



QUOTE

*Where all thinks alike,
No one thinks very Much...*

BATCH PROGRAMMING DECODED – II

[For Beginner]

By Kvc

karanveerchouhan@gmail.com

batchprogrammers.blogspot.in

Index

1. 10 Lines About this Part
2. Some more Commands of cmd
 - i. Prompt
 - ii. Attrib
 - iii. Call
 - iv. Color
 - v. Copy
 - vi. Move
 - vii. Rem
 - viii. Date
 - ix. Time
 - x. Del
 - xi. Mode
 - xii. Ren
 - xiii. Type
3. Important Shortcuts While executing commands in cmd
4. What is Batch scripting (or **Batch Programming**)
5. Variable handling in Batch
 - i. The 'set' command...
 - ii. How a variable looks like in batch??
 - iii. Arithmetic Operations on variables...
6. User interactions with Your Program ...(*Getting inputs from user*)
7. Some Pre-defined variables in batch ...(*The environment variables*)
8. Conditional statement – The If Statement

10 Lines about this Part

As this is **part – II**, I'm considering that you had read and understand the basic concepts of batch programming written in **part – I** of this book's series, and you are not a newbie.

As you already had a look on the index, and you know that you are going to learn essential basics of batch programming. While teaching about batch commands I'm considering the level of teaching in this part [**for beginners**]. If there are some advanced features of some commands...they will be revealed in next parts of this book's series...

Some more Commands of cmd

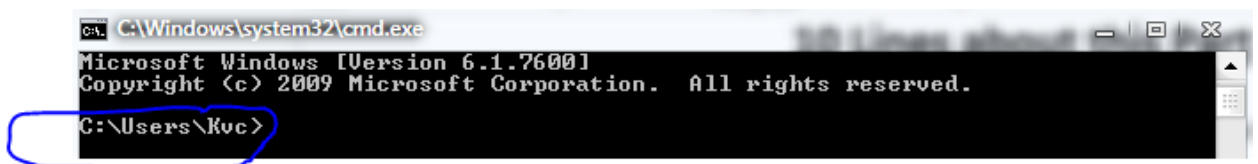
Tip: You can get help about any command in cmd just by typing '**command name**' and **"/?"** (Without Quotes) after that... The cmd will automatically show help for that command. (E.g. **Echo /?**)

1. The Prompt command

As when you open cmd, traditionally what you see on console is the version of cmd in 1st line, a Copyright © line by **Microsoft** and then a line showing your current path (**working directory**) after a blank line. Whatever you type in cmd console, the line telling your working directory automatically pops-up, even when you are just pressing enter without typing anything...

And do you remember that we used **echo off** to remove that line... Instead of removing that line, we can also modify it with anything we want... we can do this with the help of **prompt** command...

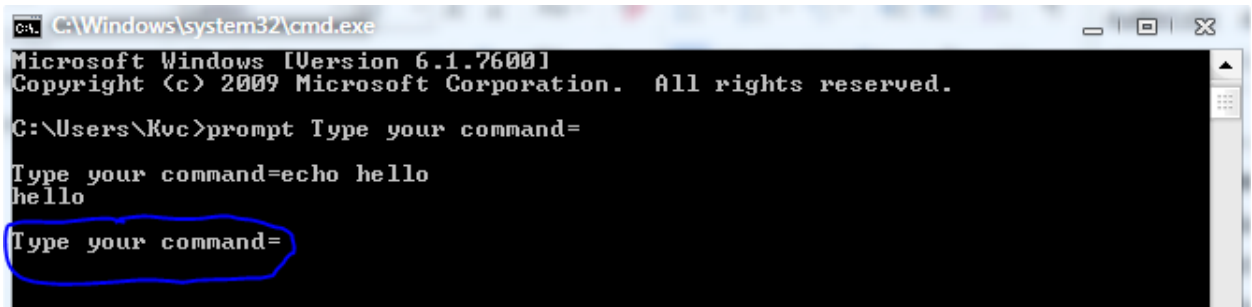
Traditional Look:



```
C:\Windows\system32\cmd.exe
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.

C:\Users\Roc>
```

Now try typing “**Prompt Type your command=**”, what happened??? Instead of showing current path, cmd starts showing ‘**type your command=**’ after every command execution.



The screenshot shows a Windows Command Prompt window titled "C:\Windows\system32\cmd.exe". The text inside the window is as follows:
Microsoft Windows [Version 6.1.7600]
Copyright (c) 2009 Microsoft Corporation. All rights reserved.
C:\Users\Kvc>prompt Type your command=
Type your command=echo hello
hello
Type your command=

The last line, "Type your command=", is circled in blue, illustrating that the prompt has changed from the default path to the text specified in the command.

Syntax: Prompt [text to show on cmd console]

If you type ‘**prompt /?**’ in cmd console then it will show many codes to show Special characters on console, but trust me...you don’t need to know these, just know the following **4**:

- \$H Backspace
- \$V Windows Version Number
- \$Q equal to sign (=)
- \$\$ Dollar Sign (\$)

As per my experience, you need **Backspace** only in some of the case, same for **windows version**, and as you noticed in above example- we had shown **equal to (=)** directly, so...there you go...

All rest characters you can show without using those Codes...

In most of the cases, while you try to display a special character in cmd console, then you just have to put a caret (^) before the character.

Have a look at following table

<u>Code (as per prompt command)</u>	<u>Special Character</u>	<u>Trick to Display [code]</u>
\$A	& (Ampersand)	Use caret [^&]
\$B	(pipe)	Use Caret [^]
\$C	((left parenthesis)	Simple or use caret [^(]

\$D	Current Date	Use simply [%date%]
\$E	← (escape code)	Use Alt+27 combo directly.
\$F) (right parenthesis)	Simple or use caret [^)]
\$G	> (greater than sign)	Use caret [^>]
\$L	< (less than sign)	use caret [^<]
\$N	Current Drive	use simply [%cd:~0,1%]
\$P	Current Drive and path	use simply [%cd%]
\$S	Space ()	simply write space b/w words
\$T	Current time	use simply [%time%]
\$_	Carriage return and line feed	use Alt+13 combo directly.

Tip: To get back the default prompt (**working directory line**), just type **prompt** simply and press enter. **Or try** `prompt %cd%^>`

Time for example:

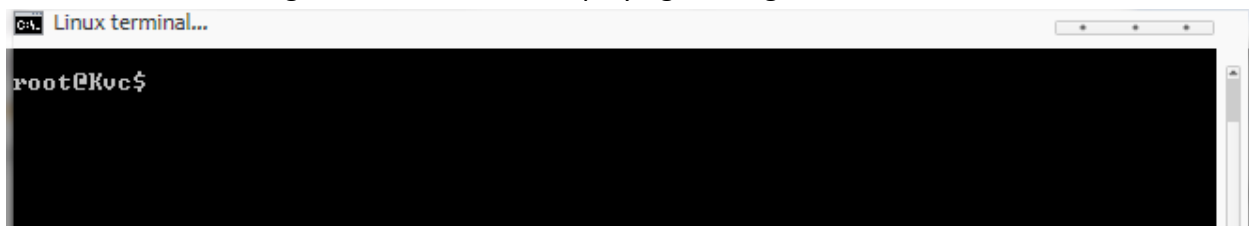
Q: WAP to change default prompt of cmd to Linux's terminal like prompt.

Code:

```
@echo off
Cls
Title Linux Terminal
Prompt root@%username%$$
Pause>nul
```

Explanation: (line by line)

- Line 1: Turning extra line's Display off
- Line 2: Clearing screen.
- Line 3: Changing Title of Cmd console.
- Line 4: Changing default prompt to our desired prompt. We will talk about '**%username%**' later in this book.
- Line 5: Pausing the console...with displaying nothing on console.



2. The **Attrib** Command

This command is used to change the attributes of files & folders, and learning this command is your 2nd step of being a super user... [1st step was decision of learning batch...hahaha :D]

You can **hide/unhide** a **file/folder** with this command...

Syntax: attrib [+/-r] [+/-s] [+/-a] [+/-h] [/s] [/d] [/L]

Where, +/- means, either put '+' or '-' .

By putting '+' you'll add the attribute to file, while '-' will remove corresponding attribute. But you can't use both + & - with the same attribute type.

And **r** represents 'Readonly attribute' (*file/folder becomes Readonly*)

And **s** represents 'System-file attribute' (*file/folder becomes SystemFile*)

And **a** represents 'Archive-file attribute' (*file/folder becomes Archivable*)

And **h** represents 'Hidden-file attribute' (*file/folder becomes Hidden*)

While,

/s to apply attribute settings to sub-files too...

/d to apply attributes to Folders too...

/L Work on the attributes of the Symbolic Link and also on the target

Time for example:

Q: WAP to create a folder named "kvc" and then to make it Read-only and hidden, and then make it visible again, then delete it.

Code:

```
@echo off
```

```
Cls
```

```
Md kvc
```

```
Echo. Press any key to make it hidden
```

Pause>nul

Attrib +r +h kvc

Echo. Press any key to make it visible

Pause>nul

Attrib -r -h kvc

Echo. Press any key to delete it

Pause>nul

Rd /s /q kvc

Pause>nul

3.The Call Command

Used to call other batch files from current batch file...like including Header files in **c program**...you don't need to write code of header files again, Just include them in your program...Same case here... if you had Some important batch files in your system...just call them from your current batch script.

Syntax: Call [full/relative path of file]

4.The Color command

As a human being, we are always astonished by colors around us. And we like colorful things...so, Microsoft provided a **color command** with the help of this command, you can change color of your cmd console...

Syntax: Color [ColorCode]

The **colorCode** consist of only two characters...without any space. The **first** character represents **background color**, and **second** character represents **Foreground color**.

Try 'Color /?' in cmd console...for color codes. Because repeating the same thing you already have in your system will just make my book lengthier not more than that, I think you can understand.

Tip: Most of the Batchers (**batch programmers**) across the world, likes the Color code **0a**, but I personally like **0c** color combination... Try '**Color 0a**' & '**color 0c**' in your cmd console and see the difference...

5. The Copy Command

As the name of command suggests...This command is used to copy files from one place to other...

Syntax: Copy [/Y] [Source file path] [destination file path]

Where,

/y If the same file presents in destination path, cmd will ask you for overwriting file.

But **/y** will override that confirmation & cmd will overwrite the file without asking...

6. The Move Command

As the name of command suggests...This command is used to move files from one place to other...

Syntax: Move [/Y] [Source file path] [destination file path]

Where,

/y If the same file presents in destination path, cmd will ask you for overwriting file.

But **/y** will override that confirmation & cmd will overwrite the file without asking...

7. The Rem Command

This command is used to give remarks /comments in batch scripts... These comments sometimes becomes very helpful in big messy code... :p

Syntax: Rem [Your comment here...]

Trick: As you know about declaring labels {for goto command}, just type two colons (::) and then after giving a space you can also give your comments...as Double Colons will also be ignored while executing commands.

6. The Date command

Displays or Sets the date of your system...

Syntax: Date [/t]

If you write **/t** after date, it will simply displays the current date, while without using **/t** will ask you for re-setting system's Date settings.

Give this command a try!!

7. The Time command

Displays or Sets the Time of your system...

Syntax: Time [/t]

If you write **/t** after Time, it will simply displays the current time, while without using **/t** will ask you for re-setting system's Time settings.

Similar to Date command...isn't it?

Time for Example:

Q: WAP to Show running time and date in cmd.

Code:

```
@echo off
```

```
Title My first temporary watch
```

```
:loop
```

```
Cls
```

```
Date /t
```

```
Time <nul
```

```
Goto loop
```

Code Explanation: (line by line)

Line 1: As usual, turning off display of unwanted text in console...

Line 2: Changing title of cmd console to my desired one...

- Line 3: Creating a Label named 'Loop'
- Line 4: Clearing the console
- Line 5: Displaying date (without prompting for re-setting system date)
- Line 6: Displaying time (*we can use '/t' here too... 'cuz the change on console will be visible only after 60 sec, when system's 1 minute updates, so giving **nul** as input by '<' operator [talk about it in next-part of this book],so it will not stop for our input for resetting time...*)
- Line 7: Sending the control back to Line 3, and whole process from line 3-7 repeats...

Warning: Please run the above code only for few seconds (5-10 sec.), as it may damage your system... I'm not responsible for any damage due to your carelessness... #kvc

8. The del command

As the name suggests, this command is used for deleting files ...

Caution: Files deleted by this command are deleted directly and don't expect those files in your recycle bin... :p

Syntax: Del [/s] [/q] [/a:**attr**] [/p] [path of file to delete]

Where, **/s** Deletes files in sub-directories too...

/Q Don't confirms the deletion, file will be directly deleted.

/A:attr deletes files with specific attributes...where **attr** can be
R, A, S, H (same meaning as in case of **attrib** command)

If you used '-'(minus) before attribute type,it will delete all the files which don't have that attribute.

/p Cmd asks before deleting each file.

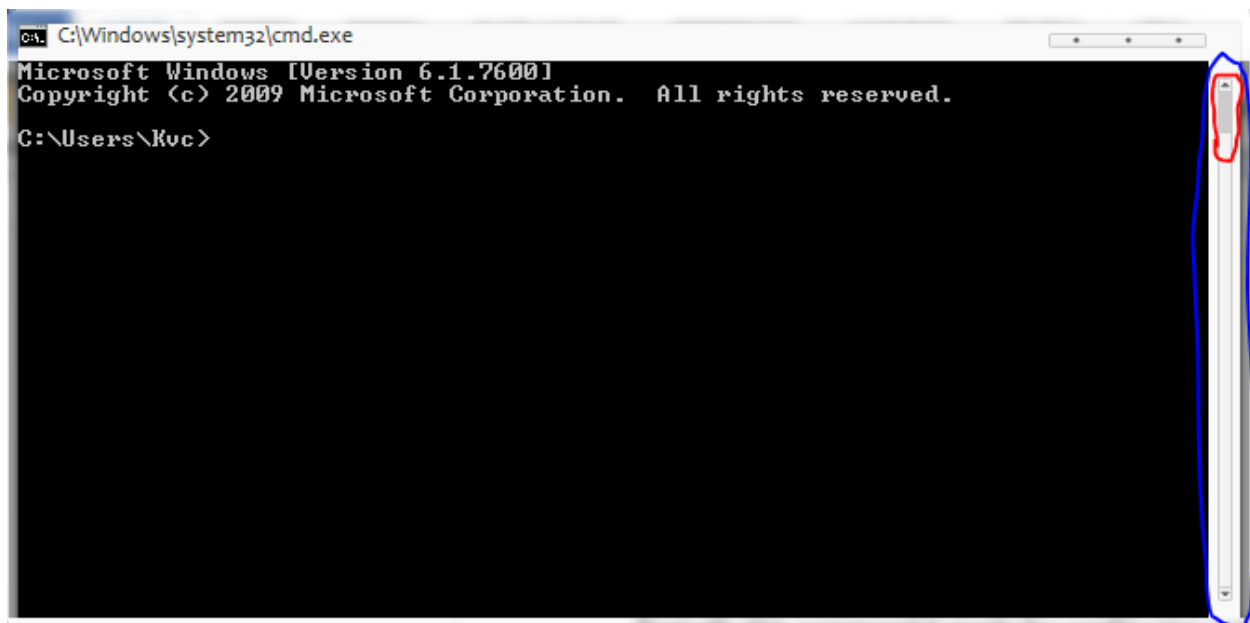
9. The Mode Command

Basically this command it made for configuring system devices, ports etc. But what I know about this command is that it gives you freedom of changing size of your cmd console. Lets have a look...

Syntax: Mode [Columns],[Rows]

Notice: Don't forget the **comma** between **columns nos.** and **row nos.**

Consider your cmd console as a big 2D matrix board, like a chess board where we have 8x8 matrix. Here in cmd we have 300x80 matrix by default... where 80 are columns and 300 are rows...that's why we always see a small scroll bar in right of our cmd console. Because at one time we can only see 25 rows of console.



Just look at the size of **scroll position teller...** (Surrounded by red Block), I really don't know what this thing is called, but I think you got it!!!

And do remember that 80 and 300 are not pixels as in our default GUI environment we consider. Here 80 are no. of characters that can be in one columns, and we have 300 rows.

Give it a try... Open cmd and type 'mode 80,25' ...what happened??

the change seems to be nothing, but if you have noticed carefully, then see scroll bar at right of your cmd console is vanished... also experiment with different numbers...

10. The Ren Command

This command is used for Rename files/ folders.

Syntax: Ren [path of file] [new name of the file]

Consider, don't give the full path again in 2nd parameter, there we have to declare only new name of the file.

11. The Type Command

Displays the content of text files on console... (You can't write the content, but can see...)

Syntax: Type [Path of file]

Important Shortcuts While executing commands in cmd

1. If you are roaming through folders, and some of the folders in your path got very long names as in case of names of **movies folders, music folders** etc. and you want to change your Working directory to that folder, but also don't want to write Such long names...so just type your command, in this case it is 'CD' and then type first 2 or 3 letters of the

folder name and press **tab-key**, if you don't get name of desired folder on console...keep on pressing tab...until you get it.

If you mistakenly passed your desired folder, don't worry, just press **Shift+tab** combo to get previous suggestion again.

2. Use **arrow keys** to again execute previously executed commands, don't bother to type long commands again and again.
3. Use **ESC key** to clear what you have typed to clear input field yet.

What is Batch scripting (or Batch Programming)

Batch Scripts may be roughly defined as the collection of cmd commands in one text file with a special file-extensions as **.Bat** or **.Cmd**.

What we had done till now in '**Time for example**' section...were also examples of batch Programs or scripts.

The above Sub-topic came to my mind when I was typing this part - II, and I thought that this should also be in their beginner's part to make this book more Generalize.

Variable handling in Batch

I think you already know what a variable is? , because this is what I had assumed before writing this Book, that you have knowledge of some basic terms of computer programming, (*try to remember about '**10 lines about this book**' topic in part-I of this book*)

Playing with variables will make your program **dynamic**... and if your memory is not so much weak as mine, then you should remember that we had made some **static programs** in Part-I. Which means in any of the conditions, those program will not change their output. But now we gonna learn to make programs that can change output or we just gonna learn to make our programs more intelligent...

*i. The **Set** command*

It is the most useful command in cmd, this is used for setting the value of variables...with the help of this command you can make your program fully **dynamic** and much more advanced...

Syntax: Set [VariableName]=[VariableValue]

E.g.: Set a=1

Where, 'a' is the name of variable...and it is assigned to value '1'.

Now you know, how to declare a variable in batch programs...and unlike C or any other programming languages, you can declare any variable anywhere in your batch Program. **(Same as in C++)**

ii. How A Variable Looks like in Batch??

Till now, we have learnt to declare a variable in batch, but don't know how to display its value on console or do anything else with its value, in our program...

So, here's the solution to this problem; a variable in batch is always surrounded by **percent (%)** symbols... a variable in any Batch file generally looks like: **%VariableName%**

If you write anything surrounded by '%' in cmd, cmd treats it like a variable and its value is replaced instead of '%varName%' whatever be the value of that variable, but if you didn't declared that variable and try to show its value using % symbol, then cmd simply treats %varname% as a string...and displays it as %varname%, because it is not being initialized by any value....

Unlike C, Batch will not show any Garbage value...

Time for example: (making of a Semi-Dynamic Program)

Q: WAP to show use of variables in batch.

Code:

@echo off

Cls

Set myVariable=Kvc

Rem now, the variable 'myVariable' is declared. And displaying its value in next line...

Echo. Value of myVariable is: %myvariable%

Pause>nul

Output : 

iii. Arithmetic Operations on variables

What is the use of variables, if you can't perform mathematical operations on them? As we already know that Mathematics is the language that is understandable anywhere in the universe... so, we should better know how to apply mathematical operations on variables...

You can apply mathematical operations like, +, -, *, /, % (modulus) etc. by using only set command. You just have to use a *new parameter* with the traditional **Set command's Syntax**.

Syntax: (for arithmetic operations) Set /a [VariableName]=[Variable1]+[variable2]...

Instead of using '+', you can perform any other mathematical operation you want, and perform different operations on any number of variables...or constants.

Note: If you want to delete a pre-existing variable...just use following syntax: **set [variableName]=**

i.e. leave the blank space after '=' then if you try to display its value...it will give you output as a variable which is not initialized yet !!

User interactions (*Getting User-inputs*)

As Computers are still not that much intelligent that they can understand what we want from them...so, we have to code...to make a computer understand our requirements. But, you can't code the

intelligence of a human (*till now, I don't think that it is possible...*). So, we still need to get inputs from user...who is operating the computer...

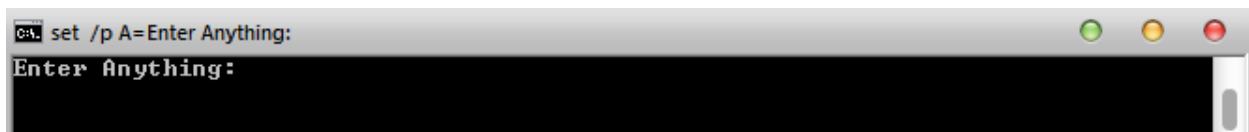
E.g. if there are 2 Users, one of them wants to see 5 natural nos. on console, while other one wants to see 10 natural nos. on console. So, if you are a wise programmer...then instead of making 2 separate programs, you will make such flexible program which can fulfill the requirement of both the users as both want to see '**Natural Numbers**'

Just make a program, which asks user about numbers of 'natural nos.' to display...and then displays them...

That's why User input is very much important in any programming language...

You can get user input in cmd by using '**Set command**'

Syntax: Set /p [variableName]=[Msg to show]



```
C:\> set /p A=Enter Anything:
Enter Anything:
```

Time for example

Let us make the same program in batch that we'd talked about above in **e.g.** section...

Q: WAP to show 'n' natural Numbers in cmd console.

Code:

@echo off

Cls

Title Natural numbers

Echo.

Set loopVar=0

Set /p n=How many natural numbers :

:loop

If %loopvar% neq %n% (

Echo %loopvar%

Set /a loopvar=%loopvar%+1

Goto loop


)

Pause

Exit

Output: same program having different output as per **user inputs...**

For user 1:

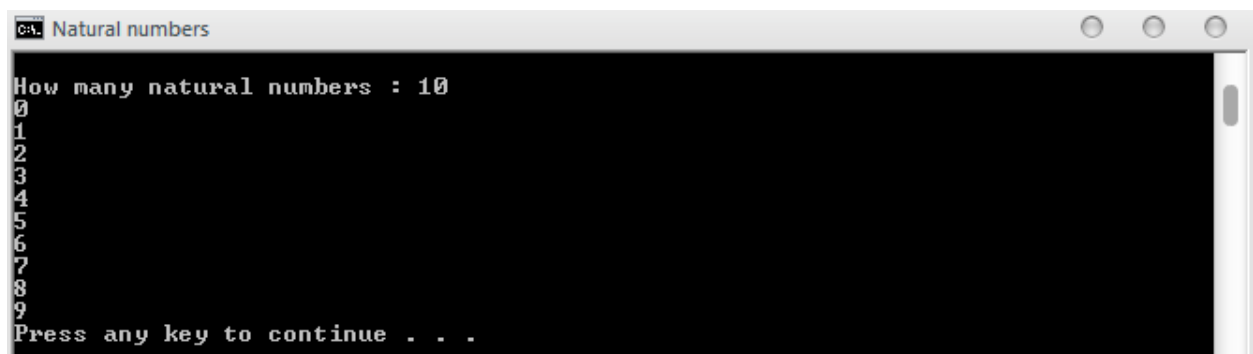


```

Natural numbers
How many natural numbers : 5
0
1
2
3
4
Press any key to continue . . .

```

For user 2:



```

Natural numbers
How many natural numbers : 10
0
1
2
3
4
5
6
7
8
9
Press any key to continue . . .

```

Explanation:

Skipping 1st 4 lines, as now you are not **newbie**. So, I'm assuming that you read the newbie part...and you know basics of batch programming.

Lines 5: Creating a variable named "loopVar" and setting its value to "0" *(as natural nos. starts from "0")*

Lines 6: Getting input from user...and saving user-input to variable named "n".

Lines 7: Creating a label...to repeat the loop till our desired result on console.

Lines 8: Checking whether to print the number or not. *(use of if will be discussed later in this part)*

Lines 9: Printing the number on console.

Lines 10: Incrementing the value of 'LoopVar' variable for next value.

Lines 11: Changing control to label :loop again...and then the steps from 8 to 11 repeats until the condition becomes false.

Lines 12: Pausing console...

Lines 13: Exiting program !!!

Some Pre-defined variables in batch (The environment variables)

Now we know to declare, assign and use values within variables in cmd... but there are others variables also which are not declared or initialized by us...but they are still there...

Try typing '**echo %username%**' in cmd...it will show your username...(in my case it's **kvc**)

Or try '**echo %computername%**' in cmd...it will show your computer-name (in my case **kvc-pc**)

These variables are Pre-defined by '**Microsoft**'...and you can get detail about how many variables are defined in cmd till any moment of time by typing '**set**' in cmd console alone...(without any parameter).

But if you type '**Set <any character>**', e.g. **set p**

Then cmd will show detail about all variables starting from letter 'p'...

These pre-defined variable may be helpful when you are trying to make a program which needs to put some files in desktop of user...but in batch programming you need to know full path of any location to copy a file there...and on every other computer the username varies...so you can handle this problem by using pre-defined variables...

TIP: To put a file in user's desktop on any computer...use

Copy /y "[your filename.itsExtension]" "%userprofile%\desktop"

Conditional statement – The If Statement

If statement is the most famous statement in any programming language...it is basically used to build the logic of any program in any programming language.

But in batch it is not an statement...like all other things in batch...it is also a command...we'll call it as '**if command**'.

Syntax: If [**your logic**] (commands to execute) ELSE (commands)

Your logic: may be any condition in your program, that you want to check.

If logic is true...then commands in 1st brackets are executed otherwise the else part will be executed.

To compare two variable or constants with **if** command use:

EQU - equal or use '==' (two equal to signs together)

NEQ - not equal

LSS - less than

LEQ - less than or equal

GTR - greater than

GEQ - greater than or equal

E.g. *if 4 lss 5 (echo True) else (echo false)*

You can also perform above operations on strings using the same syntax...no special treatment for strings...

You can also check, if the file is present in specified location in any system...by using if statement.

Syntax: *if exist "file path" (commands to execute) else (commands)*

If you've noted that...the basic syntax of **if** command remains same...as told firstly...

Here, **your logic** part is: **[exist "file path"]**

And if you want the output as opposite of **your logic**, just use **not** before writing **your logic**.

E.g. *if not exist "myfile.txt" (echo not found)*

Now you can easily understand the example in 'time for example' section above.

Finally this part ends

After reading contents of this Book, I think that you successfully understood the concept of *variables-handling* in **Batch Programming**... What we had done till now, is assigning values to variables...and using these values to make our programs more professional.

If you like the Way I've explained...do tell me because your one message can boost me up for writing more books and programs in batch for you all.

Check out Next part of this series for a deeper understanding of **Loops, file-handling, Functions and string Manipulations** in batch.

Contact: karanveerchouhan@gmail.com

Blog: batchprogrammers.blogspot.in

#kvc