CSCI 1100L: Topics in Computing Lab  
Lab 3: Windows Command Prompt

Purpose: So far in Lab, we’ve played around and learned more about Windows 10 and the WIMP (Windows- Icons, Menus, Pointers) interface. Today, we’ll give you some experience playing around with the command-line interface of Windows.

Why should you care? Sometimes it’s easier or quicker to do things via the command line than to use the WIMP interface. We’ll have you think about this more towards the end of the lab. But also, to become more computer fluent, it’s imperative that we learn more about the available commands in Windows.

Today’s lab consists of:
- Learning how to use Windows’s command line tools to organize, manage, and navigate to files.

Part 1: History of the Command-Line Language

Before there was Windows... there was the Microsoft Disk Operating System commonly referred to as MS-DOS. There was also a popular operating system called UNIX, but it was more popular with scientists and super computer users, while many of the “ordinary folk” used MS-DOS. Whereas Windows has a lovely graphical user interface, DOS was all done with text commands. That is, you typed out what you wanted the computer to do.

Later the Windows operating system was built on top of DOS, so that DOS commands would run in the background, corresponding to how one interacted with the graphical interface. For example, if you click and dragged a file to a folder to move it, Windows would literally run the corresponding move command in DOS. Something similar to DOS still lives on in Windows 10, and that is what we’ll be looking at today.

Part 2: Windows Command Prompt

So, how do we start giving text commands to Windows? First, we need to open the command prompt (or prompt for short). Click on the Windows Start Button, then search for “cmd” in the Search box, and click on the program cmd.exe that appears in the search list. A cheery black and white window should pop up. Notice there are no menus! Click in the Command prompt window to begin entering commands.
The prompt.

The “prompt” is the text that appears on the screen before a blinking “_” (underscore) character. The default prompt is the Drive letter and Path of your current location on the computer; in this case, it is your I drive followed by a “>” (the greater-than sign just separates the prompt from the text you enter). Notice the prompt I:\>. A prompt sits there and waits, telling you that the COMMAND processor is expecting you to enter a command. This is a rather boring (but useful) prompt, which we can change to something more interesting if we like. We’ll learn that later. You may find a path other than your I drive. It is totally fine!

The color command and getting help.

First, notice the console has some very basic colors: black background with white foreground color. Using the **color** command, we can alter these colors to suit our taste. To find out how to use the color command (or any command, for that matter), type the command name followed by a /? and hit enter. It’s always good to put a space in between the command and anything else after it. Since we want to find out about the color command, type

```
color  /?
```

and hit enter.

Next, we need to know how to read the help text. The very first line defines what the command does, and the second line shows what you need to type to get it working. The special notation

```
COLOR [attr]
```

means that the command name **color** is mandatory, but the attributes after the command word are optional. Anything optional has brackets [ ] wrapped around it in the help text. Don’t actually type the brackets. They’re just informative, saying, anything you type after the command word **color** is optional. The help text afterwards explains how to set the attributes. Read the rest of the color help text and see if you can set the console background to blue and the foreground color to light green. (Hint: in this case, it might be easier to think of these “hex digits” it talks about as just symbols. The symbols range from 0-9 and from A-F. The help text explains what each symbol represents. The very last line of this help text gives you a useful example to see how the use of these hex digits work with the color command.)

Try a few color combinations until you find a combination you like. Or you can set it back to black and white if you don’t like the colors. If your eyes can stand it, try light blue background with light red foreground. :) 

Changing drives.

Data stored on a Windows machine is divided into **drives, folders**, and **files**. Drives normally represent physical disks, but there are also **network drives** which are provided by a remote server.

For this exercise we need to create some new files and folders. We shouldn’t alter the contents of the C: drive, but we can certainly list out what files and folders are on the C: drive. To change from one drive to another, we type in the drive letter followed immediately (no space) with a “:” (colon). For example, we need to change drives from the I: drive to the C: drive. Just type

```
C:
```

and you now see the prompt tells us we’re on the C: drive now. If you were already in the C drive you don’t need to type C:
For the next part, let's move into the Windows subdirectory of our C drive. You'll learn more about the **cd** command in a little bit, but typing the command

```
cd Windows
```

will get us into the subdirectory Windows.

**The nifty dir command.**

To list out all the files and folders that are inside of a folder (or “directory”), we use the **dir** command. Since this is a new command to us, type

```
dir /?
```

to get help on the usage of this important command. There are lots of options! Note that all the parameters are optional (shown in square brackets). This means you can use the dir command verb all by itself. Type

```
dir
```

-- and you will see a listing of all the files and directories in your prompt is currently in. The dir command by itself lists the contents of whatever folder we are in. Since we are in the Windows directory of the C: drive, it lists all files and folders in there. From where we are, if we wanted to list the contents of another drive, say the I drive, we'd type

```
dir I:
```

**Formatting dir’s output.**

- Look at the **dir /?** help text and then type a command that makes the directory listing appear in “wide” format.
  - You should find that /W used with the dir command will give you the directory listing, as listing of all files and folders in that directory in wide format. Try it.
    - **dir /W**

**Switches.**

Switches are the slash letters, like /W or /B from the dir command’s help text, that tells the command prompt to do something more than just the default. For example, when you tack on /B to the dir command, like

```
dir /B
```

it turns on “bare format” (get it? switch – turn on). Switches only make sense to the command processor when you use them with the command they are listed with.

- In the second line of the dir help text, notice also that you can piggyback any or all of the switches. Type a command that produces the “**long list**” output and also includes the **owner** of each file.
- Again piggy backing the switches, type a command that lists the current directory and all its subdirectories that pauses after each screenful of information. Once you get it working, after the third screenful of information, CTRL + C to kill the command. This is a handy tool in case you ever type a command that goes haywire.
Dir's sortorder.

- Type a command that lists the files in order of size, with the smallest file first. The `dir` help text tells you to use “`dir /O sortorder`”, where the symbol `sortorder` should be replaced by one of the codes N, E, G etc. listed in the output, e.g. the command `dir /ON` sorts by name.
- Type a command that lists the files sorted by file extension.

**Important Note:** Computer scientists do not memorize all commands, attributes, or switches. We memorize a handful of useful commands, and `dir` is one such command that is useful to know. And it is also extremely useful to know how to get help with a command by using `/?`. On your lab quiz, you will be expected to know how to use command prompt help screen in order to write commands to do things like “write a command that will **list the owners** of the files in the I: drive *sorted by the size* of the files”. Practice using the `dir` command, and many of its options until you get the hang of it. The `dir` command is one of the most useful commands because it allows you see everything in the directory that you are in.

Making and removing a folder.

Let's change drives back to the I: drive. Remember, to change drives, just type in the drive letter followed immediately by a colon.

Let’s make a new folder to contain your work. The `mkdir` command (or just `md` for short) creates a new directory, also known as a **folder**. To get help on how to use the `mkdir` command, type

```
mkdir /?
```

You will see that the usage is simple: the command verb `mkdir`, followed optionally by a drive name, followed by the path of the directory you wish to make. OK, let’s make a directory named “Joe College”:

```
mkdir Joe College
```

Windows responds by giving you another command prompt. There were no error messages, so Windows believes it has correctly carried out your command. Just to make sure, let’s list the contents of our current location:

```
dir
```

You will see that Windows actually created two new directories, one named “Joe” and the other named “College”! This is a handy shortcut for when we wish to make more than one folder at a time, but at this time, this isn’t what we wanted, so let’s remove the directories (**rmdir** or just **rd** followed by the folder name will remove the folder):

```
rmdir Joe
rmdir College
```

By listing the directory contents again using the `dir` command, you should see that the unwanted directories have been removed.

In order to clue the command interpreter into the fact that both “Joe” and “College” are part of the path name, you need to surround them with quotes.

Let’s try this again. This time use the `mkdir` command to create a new directory with *your name*, such as

```
mkdir YourName
```
mkdir "Jane Smith"
Be sure to use your own name here! Typing **dir** again, you should see that a single directory has been created with your name.

**Moving to another directory.**

Another fundamental command is the **change directory** or **cd** command. The **cd** or **chdir** command changes directories, or moves you from one folder to another. Type **cd /?** to get help on the usage of this command. Type

**cd your directory**

-- where **your directory** is the directory you just created (you may need to use double quotes around your directory name). Notice you moved into the subdirectory called **your name**. For example, typing **cd foldername** moves you down into a subfolder called **foldername** (obviously, we don’t have a folder called foldername, so don’t actually type this in). What if you wanted to go back up one directory level? Type in the dir command again. Notice at the top of the listing you see two directories, one called “.” and one called “..”. Here’s what they mean:

The single dot “.” directory is a reference to “this” directory, i.e. where you currently are. Don't worry too much about it.

The double dot “..” directory is a reference to the “parent” directory. Thus if we want to move up a directory level, we just change directories to the parent directory by typing

**cd ..**

just as if “..” was the name of a folder.

Now we’re back in the root directory of our I: drive. Before going further, let's change directories back to **your directory**.

**cd your directory**

**Creating subdirectories.**

You should now be in your personal directory located in your I: drive, and the command prompt should give you the path to your current working directory. Create two directories named PLANTS and ANIMALS (using the **mkdir** command). Change directories to PLANTS, and create the directories HERBS, TREES, and VEGETABLES. Continue until you have created the following directory tree structure:
Remember, you will eventually find that you need to use the `cd ..` command to "back up" one directory. You may find it useful to "see" these directories being made in a visual way. Use the keyboard shortcut to open up My Computer (Windows Logo + E), and open your I drive and your directory. Notice that each time you use the mkdir command, the new directory will appear in your I drive.

If you think you've finished, go back to your directory in your I: drive, and type the `tree` command. The tree command will show you graphically (using the command prompt) a directory's tree structure. When you use the command you should see something similar to what's below. If your tree looks different, then go back and correct your errors.

![Directory Tree Image]

**Exploring a directory tree.**
At this point you are probably working down somewhere in your directory tree. Type

```bash
cd your directory
```
-- you will get an error message, “The system cannot find the path specified.” The problem is that Windows is looking for the directory name as a subdirectory of the current directory. To fix this, type

```
cd \your directory
```

This should work. Starting the path with `\` creates an **absolute path**. The effect is similar to an absolute cell reference in an Excel formula.

Type `dir /S` to see a listing of the current directory plus all its subdirectories. Wow, that’s a lot of information that scrolls by quickly. What a segway for our next segment, eh?

**Part 3 Capturing output to a file.**

If any command gives output to the console, you can redirect it into a text file. Here’s how:

Suppose we wanted to capture the output of the `dir` command to a filename called DirOutput.txt. All we need to do is add the “>” symbol (the greater-than sign here is the “redirector”) and then the filename to the end of the `dir` command like so:

```
dir > DirOutput.txt
```

We can use another command called **type** to view the contents of a file. Try it out below to check the newly created DirOutput.txt file.

```
type DirOutput.txt
```

Suppose you wanted to record the contents of your directory and all its subdirectories so that, at a later date, you could tell which files existed at this time. Remember adding the switch `/S` to the `dir` command gives us this information. To capture its output in a file with this month, type

```
dir /S > oldDirInfo.txt
```

--where the file should be named to match the current month.

**Viewing files.**

You can also view the contents of this text file using the **type** command. Enter

```
type oldDirInfo.txt
```

-- to see the contents of this file. However, this only works for text files! Try displaying an Excel or Word file such as one of your excel exercise files. (You will need to enclose the file name in quotes if it contains a space.) You will see only garbage. Recall from the lecture that an Excel document file is stored in a file format that can only be interpreted by the Excel application program.

**Deleting files, directories or an entire directory tree.**

The **del** command deletes files, and the **rmdir** command deletes directories. Make sure you are in the directory which contains the file oldDirInfo.txt, and type

```
del oldDir*
```

-- using the wildcard character “*” to avoid typing the full filename. Change directories to the PLANTS directory and type

```
rmdir HERBS
```

-- which should delete the HERBS directory. Try the command
rmdir TRESSES
-- this fails because the TRESSES directory is not empty. You could go into TRESSES and clean it out. Instead, type

rmdir /S TRESSES
-- and answer Y at the prompt. As you can see, the /S switch for the rmdir command is very powerful, and with great power comes great responsibility. This command must be used with great caution since it can delete many things at once.

Writing a script file.
Let’s suppose that you need to re-create the directories TRESSES, PECAN, and OAK on a regular basis. You can automate this task by creating a script file. Launch the Notepad program by using the Search box. Type the following into Notepad, and remember to replace your directory with the name of your actual directory:

cd "your directory"
cd PLANTS
mkdir TRESSES
mkdir TRESSES\OAK
mkdir TRESSES\PECAN
cd ..
cd ..

Save this file as makedirs.bat in your I drive (make sure you Save as type: All Files (*.*) so it is saved as a .bat file not a .txt file) and exit from Notepad. The .bat file extension represents a batch file, another name for a script. To execute the script, go back to the command prompt console and change directories to where you saved your makedirs.bat file. Then type the name of the batch file:

makedirs
-- this will automatically create the TRESSES, OAK, and PECAN directories for you! Check it with the tree command.

tree

Windows maintains a collection of data available to all running programs, called the “environment.” For example, a program might need to know the special directory set aside for storing application data. This variable is called APPDATA. To find out the value of this variable, use the echo command. Type

echo %appdata%

-- note that the variable name is not case-sensitive, but must be enclosed in percent signs.

Use the echo command to find out the values of the following environment variables:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>PATH</td>
<td>List of directories that the command processor will search to find a command</td>
</tr>
<tr>
<td>COMSPEC</td>
<td>The command processor executable file that you are using now.</td>
</tr>
<tr>
<td>HOMEPATH</td>
<td>The path to your home directory</td>
</tr>
<tr>
<td>PROCESSOR_IDENTIFIER</td>
<td>Description of the CPU on this computer</td>
</tr>
<tr>
<td>WINDIR</td>
<td>The directory where the OS is located</td>
</tr>
</tbody>
</table>
Now let’s try a few commands to investigate the hidden resources of your system. Use the command

```
  driverquery
```

to get a list of all the drivers installed on your system. Yikes, there are lots of drivers installed!

**Useful networking commands.**

Use the command

```
  ipconfig /all
```

to see the computer’s IP address and other info that would be interesting if you knew what it all meant! Don’t worry; we’ll be explaining this later on. This command is useful for debugging network problems. Here are some other useful commands to try:

```
  ping cs.uga.edu
```

-the **ping** command sends a probe message out to the remote computer, to see if it is possible to connect, and if so how long it takes for the response to return. You should find that the response is so quick that it is recorded as less than 1 ms (milliseconds). Now try using the ping command with the address **in.pool.ntp.org**, a computer located in India. This time the response will be a lot longer!

```
  ping in.pool.ntp.org
```

Next, try the tracert ("trace root") command.

```
  tracert elc.uga.edu
```

- this command traces the route that packets travel to find the remote computer webct.uga.edu. The first hop might be similar to 128.192.something, which is a computer in the CS department; however, keep in mind that networks and computer change so this IP address may be different. The other hops on the route are part of the UGA networking infrastructure. This clever command works by sending out a whole series of probes. Each probe makes it one hop further before being returned.

**Stopping a runaway command.**

Use the **tracert** command to find the path to a distant computer, **in.pool.ntp.org**. You should see the first few hops appear quickly. After that, the display slows down. This trace could go on a long time, so do you remember how we stop it? Hold down the **Ctrl key and press the C key**. This **Ctrl+C** sequence interrupts a stuck program. It’s a useful trick to remember!

**Yes, there’s even a help command.**

Want a big list of commands? Type

```
  help
```

and you get a big honking list of commands you can play with on this computer.

This command outputs a long list of information about your computer:

```
  systeminfo
```

Check out some of the information you receive. (You probably have to scroll up a long ways). Look at what the computer is called (the Host Name), check out the last time your computer has been rebooted (System Boot Time), check out the processor speed (Processor(s)).
The `where` command.

This is a cool little command that will allow you to find the locations of some files or programs on your computer. For example, to find where notepad is located, type in

```cmd
where notepad
```

Take a look at the help text for the `where` command. Notice in the description that by default the `where` command searches along the current directory and the PATH variable locations. This works since java.exe is in our PATH variable.

Notice it can even work with the wildcard character ("*"). For example, let’s say you forgot where you saved your Lab instruction files. To find all files that start with “Lab” type

```cmd
where Lab*
```

Depending on where your Lab files are located and which directory you are currently in, **this might fail**. The files are, however, probably on your I: drive. Notice to recursively search a location, add the flag `/R` and then the place to search (check out the example in the help text). Try searching for any saved Lab file on your I: drive using the recursive flag and specify the starting location as your I: drive.

```cmd
where /R I:\ Lab*
```

The `prompt` command.

To change the prompt text, we use a command called Prompt (yes it gets confusing, but the command Prompt changes the text that appears before the blinking “_” cursor). To find out how to use the command, type

```cmd
Prompt /?
```

in the window. This time, we get a list of options for changing our command prompt. Remember the special notation

```
PROMPT [text]
```

means that the **command word** `prompt` is mandatory, but the text after the command word is optional. To see how this works, type

```cmd
Prompt Hello, dear
```

--you will see the Command prompt change to “Hello, dear”!

This is actually pretty ugly because it’s hard to tell where the prompt ends and your command begins. We typically like to end our prompt with the “>” sign. Note from the help output that for this command, the “>” sign has a special symbol, $G. Try entering the command this way:

```cmd
Prompt Hello, dear$G
```

--this time you will see a nicer looking prompt.

To return the prompt back to normal just type:

```cmd
prompt
```

Now that you have a feel for how the options work, use the prompt command to create a prompt which looks like this:

```cmd
You're late! It's 08:15:13.56>
```
--where the time shown is the actual **current time** (look for its symbol in the prompt command’s help text). Once you get it figured out, hit enter a few times and you will see the time change between prompts.

OK, that’s enough irritating experiments for now. Let’s restore the prompt back to normal.

**Congratulations!** This completes your introduction to the Windows command prompt. Of course there are greater depths for the truly curious. You should be comfortable using the fundamental commands of **dir**, **cd**, and **mkdir**, and you should also be comfortable using the help menus for the commands. The lab quiz will test your knowledge over these fundamental commands and some new commands, but don’t worry, new commands can be easily be found using your favorite search engine, and you’ll have the help commands to assist you on the lab quiz. If you’re curious to further explore the world of the prompt, then go ahead and play with a few new commands you find online. Some may work on these lab machines, but some may not due to access restrictions on user accounts.