Commanding Windows

How to get more efficient by using the cmd command line interface.

Mauricio Fernández

Commanding Windows

How to get more efficient by using the cmd command line interface.

Mauricio Fernández

This book is for sale at http://leanpub.com/commandingwindows

This version was published on 2015-01-03

This is a Leanpub book. Leanpub empowers authors and publishers with the Lean Publishing process. Lean Publishing is the act of publishing an in-progress ebook using lightweight tools and many iterations to get reader feedback, pivot until you have the right book and build traction once you do.

Contents

Chapter One: Introduction

The Windows command line

Current windows command line `cmd.exe` is a descendant of of the DOS command.com, the old oeperating system that came with PC compatible computers, as such it contains all of the functions needed to move around the disks, manage files, changing and consulting system settings. Of course cmd has evolved with the operating system, and commands have been added to manage all other aspects of the operating system. In general, you can do everything thay you do in the graphical interface by typing commands in a cmd.exe window.

Why is that inportant?, Well there are several reasons, the ones we are more interested in this book are efficiency and automation. On the efficiency front, there are things you can do in the command line that would take a lot of steps in the graphical user interface, things like selecting all files with extension mp3, and copy them to a USB drive, or finding the biggest files on your system, etc. The automation is another way of increasing efficiency, and allows us to safely perform error prone operations and eliminating repetition.

In this book I will assume no prior knowledge of the command line or Linux Shells, but we will try to get to a level where you can think in terms of windows commands, how to combine them and accomplish things with smaller involvement from the user.

In the first chapters will go trough the basic commands and concepts, things like piping, redirection, environment variables, for loops, etc. The idea is to explain by creating useful examples. At first, we want to create one liners. Things that you would type in your command line directly. Later chapters will look at batch files and what we can accomplish with those.

It is important to understand right away that the book does not intend to be a programming course, we will simply combine tools, for complex behaviors I suggest to learn programming languages with scripting capabilities, I think it is not worth to complicate things when proper programming languages are the best solution.

Why would I want to read this book?

The book is intended to an audience of people that are not afraid of typing commands, experiment, and make mistakes, but who know, that in the end, the gained efficiency is worth it, people that don't mind spending time to figure things out once, because they know they can use what they learn in future occasions, and once their brain is wired to work with commands they can accomplish a lot with little actual interaction with the computer. xb Computers are meant to do a lot of our work for us, we just have to learn how to tell them what to do in an efficient and unambiguous way, such that

the user can leave the computer doing something tedious while he faces the real challenges that the computer can not figure out by itself. Today all operating systems have multitasking capabilities, so we can have the computer do one (or several) thing(s) while we do another.

So, if you feel like you are doing the same thing over and over, clicking, right clicking, going trough menus, drop down lists, radio buttons, and check-boxes repeatedly, and you fill there should be a better way, stick around and maybe you'll find a way to stop going through a lot of mechanical motions, and reduce the chances of making mistakes.

Hopefully you are intrigued enough, and you want to spend time with the rest of the book, and hopefully you will find it not only informative but also entertaining, and maybe you can let me know what you think is good and what is not so good about the book, and help me learn from this book too.

Chapter Two: First Steps

Starting the command prompt.

As I said before I won't assume any knowledge of the command prompt, so of course I won't assume you know how to open one, but just before I tell you how, and because I know myself, lets get trough some vocabulary, I will often use the term "terminal", which refers in our book, unless otherwise stated, to a cmd.exe window, in general the interaction with terminal window requires typing, and most of the time the output is readable text, shown in the terminal. I will also sometimes call the terminal a command window, which is really what it is, cmd is short for command, which is the **command line interpreter**, the sole purpose of that program is to take the commands that you write, interpret them, and if correctly formed, execute them, the title of this book is "Commanding Windows" for that reason. We will be telling our computers what to do with commands.

The easiest way to open a command window, is by using the windows key, that is the key which normally in the bottom row of your keyboard, and has the windows logo or something very similar printed in it, if you just pressed that key, ask I know many of the eager readers did, it will just open the windows bar, the thing that many people don't know is that that key can be used in combination with others, as a shortcut to execute programs, in our case we want to press both the windows key and the "R" key at the same time, this will open a small window with a text box, that textbox can be used to launch any application by name, and that is what we want to do, the name we will type there is "cmd", if we hit the enter key now, a window will open, that window normally has a black background, and looks a bit like this:

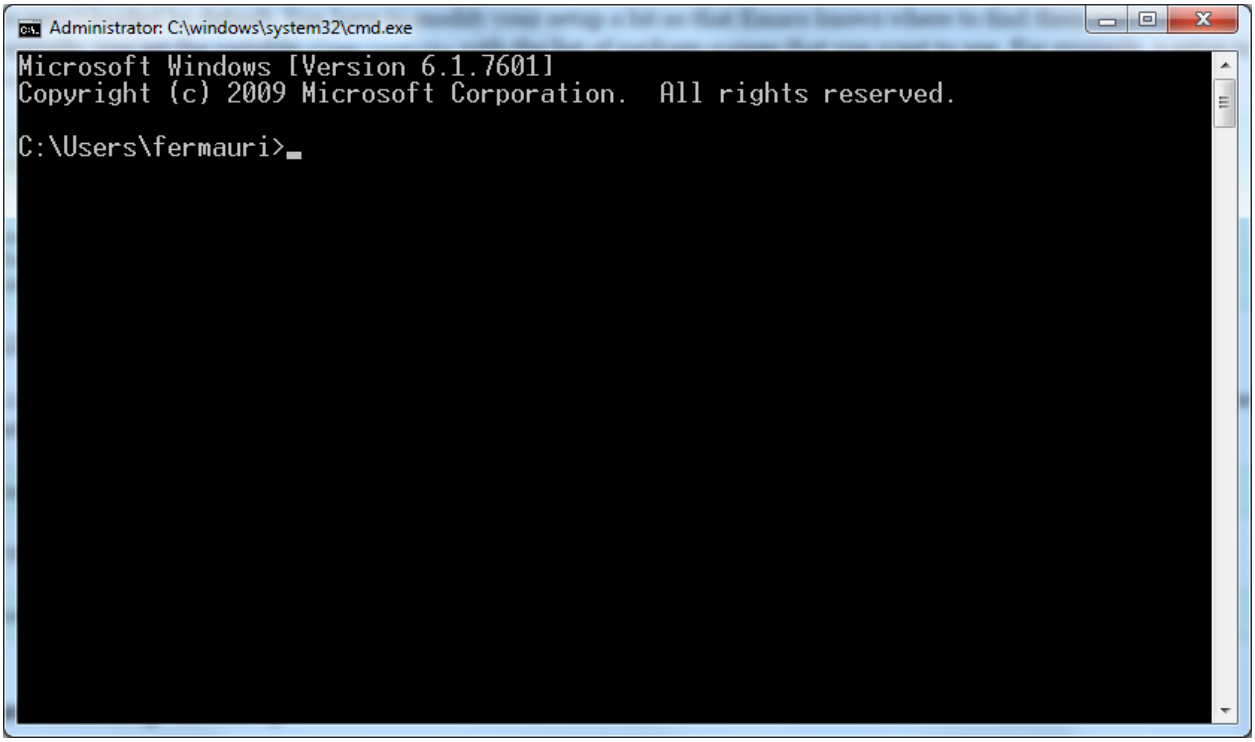

Figure 2-1. Terminal window

Wow!, that looks... not very impressive, I know, but that is the type of minimalistic interface the computers had back in the day, no buttons, no icons, no menus, just some letters and symbols, What are we supposed to do with this?

I the old days, that window occupied the whole screen, and it was the only way to "speak" to our computer, so let's describe what we are seeing, to gain some additional vocabulary, which we will be using. The blinking underscore ("_") symbol is called cursor, and it is where whatever you write will appear, the line where it is is called the command line, that is why some applications that can be used directly in the terminal are said to have a command line interface or CLI.

The command line always start with some characters which normally describe the drive letter (i.e. C:), and path to the current directory (i.e. `\Windows\System32`), and finally a greater than sign "`>`" aka closing angle bracket, whichever name you prefer. All of this is called the prompt, because it is there to prompt you to write your command.

Historical note

Way, way back IBM PC computers did not come with a hard drive, instead you booted from a floppy disk, a removable media that was inserted in a reader unit, the first such unit was called A:, and many people bought computers with two such readers to be able to copy disks, or information from one floppy to another one, that second drive was called B:, so when computers started having a hard disk drive (HDD) it was the C: drive. And so it remains until today, in most computers the primary drive is C:.

What is this Path thing?

Thanks to the graphical user interfaces we are used to see the directory structure as a tree, something roughly like this:

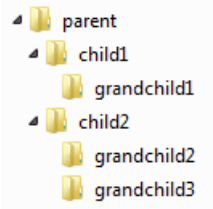

Figure 2-2. A simple tree

In this picture, there is a directory named parent which has two child directories and three grandchild directories, the first grand child is a child of child1, and the other two are children of child2. Parent itself can have a parent, all directories do except for one called appropriately "root directory", each drive in a windows based operating system has a root directory, all children of a directory are said to branch from that directory. Lets say that we have everything stored in our C drive (C:), the root of C as a path in Windows is written as `C:\`. If the tree illustrated in our figure is in the root directory, and we were in that directory the string would look like `C:\parent`, and the child1 path to child1 would be described as `C:\parent\child1`. By the way the `\` is called a backward slash or backslash, while `/` is called forward slash or simply slash, currently windows can use either one to describe paths, but traditionally the backslash is used.

Then, What is a path?, As the name suggests, it is a "road" or "way" we follow to get from one place (a directory) to another, a path can also describe how to get to a file, so it is a way of giving directions. So for those of you that enjoy top of the line illustrations (I made this one myself), consider the following map, where each blue line is a town (a directory) and each line is a road.

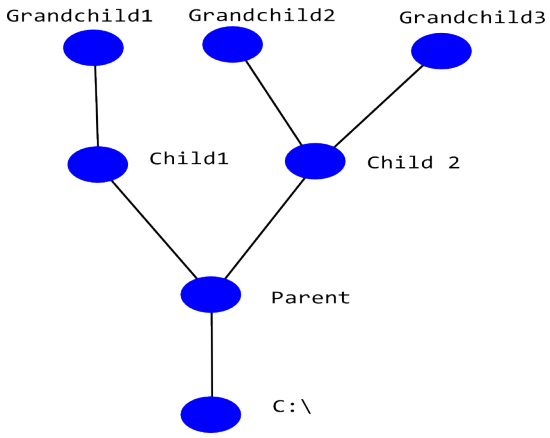

Figure 2-3. Map of the republic of C:

Let's say that the Republic of C: international airport is in it's root (\) town, you arrive and rent a car, you want to get to grandchild2, because that is where "cool people" hang, you ask the person at

the counter how to get there, because for some reason GPS signals don't work here. The guy tells you:

- From root city there is only one road leaving, it goes to Parent town, it has a sign that says so, you pass trough the town and get to a crossroad, follow the sign that says child2, that will take you to child2 town, but you don't want to stop there, just keep driving until you find another crossroad, there follow the sign that says grandchild2, and keep driving until you find the town, can't miss it.

In the meantime, since you are a smart guy, you just write the directions without so much verbosity. You decide that `\` means "go toward", and you will start by writing the name of the place where you are, so you write `C:\Parent\Child2\GrandChild2`, and you leave the place while the guy in the counter is still talking, since you already have what you need. How rude of you!!.

You spend a nice day at grandchild2, and find a hotel there called "The Spreadsheet Inn", the rooms feel like cells, but it's cheap enough, the next day you have to go to Child1 Town, because that is where you are doing business, so you go to the lobby and grab a map of C:. Looking at the map you notice that you want to go to `C:\Parent\Child1`, but of course you don't want to go all the way back to the airport to be able to get your bearings. So you study the map, and decide that you just have to drive back to Child2, since there is no other place to go back to you just write `..`, then you have to go back to Parent town, you write `\..` and then go toward (`\`) Child1 town, so your directions look like this now `..\..\child1`.

The `C:\Parent\Child1` directions is what we call an absolute path, it tells you how to get to where you want to go from the root of the drive. The `..\..\child1` path is called a relative path, because it is relative to where you are, any path starting at root, either with the C: or without it is an absolute path, any path starting with `.`,`..` or a name is a relative path. Since the directory structure is represented as a tree, each directory can have just one parent directory, and we can refer to it by `..`, we can refer to the current directory as `.` for everything else we need to use a name.

Some examples to help clarify things, we are right now at child1 and want to go to `C:\parent\child1\grandchild1`, we can write our path as `.\grandchild1` (from where I am `.`, go toward `\` GrandChild1) or simply `GrandChild1` (goto GrandChild1). If we wanted to go to `C:\Parent` instead of going to GrandChild1 our path could be written `..`, since we only need to go back to the parent directory.

Navigating around

Hopefully by now you know how to create strings that represent paths to different directories, but, Why did we learn that?; The fact is that that is how we tell the command line where we want to be. So let's learn our first command, after the prompt we can write `cd` and then press the enter key.

Used like that the cd command only tells us where we are. That information is already on the prompt so, Why bother?. there are several reasons, which we will cover in future chapters, for now let's just be happy with the fact that we can ask about our current location and get an answer.

But what if we want to move to the root directory, then we can type `cd \` and then hit enter, and voila. You can see that the prompt now changes to `c:\>`, if you were in the c: drive, if you were in a different drive then you will see the letter corresponding to that other drive.

If the directory structure we described before was in our root directory, and we wanted to get to child1 we could type `cd \parent\child1` and then press enter and we would move to that directory. And afterward you could give the command `cd ..` and move to `parent`, so everything works as we described before.

Now, imagine that you plug a thumb drive in your computer, if you go to MyComputer in the Explorer you will see that a new drive has appeared, for purposes of our explanation we will call it F: (you can try this in your computer by changing the letter to the right one), if you wanted to go to the root directory of the F: drive you would issue the command `cd F:\`, and after hitting enter you would be disappointed to see that nothing happened. Wait.. What?

I've been hiding information from you, there is actually a current directory in each drive, so by saying `cd F:\` we are actually telling the command line interpreter, make the current directory on the F: drive `\` (the root), but our terminal is still operating on C: (or whatever drive you were on). To operate on the F: drive we simply type "F:" and hit enter, and now you will be where you wanted to be.

In future chapters we will see why this is convenient, in the mean time, remember that you can change the current directory (cd) on any drive with a cd command, but if you want your commands to operate on a different drive you have to change explicitly.

Getting the map

Before we leave this chapter I want you to be able to navigate your own disk, that way you can practice a bit. I my metaphor before we had a map we could follow, and that mad it easy for us to form a string describing where we wanted to go, but asides from the root directory you do not know what are the places you can go to, let us solve that, there is a command called "tree" if you type tree in the terminal and then hit enter you will get a map of all child locations starting from the current directory in the drive you are currently at, if you do that in `c:\` you will most likely get a very big map that won't fit in your screen. For practice purposes using the Explorer GUI crate folder in C: called grandpa, and in it create the tree structure we have been working with, then in the terminal move to your newly created directory `cd c:\gradpa` and try the `tree` command there. You will get the ASCII chart representing the tree we just created, which is our map.

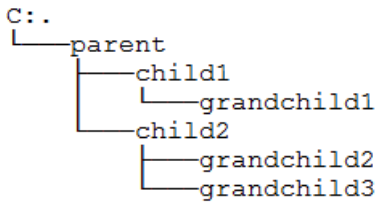

Figure 2-4. Ascii tree representation

Just checking the water

Our feet are now wet, but we are just checking the water, in future chapters we will see additional commands which we will use to determine where we want to go. We will also configure our registry file to have auto completion when typing a path, and we will do other cool stuff that will allow us to work just in the terminal without ever having to go to the graphical interface. This chapter's purpose is to allow you to be able to understand the initial screen you get when you start a terminal, but we will do lots of other stuff, and have better explanations on what we are doing.