Windows Command Prompt in 15 Minutes

There are only a few Command Prompt commands that you will need to master to make it through COS 126. The following is a brief tutorial covering the most important ones.

What is Command Prompt?

The Command Prompt program allows you to work in an environment that looks more like a traditional operating system as opposed to the icon based Windows environment. In Command Prompt, you will use your keyboard. You won't use your mouse at all. Command Prompt works at a lower level than Windows. This means that you will have more control over the machine. The disadvantage is that it is less user-friendly.

You will need the command prompt in COS 126 to compile and execute your Java programs. Learning the Command Prompt also provides a gradual transition to Unix and Linux systems, which are prevalent in science, engineering, and industry.

To launch Command Prompt select *Start -> Run* and type `cmd` in the box.

The Command Prompt shows up as a black terminal window. The *command prompt* should look something like:

```
C:\>
```

This is where you type commands. The boldface type below (that follows the command prompt) is what you should type as you work through this tutorial. Windows does not care if you use upper or lower case. That means that command `cd` is the same as `CD`. It also means that, in Windows, file `HelloWorld.java` is the same as `helloworld.java`. This is NOT true in the system to which you will be submitting your files. Be very careful!!!

Some Useful Commands

- **javac:** To compile a Java program, use the `javac` command. Your program should compile without any errors or warnings (or if there are warnings be absolutely sure that they do not indicate a flaw in your program).

  ```
  C:\>javac HelloWorld.java
  ```

- **java:** To run a successfully compiled Java program, use the `java` command.

  ```
  C:\>java HelloWorld
  ```

- **more:** Display the contents of a file one screenful at a time.

  ```
  C:\>more HelloWorld.java
  ```

- **exit:** Exit the Command Prompt program and close the terminal window.

  ```
  C:\>exit
  ```

Working with Files and Directories

You can also use Command Prompt commands to organize files into a directory hierarchy. These commands are equivalent to corresponding commands that you access via the Windows point-and-click interface. It is useful to be familiar with both interfaces for managing files.

- **dir:** To view the contents of a directory, type `dir`. This command will list all the files and directories within the current directory. It is analogous to clicking on a Windows folder to see what's inside.

  ```
  C:\> dir
   Volume in drive C has no label.
   Volume Serial Number is C8C7-B0CD

   Directory of C:\

  10/26/2004  01:36 PM         0 AUTOEXEC.BAT
  10/26/2004  01:36 PM         0 CONFIG.SYS
  02/10/2005  01:36 PM       126 HelloWorld.java
  12/09/2004  12:11 AM    DIR    Documents and Settings
  02/10/2005  08:59 PM    DIR    introcs
  11/02/2004  08:31 PM    DIR    j2sdk1.4.2_06
  12/29/2004  07:15 PM    DIR    Program Files
  01/13/2005  07:33 AM    DIR    WINDOWS
                 3 File(s)            126 bytes
                 5 Dir(s)  32,551,940,096 bytes free
  ```

  There are 7 items in this directory. Some of them are files, like `HelloWorld.java`. Others are directories, like `introcs`.

- **cd:** It is frequently useful to know in which directory you are currently working. In order to find out, type `cd` at the command prompt.

  ```
  C:\> cd
  C:\
  ```

  To change directories, use the `cd` command with the name of a directory.

  ```
  C:\> cd introcs
  ```

  Now, the command prompt will be:

  ```
  C:\introcs>
  ```

  To see what is in this directory type:

  ```
  C:\introcs> dir
   Volume in drive C has no label.
   Volume Serial Number is C8C7-B0CD

   Directory of C:\introcs

  02/10/2005  08:59 PM    DIR            .
  02/10/2005  08:59 PM    DIR            ..
  02/03/2005  11:53 PM               126 HelloWorld.java
  01/17/2005  01:16 AM               256 readme.txt
                 2 File(s)            382 bytes
                 2 Dir(s)
  ```

  To return to the previous directory, use the `cd` command, but this time followed by a space and two periods.

  ```
  C:\introcs> cd ..
  C:\>
  ```

- **mkdir:** To create a new directory, use the command mkdir. The following command creates a directory named `hello`, which you can use to to store all of your files associated with the Hello World assignment.

  ```
  C:\introcs> mkdir hello
  ```

  To see that it actually worked, use the `dir` command.

  ```
  C:\introcs> dir
   Volume in drive C has no label.
   Volume Serial Number is C8C7-B0CD

   Directory of C:\introcs

  02/10/2005  08:59 PM    DIR            .
  02/10/2005  08:59 PM    DIR            ..
  02/11/2005  02:53 PM    DIR            hello
  02/03/2005  11:53 PM               126 HelloWorld.java
  01/17/2005  01:16 AM               256 readme.txt
                 2 File(s)            382 bytes
                 3 Dir(s)
  ```

- **move:** Now, move the two files `HelloWorld.java` and `readme.txt` into the hello directory using the `move` command.

  ```
  C:\introcs> move HelloWorld.java hello
  C:\introcs> move readme.txt hello
  C:\introcs> dir
   Volume in drive C has no label.
   Volume Serial Number is C8C7-B0CD

   Directory of C:\introcs

  02/10/2005  08:59 PM    DIR            .
  02/10/2005  08:59 PM    DIR            ..
  02/11/2005  02:53 PM    DIR            hello
                 0 File(s)              0 bytes
                 3 Dir(s)
  ```

  The two files are no longer visible from the current directory.

  To access the two files, change directories with the `cd` command. Then use the `dir` command to see what is in this new directory.

  ```
  C:\introcs> cd hello
  C:\introcs\hello> dir
   Volume in drive C has no label.
   Volume Serial Number is C8C7-B0CD

   Directory of C:\introcs\hello

  02/10/2005  08:59 PM    DIR            .
  02/10/2005  08:59 PM    DIR            ..
  02/03/2005  11:53 PM               126 HelloWorld.java
  01/17/2005  01:16 AM               256 readme.txt
                 2 File(s)            382 bytes
                 2 Dir(s)
  ```

  You can also use `move` to rename a file. Simply specify a new filename instead of a directory name. Suppose you accidentally messed up the upper and lower case and had saved HelloWorld.java as helloworld.java. Use two move commands to fix it.

  ```
  C:\introcs\hello> dir
   Volume in drive C has no label.
   Volume Serial Number is C8C7-B0CD

   Directory of C:\introcs\hello

  02/10/2005  08:59 PM    DIR            .
  02/10/2005  08:59 PM    DIR            ..
  02/03/2005  11:53 PM               126 helloworld.java
  01/17/2005  01:16 AM               256 readme.txt
                 2 File(s)            382 bytes
                 2 Dir(s)

  C:\introcs\hello> move helloworld.java temp.java
  C:\introcs\hello> move temp.java HelloWorld.java
  C:\introcs\hello> dir
   Volume in drive C has no label.
   Volume Serial Number is C8C7-B0CD

   Directory of C:\introcs\hello

  02/10/2005  08:59 PM    DIR            .
  02/10/2005  08:59 PM    DIR            ..
  02/03/2005  11:53 PM               126 HelloWorld.java
  01/17/2005  01:16 AM               256 readme.txt
                 2 File(s)            382 bytes
                 2 Dir(s)
  ```

  It takes two moves because Windows won't let you move to an already existing filename and, to Windows, `helloworld.java` is the same as `HelloWorld.java`.

- **copy:** To make a copy of a file, use the `copy` command. The following command creates a backup copy of our HelloWorld.java program. This is especially useful when you modify a working program, but might want to revert back to the original version if your modifications don't succeed.

  ```
  C:\introcs\hello> copy HelloWorld.java HelloWorld.bak
  C:\introcs\hello> dir
   Volume in drive C has no label.
   Volume Serial Number is C8C7-B0CD

   Directory of C:\introcs\hello

  02/10/2005  08:59 PM    DIR            .
  02/10/2005  08:59 PM    DIR            ..
  02/03/2005  11:53 PM               126 HelloWorld.java
  01/17/2005  01:16 AM               256 readme.txt
                 2 File(s)            382 bytes
                 3 Dir(s)
  ```

- **del:** Subsequently, you might want to clean up useless files. The `del` command deletes a file.

  ```
  C:\introcs\hello> del HelloWorld.bak
  C:\introcs\hello> dir
   Volume in drive C has no label.
   Volume Serial Number is C8C7-B0CD

   Directory of C:\introcs

  02/10/2005  08:59 PM    DIR            .
  02/10/2005  08:59 PM    DIR            ..
  02/03/2005  11:53 PM               126 HelloWorld.java
  01/17/2005  01:16 AM               256 readme.txt
                 2 File(s)            382 bytes
                 3 Dir(s)
  ```

  WARNING: When you revise a file in jEdit, the jEdit program will automatically save a backup copy of your original file in the same directory. The name of the backup file will be the name of the original file with a ~ at the end. When you submit your program be careful to submit `HelloWorld.java` and not `HelloWorld.java~` which is an old version of the file and has the wrong name.

- **wildcards:** You can also apply the `copy`, `del`, and `move` commands to several files (or directories) at once. To create a new directory called `loops`, and copy all of the files in the `hello` directory `C:\introcs\hello\` into this newly created directory type:

  ```
  C:\introcs> mkdir loops
  C:\introcs> copy c:\introcs\hello\* loops
  ```

  Here the `*` matches all files in the `C:\introcs\hello` directory. It copies them to your newly created `loops` directory.

Redirection

Two important abstractions in Command Prompt are *standard input* and *standard output*. By default standard input is your keyboard, and standard output is your computer screen. For example, in Assignment 1, we write a program `CenterofMass.java` that reads input using StdIn.java and writes output using System.out.println(). To run our program, the user types the command "`java CenterofMass`" and enters double type values in triplets: `xposition yposition mass` from the keyboard. The results appear in the terminal window.

```
C:\introcs\loops> java CenterofMass
0 0 10
1 1 10
0.5  0.5  20
```

- **Redirecting standard input.** As an alternative, we can create a file that consists of the same six input numbers. Using a text editor (like jEdit), create a file named `input.txt`, and type in the six numbers. After saving the file in the `loops` directory, type the following command to verify that you entered the integers correctly:

  ```
  C:\introcs\loops> more input.txt
  0 0 10
  1 1 10
  ```

  Then to read the integers from the file instead of the keyboard, we use the redirection symbol "<".

  ```
  C:\introcs\loops> java CenterofMass < input.txt
  0.5  0.5  20
  ```

  This produces exactly the same result as if the user had typed the numbers, except that the user has no opportunity to enter numbers from the keyboard. This is especially useful for two reasons. First, if there are lots of input values (there are over 700 inputs for Assignment 2) it would be tedious to retype them in each time we run our program. Second, it allows programs to be automated, without waiting for user interaction. This means that your grader can process your homework programs without typing in the input values by hand each time

- **Redirecting standard output.** Similarly it is possible to redirect the output to a file instead of to the screen. Continuing with the same example, if we want to save the output permanently, we can use the output redirection symbol '>'.

  ```
  C:\introcs\loops> java CenterofMass > output.txt
  0 0 10
  1 1 10
  ```

  The user still types in the input values from the keyboard, but instead of sending the output to the screen, it is sent to the file named `output.txt`. Note that all printf output is sent to the file, even the statement that tells the user what to do. Be careful, if the file `output.txt` already exists, it will be overwritten. (To append use ">>" instead.)

  ```
  phoenix.Princeton.EDU$ more output.txt
  ```

- **Redirecting standard input and standard output.** It is often useful to use both redirection operations simultaneously.

  ```
  C:\introcs\loops> java CenterofMass < input.txt > output2.txt
  ```

  After executing this command, no output appears on the screen, but the file output2.txt now contains exactly the same data as `output.txt` above.

Piping

Another useful abstraction is *piping*. Piping is when the output of one program is used as the input of another program. For example, suppose we want to view the output of a program, but there is so much that it whizzes by on the screen too fast to read. (The program `RandInts.java` prints out a bunch of random integers.) One possible way to accomplish this is to type the following two commands.

```
C:\introcs> java RandInts > temp.txt
C:\introcs> more < temp.txt
```

Note that `more` will work by redirecting the file temp.txt to standard input (as is done here) or by simply using the filename (as is done at the beginning of the document). Instead, we could do this in one line using the pipe symbol '|'

```
C:\introcs> java RandInts | more
```

This is often useful when debugging a program, especially if your program goes into an infinite loop and you want to see the first few values that it prints.

Written by Jake Brenner, Donna Gabai and Kevin Wayne.