Windows Batch Files 10.1

IF – ELSE operations

GOTO and LABELS

IF Command

- Decision operations based on using IF-ELSE ideas are common in computing.

- In today’s lab, you will be looking at implementing Batch Files which make use of the IF-ELSE construct to control the output to the screen and also the flow of operations that will be performed in the Batch File.

IF Command, continued

- The IF command allows you to make decisions within a Batch File.

- The IF command should only really be used in Batch Files as it is the only location where results can easily be stored and multiple lines of operation used correctly

- **As you read through the following slides, type out all the examples yourself into a batch file and try them out/experiment with them.**

Format of the IF command is

```
IF Condition Command
```

- To check if two strings are equivalent:

  ```
  IF  "String1"=="String2" Command.
  ```

  This comparison is often used with parameters to check for a particular entry.
  Note: No spaces between quotes and equal symbols.

- Example:

  ```
  SET myvar=Hello World

  IF "%myvar%"=="Hello World" (
          ECHO message received
  )
  ```

Format of the IF command is

```
IF Condition Command
```

- To check if two strings are NOT equivalent:

```
IF  NOT "String1"=="String2" Command.
```

  This comparison is often used with environment variables or command line parameters to check for a particular entry

- Example:

```
SET myvar=Hello Earth

IF NOT "%myvar%"=="Hello World" (
   ECHO The strings are not the same.
)
```

Format of the IF command is

```
IF Condition Command
```

- To check if two strings are NOT equivalent:

  ```
  IF  NOT "String1"=="String2" Command.
  ```

  This comparison is often used with environment variables or command line parameters to check for a particular entry

- It is recommended to always quote both operands.
  (Recall: operands are what appear on either side of an operator. For example, the operator above is == and the operands are String1 and String2).

  Question:
  Why is it recommended to always quote both operands?

Format of the IF command is

```
IF Condition Command
```

- **Answer:**
  This avoids a bug when a variable doesn’t exist, which causes the operand to effectively disappear and cause a syntax error.

- **Remember, you are to type out all the examples yourself into batch file(s) and try them out/experiment with them.**

Form of the IF command is

```
IF Condition Command
```

- To check if a file exists in the current directory. The logical test checking for a file has the format:

```
IF EXIST filename
IF NOT EXIST filename
```

- Example:

```
IF EXIST "martin.txt" ECHO martin.txt exists
IF NOT EXIST "martin.txt" ECHO martin.txt does not exist
```

Form of the IF command is

```
IF Condition Command
```

- Although you may put the single command on the same line as the IF statement. E.g.:

```
IF EXIST "martin.txt" ECHO the file martin.txt exists
```

- It is good practice to always enclose the command inside an open and close parenthesis. E.g.:

```
IF EXIST "martin.txt" (
        ECHO the file martin.txt exists
)
```

- Why is this good practice?

Form of the IF command is

```
IF Condition Command
```

- So, following good practice: instead of typing

```
IF EXIST "martin.txt" ECHO martin.txt exists
IF NOT EXIST "martin.txt" ECHO martin.txt does not exist
```

- It is always better to use the following structure:

```
IF EXIST "martin.txt" (
        ECHO martin.txt exists
)  ELSE  (
        ECHO martin.txt does not exist
)
```

- Why is it good practice to use IF-ELSE instead of two IF statements in the example above?

Form of the IF command is

```
IF Condition Command
```

- If you want to execute MULTIPLE commands inside an IF statement, you MUST enclose the commands inside an open and close parenthesis. E.g.:

```
IF EXIST "martin.txt" (
        ECHO About to display contents of martin.txt
        TYPE martin.txt
)
```

IF, continued

- IF statements cannot be nested i.e. an IF statement cannot be called within another IF statement.
  - If this operation is required, another Batch File need to be called and what would have been the inner IF statement, is to be placed inside that second batch file.

GOTO and LABELS

- GOTO and LABELS allow you to control the flow and operations that are performed in a batch file.
- By using an IF command, it is possible to select locations in the Batch File that should be executed and other lines of code that may be skipped as required.
- This is similar in operation to JUMP commands that were shown in Computer Architecture.

Form of the GOTO statement

```
GOTO label
```

- `label` specifies a text string used in the batch program as a label.

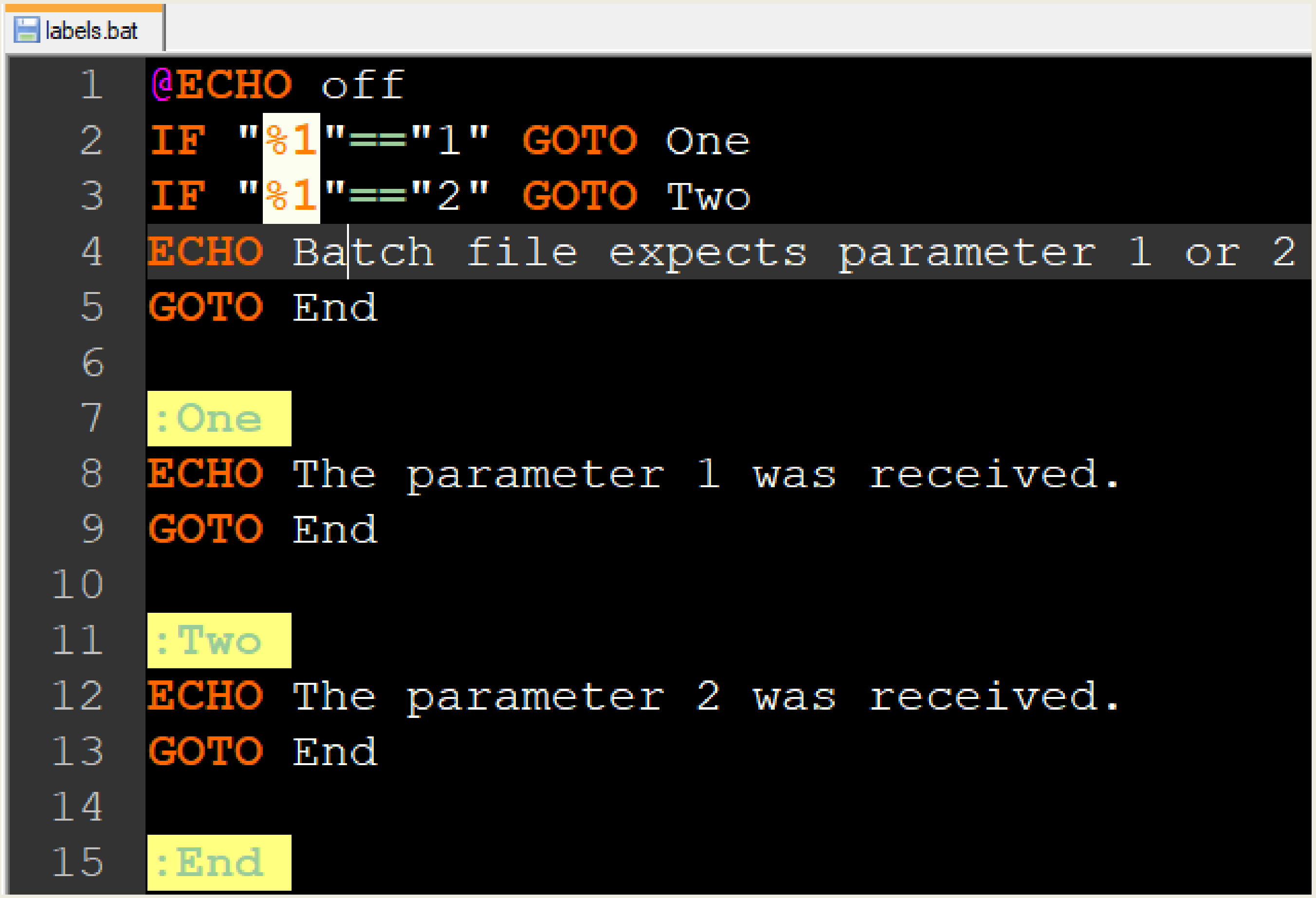
- The structure and use of the GOTO statement in a batch file requires the GOTO label command as part of the batch file, while a label destination point must be included in the batch file. The destination is written as :**label**

- Label is <u>not</u> a keyword. It is a text string that the programmer defines

- To make your batch file more readable, it is advisable to put the LABEL on a line by itself in the batch file.

Labels.bat

- Create a batch file called **labels.bat** and save it to the folder C:\users\<student-id>\OS1\lab10



```
@ECHO off
IF "%1"=="1" GOTO One
IF "%1"=="2" GOTO Two
ECHO Batch file expects parameter 1 or 2
GOTO End

:One
ECHO The parameter 1 was received.
GOTO End

:Two
ECHO The parameter 2 was received.
GOTO End

:End
```

Labels.bat

- Run the batch file as follows and observe the output generated:
  - C:\users\x00123456\OS1\lab10> labels.bat
  - C:\users\x00123456\OS1\lab10> labels.bat 1
  - C:\users\x00123456\OS1\lab10> labels.bat 2
  - C:\users\x00123456\OS1\lab10> labels.bat 3

- Replace line 5 with a blank line and save the batch file.
  Run the batch file as follows and observe the output generated.
  Explain the output generated.
  - C:\users\x00123456\OS1\lab10> labels.bat

- Restore the original batch file labels.bat. Then, replace line 9 with a blank line and save the batch file. Run the batch file as follows and explain the output.
  - C:\users\x00123456\OS1\lab10> labels.bat 1

EXAMPLE 1 using `IF EXIST`

In this next batch file you will use an IF command of the form:

```
IF EXIST filename COMMAND
```

You will also use the **CALL** command which <u>calls another batch file</u>.

Firstly, create a batch file called “todaysMenu.bat” and add some appropriate text.

A second batch file called “Menu.bat” should call this batch file and display the menu. You need to deal with the situation where the first batch file is unavailable. For this you will use **GOTO** and “labels”.

Example todaysMenu.bat

```
REM todaysMenu.bat
@ECHO OFF
ECHO TODAY'S MENU
ECHO Steak and Chips
ECHO Stir Fried Veg
ECHO Seafood Pasta
ECHO Pizza
```

Example Menu.bat

```
REM menu.bat
@ECHO OFF
IF EXIST todaysMenu.bat GOTO MENU
ECHO No Menu today
GOTO END

:MENU
CALL todaysMenu

:END
```

What are the labels in this batch file?

- The labels are :MENU and :END

- Recall the form of the GOTO command is

  `GOTO label`

  and In this batch file we see

  `GOTO MENU` and

  `GOTO END`

EXAMPLE 2
using IF to compare strings

Write a batch file called **openFile.bat**, where the user can type the following command on the command line

```
openFile notepad
```

If the wrong parameter is passed or mistyped by the user e.g. notpad, a message should be displayed informing the users of the mistake.

If the right command is executed, the notepad application should be run.

In your answer (the batch file): Use the IF statement of the form

```
IF "string1"=="string2" COMMAND
```

Where `string1` is a parameter, and the `COMMAND` is a "`GOTO`" command where notepad will be "`CALL`"ed.

You should also use GOTO statements and LABELs.
Do NOT look at the next slide until you have completed your attempt.

Example 2 Solution

```
REM openFile.bat
REM usage: openFile filename

@ECHO OFF
IF "%1"=="NOTEPAD" GOTO NOTEPAD
ECHO "%1" NOT Found
GOTO END

:NOTEPAD
CALL NOTEPAD

:END
```