CIS-350
INFRASTRUCTURE TECHNOLOGIES

LAB #2

Due date: **See Blackboard**

Objectives: Learn about **Windows Command Prompt – Batch/Script Files**

This is not a group lab. You should do this lab individually.

A typical work pattern for an inexperienced user is to issue one command at a time, obtain the system's response, and then type the next command. This is what you did in Lab 1. With **batch files**, which are also called **batch programs** or **scripts**, you can simplify routine or repetitive tasks. Let's say you have to issue a group of the same commands several times a week/day, every hour or at a specified time to accomplish a certain task. Instead of typing them in and executing one command at a time, you may group commands in a batch/script file, save the file under a user-defined name, and then type the name of the file on the command line prompt and press Enter. The cmd.exe runs the commands in the batch/script file sequentially as they appear in the file.

A batch file is an unformatted text file that contains one or more commands and has a .bat or .cmd file name extension. You can include any command in a batch file. Certain commands, such as **FOR**, **GOTO**, and **IF**, enable you to do conditional processing of the commands in the batch file. For example, the **IF** command carries out a command based on the results of a condition. Other commands allow you to control input and output and call other batch files.

In this lab you will be exposed to several simple and more complicated batch files (extension **.bat**). Note that the command language used in the batch files shown in this lab allows one to pass on parameters/arguments to a batch file, call other batch files from within a batch file, implement loops and if-then-else statements/conditions in a batch file, and allow the user to input his/her choice and execute it. For the latter, see for example the CREATEMENU batch file on pages 6-7.

Another possibility would be to use Windows PowerShell for a scripting language and a full command-line experience. Windows PowerShell is Microsoft's task automation and configuration management framework, consisting of a command-line shell and associated scripting language built on .NET Framework. Windows PowerShell had been in development for several years by Microsoft. Windows PowerShell 1.0 was released in 2006. Then the add-on to Windows 2008 Server was created. The Windows 7 release, in 2009, is version 2.0 and includes some much-anticipated functionality, not the least of which is remote management. PowerShell 3.0 with its 2400 command-lets was released in 2012 (http://redmondmag.com/articles/2012/09/06/powershell-3.0-released.aspx). Also, see http://en.wikipedia.org/wiki/Windows_PowerShell. Learning Windows PowerShell is beyond the scope of this course, however.

In Unix/Linux batch files are called script files. You will see several of them in Labs 4 and 5.

Follow this list of explanations and activities. Do Lab 2 on the same thumb drive as you did Lab 1. The thumb drive on your computer will probably be referenced as the E drive. I will use the same naming conventions as in Lab 1.

***Go to the command prompt*.**

From the Windows command prompt "C:\Users\jmzura01>" (or another), type

```
E:                    <Enter>.
```

If your thumb drive is represented by a different letter, change E to that letter.

You should see the E:\> prompt.

To create Lab2 directory in the root directory of the thumb drive, type:

```
MD Lab2               <Enter>
```

Now to traverse to directory Lab2, type:

```
CD \Lab2              <Enter>
```

Your prompt should read "**E:\Lab2>**".

You will be saving **all** batch files you will create in this lab there, i.e., in the Lab2 subdirectory.

To create batch files in this assignment you may use program *edit* in Windows 7. It opens the screen editor. If *edit* is not available in Windows 7, use *Notepad*.

PARAMETERS - Parameters may be used in any place in a batch file where a parameter would normally be used as a part of the Windows command being run. Markers are used within the batch file to signify which parameter goes where. Markers are comprised of a percent sign (%) and a single digit between 0 and 9.

(1) Create a batch file named REPEAT.BAT

At the E:\Lab2> prompt, type:

EDIT REPEAT.BAT <Enter> or **NOTEPAD REPEAT.BAT** <Enter>

Put the following command in the file:

```
ECHO  %0  %1  %2  %3
```

Open the File pull-down menu, save the file as REPEAT.BAT, and exit the editor or Notepad.

To run it, type:

```
REPEAT Red Blue Green         <Enter>
```

ECHO can show messages on the screen. Also, one can pass parameters to a batch file.

Four parameters are passed to the REPEAT batch file and are placed into the ECHO command in order received. Note an important point -- marker zero is assigned the name of the .bat file, i.e., REPEAT, in the form you typed it in. The parameters and markers were related as follows.

	REPEAT	Red	Blue	Green
ECHO	%0	%1	%2	%3

The message ECHO REPEAT Red Blue Green is displayed on the screen. The words REPEAT, Red, Blue, and Green substitute markers %0, %1, %2, and %3, respectively, in this order.

BATCH SUBCOMMANDS - In addition to the normal commands, batch files have their own subcommand structure. Here are 3 simple Batch subcommands.

a. ECHO – The ECHO command turns command display on/off or may display a message. You have already seen it send a message to the screen. It can help clear up the clutter on the screen when a batch file executes. Normally, all the batch file commands would show up on the screen as if you were typing them. This can get distracting. If you want to suppress the command echoing type **ECHO OFF** or to start echoing, type **ECHO ON**. For example, the command *ECHO I love New York* will display the message *I love New York* on the screen. To suppress the word *ECHO* itself from displaying, type **@ECHO OFF**.

b. REM - REMark can be used to send messages to the screen or simply to document some part of your batch file's operation.

c. PAUSE - This command is to stop the execution of the batch file until you press a key. This can allow you to perform a necessary task - like changing a disk or verifying errors – as the remaining parts of the batch files are executed. PAUSE will optionally display a message if you desire. Type and run this short batch file.

(2) Create a batch file named TRYECHO.BAT

In response to the DATE and TIME commands in TRYECHO.BAT, the system will display the current date and time and will prompt you to change them by displaying "Enter the new date: <mm-dd-yy>" and "Enter the new time: ". Ignore the prompts. Do not enter the new date and the new time. Each time you see the prompt, press Enter.

From the E:\Lab2> prompt, type:

EDIT TRYECHO.BAT <Enter> or **NOTEPAD TRYECHO.BAT** <Enter>

Put the following commands in the file:

```
ECHO OFF
DATE
ECHO THE ECHO IS OFF
TIME
ECHO ON
DATE
ECHO THE ECHO IS ON
TIME
```

Save the file as TRYECHO.BAT and exit the editor.

To run it, type:

TRYECHO <Enter>

When you run this, you should see how the ECHO command works.

Again, in response to the DATE and TIME commands in TRYECHO.BAT, the system will display the current date and time and will prompt you to change them by displaying "Enter the new date: <mm-dd-yy>" and "Enter the new time: ". Ignore the prompts. Do not enter the new date and the new time. Each time you see the prompt, press Enter.

(3) Create a batch file names REMPAUSE.BAT

From the E:\Lab2> prompt, type:

EDIT REMPAUSE.BAT <Enter> or **NOTEPAD REMPAUSE.BAT** <Enter>

Put the following commands in the file:

```
REM THIS WILL SHOW HOW REM WORKS.
PAUSE PUT ANY MESSAGE HERE.
REM THIS WILL SHOW PAUSE ALSO.
PAUSE
REM YOU ARE NOW DONE.
```

Save the file as REMPAUSE.BAT and exit the editor.

To run the batch file, type:

REMPAUSE <Enter>

CTRL-Break and **CTRL-C** terminate and **CTRL- S** pauses to resume an endless loop batch file. Try it. The next batch file will give you a chance to verify it.

Here are three more complicated batch commands to try.

a. GOTO - The format for this command is: GOTO LABEL where LABEL is a line in your batch file that starts with a colon (:) followed by up to an eight character name.

Here is an endless loop program.

(4) Create a batch file named LOOP.BAT

From the E:\Lab2> prompt, type:

EDIT LOOP.BAT <Enter> or **NOTEPAD LOOP.BAT** <Enter>

Put the following commands in the file:

```
:START
REM ... This is being typed by an endless loop.
GOTO START
```

Save the file as LOOP.BAT and exit the editor.

To run the batch file, type:

```
LOOP <Enter>
```

When executed, this file will continue to print the endless loop line and the GOTO command until you use **CTRL-Break** or **CTRL-C** to terminate it, or **CTRL- S** to suspend and resume it.

b. IF - This subcommand allows you to make decisions within your batch file. The format is:

```
If Condition Command
```

where,

Command = Any legal Windows command or batch file subcommand
Condition = One of three tests that yield true or false:

1. The ERRORLEVEL of a program.
2. Two strings are equivalent.
3. A file exists in the current directory.

Unlike programming languages, which allow many logical tests in an IF statement, the batch IF statement in the mentioned form is limited to only the three above.

Condition 1:

ERRORLEVEL is a number that indicates to Windows whether the last program run was successful. A zero (0) indicates a good run, anything above zero indicates an error condition.

Condition 2:

String comparison is indicated by double equal signs: String1 == String2 compares the two strings. This comparison is often used with parameters and markers to check for a particular entry. For example,

```
IF %1==XYZ GOTO GOOD
```

checks parameter one for XYZ and, if so, alters the sequence of program execution by jumping to label GOOD.

Condition 3:

The logical test checking for a file has the generic format: EXIST E:\PathName\Filename. You can use this test to check for a particular file LOOP.BAT exists on the E drive in the directory specified in the pathname. Pathnames were not allowed in MS-DOS, but they are allowed in Windows.

An example will follow.

c. One batch file may call another. This batch file can be used to establish a password for running a program. The batch file is named GO.BAT and calls the program named LOOP.BAT. (It could call any .COM file or application package.) This file attempts to demonstrate all the batch file commands you have learned.

(5) Create a batch file named GO.BAT

From the E:\Lab2> prompt, type

EDIT GO.BAT <Enter> or **NOTEPAD GO.BAT** <Enter>

Put the following commands in the file:

```
ECHO OFF
IF %1==XYZ GOTO GOOD
ECHO BAD PASSWORD... ENDING
GOTO END
:GOOD
ECHO YOU'RE OK...STARTING
IF EXIST E:\LAB2\LOOP.BAT ECHO YES THERE IS A LOOP.BAT FILE
PAUSE
REPEAT RED BLUE GREEN
:END
```

Save the file as GO.BAT and exit the editor.

Run this by typing GO ABC and also with GO XYZ. The parameter ABC or XYZ will be passed on to the batch file and replace the %1 marker/token.

Type:

GO ABC <Enter>

and then

GO XYZ <Enter>

One can see that the program takes two actions, i.e., two different segments of the program are executed depending on the parameter ABC or XYZ passed to the program. Also note that the program checks if batch file LOOP exists and executes batch file REPEAT. Note that file LOOP.BAT should be in the E:\Lab2 subdirectory. If your current drive is F, then LOOP.BAT should be in the F:\Lab2 subdirectory.

Here is another batch file which I would like you to try. I copied the batch file from http://www.sevenforums.com/tutorials/78083-batch-files-create-menu-execute-commands.html. The batch file creates a menu to execute several commands of your choice. You will create a similar script and run it on RedHat Linux in Lab 5. The link above also provides the description of the commands used in the batch file. So please read it thoroughly.

(6) Create a batch file named CREATEMENU.BAT

From the E:\Lab2> prompt, type

EDIT CREATEMENU.BAT <Enter> or **NOTEPAD CREATEMENU.BAT** <Enter>

Copy all the commands below, paste them into Edit or Notepad, and save a file as CREATEMENU.BAT.

```
ECHO OFF
CLS
:MENU
ECHO.
ECHO ...............................................
ECHO PRESS 1, 2 OR 3 to select your task, or 4 to EXIT.
ECHO ...............................................
ECHO.
ECHO 1 - Open Notepad
ECHO 2 - Open Calculator
ECHO 3 - Open Notepad AND Calculator
ECHO 4 - EXIT
ECHO.
SET /P M=Type 1, 2, 3, or 4 then press ENTER:
IF %M%==1 GOTO NOTE
IF %M%==2 GOTO CALC
IF %M%==3 GOTO BOTH
IF %M%==4 GOTO EOF
:NOTE
cd %windir%\system32\notepad.exe
start notepad.exe
GOTO MENU
:CALC
cd %windir%\system32\calc.exe
start calc.exe
GOTO MENU
:BOTH
cd %windir%\system32\notepad.exe
start notepad.exe
cd %windir%\system32\calc.exe
start calc.exe
GOTO MENU
:EOF
```

Run the batch file.

In the next batch file, we will use several simple commands which were introduced in Lab 1. We will chose several commands from the list below and use them in the batch file.

DIR
TREE
VER to find the version of Windows in use
COPY
DATE
TIME
MD to create one or more new subdirectories
CD to change between subdirectories
TYPE to display file content
RD to remove one or more of the directories
SORT by piping a file in and sending a file out. Display the file contents before sorting and after sorting
Wild cards

(7) Create a batch file named MYBATCH.BAT

From the E:\Lab2> prompt, type

EDIT MYBATCH.BAT <Enter> or **NOTEPAD MYBATCH.BAT** <Enter>

Copy all the commands below, paste them into Edit or Notepad, and save a file as MYBATCH.BAT.

```
ECHO OFF
REM STARTING BATCH FILE
TREE /F
PAUSE
DIR
PAUSE
MD JOE MARY STACY
TREE /F
PAUSE
COPY *.BAT E:\LAB2\STACY
TREE /F
PAUSE
CD STACY
ERASE *.BAT
PAUSE
CD ..
TREE /F
PAUSE
RD JOE MARY STACY
TREE /F
PAUSE
REM CLOSING BATCH FILE
```

Run the batch file.

You should have the following seven batch files on your thumb drive in your Lab2 subdirectory: REPEAT, TRYECHO, REMPAUSE, LOOP, GO, CREATEMENU, and MYBATCH.

You should see E:\Lab2\ prompt so *Lab2* is your parent directory.

Type: **TREE /F > Lab2_Tree** <Enter>

Your *Lab2* directory structure and contents have been saved to file *Lab2_Tree*. You will paste this file into your Lab 2 Report,

or type: **TYPE Lab2_Tree** <Enter>

and use the *Alt-PrtScr* keys to capture the full screen output (full window) from command *TYPE Lab2_Tree* and paste that window into Lab 2 Report.

Turn in:

1. Hardcopy of Lab 2 Report. The template for the report is in \Assignments\Labs\Lab2 folder on Blackboard.