.

Spend a little time perusing the `ls` manual by doing the following. `man ls`

4.1.2 Paths

The previous section touched on something about navigating your Ubuntu machine, namely the paths on the operating system. Understanding paths is essential in using your new computer. Whenever we refer to a file or directory on the command line we are in fact referring to the actual path to it.

Referring to paths can be done using the absolute path or a relative path.

At the top of the unix system is what is called the root directory. It is denoted by a single dash (/). It has sub-directories which have other sub-directories contained within them. Files can reside in any of these directories.

Absolute paths refer to the location relative to the root directory. Relative paths specify location based on your current working directory. The example below will illustrate what we mean.

```
username@hostname:~$ pwd
/home/fkayiwa
username@hostname:~$
username@hostname: ls Desktop
test.mrc
username@hostname:~$ ls /home/fkayiwa/Desktop
test.mrc
username@hostname:~$
```

So what just happened?

- We ran `pwd` to verify where we were on the filesystem.
- We ran `ls` providing it with a relative path. Desktop is a directory in our current location. We would get different results depending on where we are.
- We finally ran `ls` providing and absolute path. This will provide the same output regardless of the location we ran it from.

Here are a few more tips that can help build on your new knowledge on paths.

- ~ (tilde) - This is a shortcut for your home directory. So if the user fkayiwa above wanted to head to their home directory. Typing ~/Desktop in the example above is the equivalent of typing /home/fkayiwa/Desktop
- . (dot) - This is a reference to your current directory. We will see more on this later.
- . . (dotdot) - This is a reference to the parent directory. You can use this several times to keep going up the heirarchy.

A few more examples to explain this.

```
username@hostname:~$ pwd
/home/fkayiwa
username@hostname:~$
username@hostname:~$ ls ~/Desktop
test.mrc
usename@hostname:~$ ls ./Desktop
test.mrc
username@hostname:~$ ls /home/fkayiwa/Desktop
test.mrc
username@hostname:~$ ls ../../
bin boot lib lost+found proc selinux usr boot home lib32
...
username@hostname:~$ ls /
bin boot lib lost+found proc selinux usr boot home lib32
...
```

Spend time listing the content of various directories on your filesystem and familiarize yourself on how the elements of building a path.

4.1.3 Moving around your filesystem

In order to move around your filesystem we use the `cd` command which stands for change directory.

A handy tip to remember is if you run the command `cd` without any arguments then it will always take you back to your home directory.

Otherwise the `cd` command can be used much like the `ls` before to change into absolute or relative paths as some of the examples below will show.

```
username@hostname:~$ pwd
/home/fkayiwa
username@hostname:~$ cd Desktop
username@hostname:~$ ls
test.mrc
username@hostname:~$ cd /
username@hostname:~$ pwd
/
username@hostname:~$ ls
bin boot lib lost+found proc selinux usr boot home lib32
...
username@hostname:~$ cd ~/Desktop
username@hostname:~$ pwd
/home/fkayiwa/Desktop
username@hostname:~$ cd ../../
username@hostname:~$ pwd
/home
username@hostname:~$ cd
username@hostname:~$ pwd
/home/fkayiwa
```

Another handy tip when navigating is Tab Completion. Typing out these paths can become tedious. If you're like me, you're also prone to making typos. The command line has a nice little mechanism to help us in this respect. It's called Tab Completion.

When you start typing a path (anywhere on the command line, you're not just limited to certain commands) you may hit the Tab key on your keyboard at any time which will invoke an auto complete action. If nothing happens then that means there are several possibilities. If you hit Tab again it will show you those possibilities. You may then continue typing and hit Tab again and it will again try to auto complete for you.

It's kinda hard to demonstrate here so it's probably best if you try it yourself. If you start typing cd /hTab/<beginning of your username>Tab you'll get a feel for how it works.

4.1.4 Summary

Commands we learned about

```
pwd - Present Working Directory
ls - List the contents of a directory
cd - Change directories
```

CHAPTER 5

Command Line: Files

5.1 Everything is a File

Ok, the first thing we need to appreciate with linux is that under the hood, everything is actually a file. A text file is a file, a directory is a file, your keyboard is a file, your monitor is a file etc. Understanding this about Linux will make managing of these files and directories much easier to understand.

Another key point to remember is that Linux has no extensions to identify what a file will do. This one can sometimes be hard to get your head around but as you work through the sections it will start to make more sense. A file extension is normally a set of 2 - 4 characters after a full stop at the end of a file, which denotes what type of file it is. The following are common extensions in the Microsoft Windows Operating systems:

- file.exe - an executable file, or program.
- file.txt - a plain text file.
- file.png, file.gif, file.jpg - an image.

Under Linux the system actually ignores the extension and looks inside the file to determine what type of file it is. So for instance I could have a file myself.png which is a picture of me. I could rename the file to myself.txt or just myself and Linux would still happily treat the file as an image file. As such it can sometimes be hard to know for certain what type of file a particular file is. Luckily there is a command called `file` which we can use to find this out.

```
file [path]
```

The command above will provide you information about the file or directory (a special kind of file really)

5.2 Linux is Case Sensitive

In order to explain this let's take a slight detour.

```
username@hostname:~$ cd ~
username@hostname:~$ touch Desktop/file1.txt
```

Before proceeding take time to look at the manual on the command `touch` and `file`. In fact from here on every time you encounter a new command spend a few minutes reading up the manual on a more detailed account of what the command does.

On Microsoft Windows Operating systems it is possible to have two or more files and directories with the same name but letters of different case.

```
username@hostname:~$ ls Desktop
test.mrc file1.txt
username@hostname:~$ file Desktop/file1.txt
/home/fkayiwa/Desktop/file1.txt: empty
username@hostname:~$ file Desktop/FILE1.TXT
/home/fkayiwa/FILE1.TXT: ERROR: cannot open `/home/fkayiwa/Desktop/FILE1.TXT' (No such file or direct
```

Linux sees these all as distinct and separate files.

Also be aware of case sensitivity when dealing with command line options. For instance with the command `ls` there are two options s and S both of which do different things. A common mistake is to see an option which is upper case but enter it as lower case and wonder why the output doesn't match your expectation.

5.3 Spaces in names

Spaces in file and directory names are perfectly valid but we need to be a little careful with them. As you would remember, a space on the command line is how we seperate items. They are how we know what is the program name and can identify each command line argument. If we wanted to move into a directory called MARC Records for example the following would not work.

```
username@hostname:~$ ls Desktop
test.mrc file1.txt
username@hostname:~$ cd MARC Records
-bash: cd: MARC: No such file or directory
```

What happens is that MARC Records is seen as two command line arguments. cd moves into whichever directory is specified by the first command line argument only. To get around this we need to identify to the terminal that we wish MARC Records to be seen as a single command line argument. There are two ways to go about this, either way is just as valid. (details on how to create this are in the next section)

The first approach invovles using quotes around the entire item. You may use either single or double quotes (later on we will see that there is a subtle difference between the two but for now that difference is not a problem). Anything inside quotes is considered a single item.

```
username@hostname:~$ cd ~
username@hostname:~$ cd Desktop/'MARC Records'
username@hostname:~$ pwd
username@hostname:~$ /home/fkayiwa/Desktop/MARC Records
```

Another method is to use what is called an escape character, which is a backslash ( ). What the backslash does is escape (or nullify) the special meaning of the next character.

```
username@hostname:~$ cd ~
username@hostname:~$ cd Desktop/MARC\ Records
username@hostname:~$ pwd
username@hostname:~$ /home/fkayiwa/Desktop/MARC Records
```

In the above example the space between MARC and Records would normally have a special meaning which is to separate them as distinct command line arguments. Because we placed a bashslash in front of it, that special meaning was removed.

CHAPTER 6

Making a Directory

Linux organises it's file system in a heirarchical way. Over time you'll tend to build up a fair amount of data (storage capacities are always increasing). It's important that we create a directory structure that will help us organise that data in a manageable way. I've seen way too many people just dump everything directly at the base of their home directory and waste a lot of their time trying to find what they are after amongst 100's (or even 1000's) of other files. Develop the habit of organising your stuff into an elegant file structure now and you will thank yourself for years to come.

Creating a directory is pretty easy. The command we are after is mkdir which is short for Make Directory.

```
mkdir [options] <Directory>
```

In it's most basic form we can run `mkdir` supplying only a directory and it will create one.

```
username@hostname:~$ cd ~
username@hostname:~$ ls
Desktop Documents Downloads Music Pictures Public Templates Videos
username@hostname:~$ mkdir libdevops
username@hostname:~$ ls
Desktop Documents Downloads Music libdevops Pictures Public Templates Videos
```

So what happened here?

- Let's start off by making sure we are in the home directory
- We'll do a listing so we know what is already in our directory.
- Run the command mkdir and create a directory libdevops

Remember that when we supply a directory in the above command we are actually supplying a path. Is the path we specified relative or absolute? Here are a few more examples of how we can supply a directory to be created

```
username@hostname:~$ cd ~
username@hostname:~$ mkdir /home/fkayiwa/foo
username@hostname:~$ mkdir ./bar
username@hostname:~$ cd Desktop
username@hostname:~$ mkdir ../dir1
username@hostname:~$ mkdir ~/libdevops/dir2
```

If these don't make sense then review the section on Files.

There are a few useful options available for `mkdir`. Spend some time on the manual to find out which. I will look at two that I've used quite a bit over the years.

The first one is -p which tells mkdir to make parent directories as needed (demonstration of what that actually means below). The second one is -v which makes mkdir tell us what it is doing (as you saw in the example above, it normally does not).

```
username@hostname:~$ cd ~
username@hostname:~$ mkdir ~/libdevops/foo/bar
username@hostname:~$ cd ~/libdevops/foo/bar
username@hostname:~$ pwd
/home/fkayiwa/libdevops/foo/bar
```

And now the same command but with the -v option

```
username@hostname:~$ cd ~
username@hostname:~$ mkdir -pv ~/libdevops/foo/bar/foo
mkdir: created directory ‘/home/fkayiwa/libdevops/foo/bar/foo’
```

CHAPTER 7

Removing a Directory

Creating a directory is pretty easy. Removing or deleting a directory is easy too. One thing to note, however, is that there is no undo when it comes to the command line on Linux so some care will have to be exercised when using this. The command to remove a directory is `rmdir` , short for remove directory.

```
rmdir [options] <Directory>
```

Two things to keep in mind. Firstly, rmdir supports the -v and -p options similar to mkdir. Secondly, a directory must be empty before it may be removed (later on we'll see a way to get around this).

CHAPTER 8

Copying a File or Directory

There are many reasons why we may want to make a duplicate of a file or directory. Often before changing something, we may wish to create a duplicate so that if something goes wrong we can easily revert back to the original. The command we use for this is `cp` which stands for copy.

```
cp [options] <source> <destination>
```

There are numerous options available to cp. As always check the manual. I will introduce some that I have used frequently over the years.

```
username@hostname:~$ cd ~/libdevops
username@hostname:~$ touch example1 foo
username@hostname:~$ cp example1 dewey
username@hostname:~$ ls
dewey example1 foo
```

Note that both the source and destination are paths. This means we may refer to them using both absolute and relative paths.

CHAPTER 9

Command Line: Permissions

We have not properly discussed permissions on your Linux files and directories. Permissions specify what a user may or may not do with respect to a file or directory. As such, permissions are important in maintaining the integrity of your computing. For instance you don't want other users to be changing your files and you also want system files to be safe from damage (either accidental or deliberate). Permissions in a Linux system are quite easy to work with.

Permissions dictate 3 things you may do with a file, read, write and execute. They are referred to in Linux by a single letter each.

- r read - user may view the contents of the file
- w write - user may change the contents of the file
- x execute - user may execute or run the file as a program of script

For every file we define 3 sets of people for whom we may specify permissions.

- owner - user owns the file. (typically the user who created the file but ownership may be granted to other users)
- group - every file belongs to the single group
- others - everyone else who is not in the group or owner

Three persmissions and three groups of people. That's about all there is to permissions really. Now let's see how we can view and change them.

9.1 View Permissions

To view permissions for a file we use the long listing option for the command ls.

```
ls -l [path]
```

```
username@hostname:~$ ls -l /home/fkayiwa/Desktop/test.mrc
-rw-r--r-- 1 fkayiwa fkayiwa 1170 Mar 27 16:25 /home/fkayiwa/Desktop/test.mrc
```

In the above example the first 10 characters of the output are what we look at to identify permissions.

- The first character identifies the file type. If it is a dash (-) then it is a normal file. If it is a d then it is a directory.
- The following 3 characters represent the persmissions for the owner. A letter represents the presence of a permission and a dash (-) represents the absence of a permission. In this example the owner has all permissions (read, write and execute).
- The following 3 characters represent the permissions for the group. In this example the group has the ability to read but not write or execute. Note that the order of permissions is always read, then write then execute.

- Finally the last 3 characters represent the permissions for others (or everyone else). In this example they have the execute permission and nothing else.

9.2 Change Permissions

CHAPTER 10

UNIX Shell Scripting

CHAPTER 11

Indices and tables

- *genindex*
- *modindex*
- *search*