

Simplified Automation for IT

A How-to Guide for Easy Data Backup

By Chad Jordan - February 8th 2008

In this guide you will learn:

- 1) The fundamentals of Windows batch scripting
- 2) How to structure an automated backup plan for simplifying client data backup
- 3) How to write and execute a fully-automated backup program using batch scripting

Introduction

Automation has been a system that many IT departments have used with great confidence, while there are some that are hesitant to use it. Through my experience with automated technologies like Autolt, and Powershell this is understandable. Even if written well, automated applications can sometimes be glitchy and freeze up. However, when it comes to batch script, I stand firm in suggesting this type of automation to everyone. If written correctly, I've learned that batch scripting is the most universally successful and easiest methods in executing automated tasks on Windows systems. While this guide is designed to provide a simplified approach to batch scripting, I realize that some or many IT technicians may not be coders, even for writing in batch script. In my opinion, coding with batch script is just a pinch above HTML coding. It's really not difficult. You'll probably spend more time trying to conceptualize and plan exactly what you want your batch program to do, rather than coding it. As long as you are comfortable with a very watered-down version of coding, and understand how and where data needs to be transferred safely for clients, then this process should be very straightforward for you. Like previous guides that I've written, I will approach this topic in a very detailed manner, explaining the process as clear as I can; and just like previous guides this program will be another real-world example of a computer program that I created and released for my current IT department at Taylor University. This program was created to simplify the data backup process for every employee across the university. Once I provided a demonstration of how this program worked, they agreed to use my script over their servers and integrated it into their imaging process. This guide will demonstrate exactly how I made my program using a simple text document in Windows to write this program. Unlike other scripting technologies, batch scripting for Windows does not require any additional application software or configuration to be used. You simply create a new blank text document in Windows, and basically creating a document with an array of Windows command prompt commands to automate in a sequence for backing up employee data over a secure network. With all of this in mind, let's jump right in.

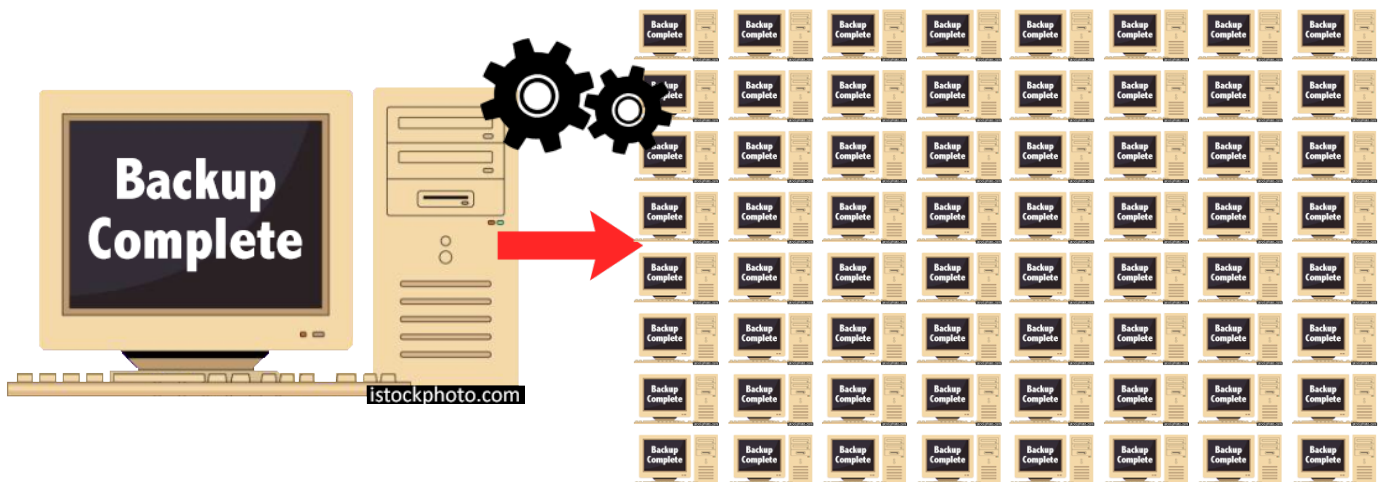
Basics of Batch Scripting

First, we have to understand what scripting/coding for batch applications are. Batch scripts are typically used for automating tedious tasks, running sequences of commands, or setting up specific configurations in a Windows environment. Batch scripting can certainly be a form of programming because we can still create data types like variables along with other conditional procedures such as if statements, for loops, while loops, etc. if we choose to. It's the manner in how we code these procedures that really separates it from more traditional programming syntax. The overall concept of scripting is far more direct (*black and white*), than a paradigm-driven process. In batch scripting, the commands are executed in sequential order, one after the other. Many developers (*myself included*) have performed these procedures over a corporate network of configured servers.

Some of the common key concepts and features of batch scripting are the following:

1. **Script File Extension:** Batch scripts typically have a file extension of **.bat** or **.cmd**
2. **Command Prompt Commands:** Batch scripts can include a multitude of commands just as you would execute single commands in the Windows Command Prompt such as **echo**, **dir**, **copy**, and many more.
3. **Variables:** Batch scripts can use variables to store and manipulate data. In batch scripting, variables are defined using the set command, and their values can be accessed using `%variable_name%`
4. **Environment Variables:** Batch scripts can access and modify environment variables to store information about the system environment and can be used for various purposes in batch scripting.
5. **Error Handling:** Batch scripts can handle errors using conditional statements and error levels. Commands may return an error level, and scripts can check this level to determine whether a command was successful.

We know that batch scripts are used to alleviate repetitive tasks in the command prompt by automating the process. When it comes to data backup commands, anyone who has used the command prompt to perform data backups knows about **copy**, **xcopy**, and **RoboCopy** to backup individual directories and or files. However, when it comes to a mass copy or backup system to alleviate repetitive tasks from computer to computer, we need a process to save on time, resources and money. Therefore, we have automation to address this endeavor.



The answer is pretty obvious right? Why physically perform the action over and over again from computer to computer? Why not have an automated system in place to execute every command for the technician? Common sense would dictate that this is the way to go, but in order to integrate this process, we have a few more areas to cover first.

Structuring A Plan

We know that the script needs to reach every client's computer in the organization, and in my particular situation, there are currently a little more than 500 employees at Taylor University that are using Windows PC's. Ergo, this was my thought process:

1. The script should be user-focused:

- Make the process easy! A maximum of three actions. Upon logging in, one action to run the script, another to confirm their choice for security reasons, and finally, one more to close out after the script has provided a "Backup complete" message at the end of the backup process.
- The user/client should have the freedom to back up the data from their profile when they feel it is needed. This is considerate for their fluctuating work schedules, but it will be strongly recommended to still back up their data on a daily basis.
- The freedom to choose also alleviates any concern from the technician's end to check in with the client on data backups. The responsibility is put back on the client, and IT technicians can see from their end each time the client has run the script should they wish to keep an eye on the process.

2. The script should be network-focused:

- Since the client's data is already backed up to their network drive on the department servers, the script will not behave any different.
- Due to the data being backed up over a network, the script will **only** account for relevant user directories, and not the entire profile per **computer name**.
- In order to run the script properly, the university's new Altiris image will need to include the script in the **startup** folder in order to run each time the client's computer is booted up, and the profile is logged into.

So, the process is, the batch file to the Altiris image, the image is placed on the database servers, which is then put on the client's computer, the server is connected to the network drive which is linked to the client's computer, once the script is run from the local machine, the script then backs the data up to the user's network drive on the database servers.



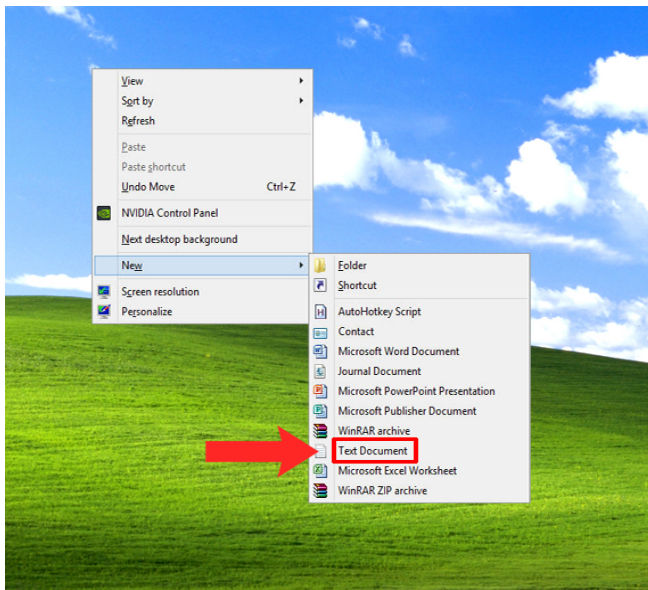
This is a very straight forward process and the data communication between the script and the servers once it's run is when the automated magic happens. Per the above illustration, once the image is sent out to all of the employee computers remotely, **only step 5** is repeated over and over behind the scenes. As mentioned earlier, this is because the script will be placed in the startup folder on the client's computer via the Altiris image. With this process understood, it's time to implement the code into the batch script.

Creating the Batch Program

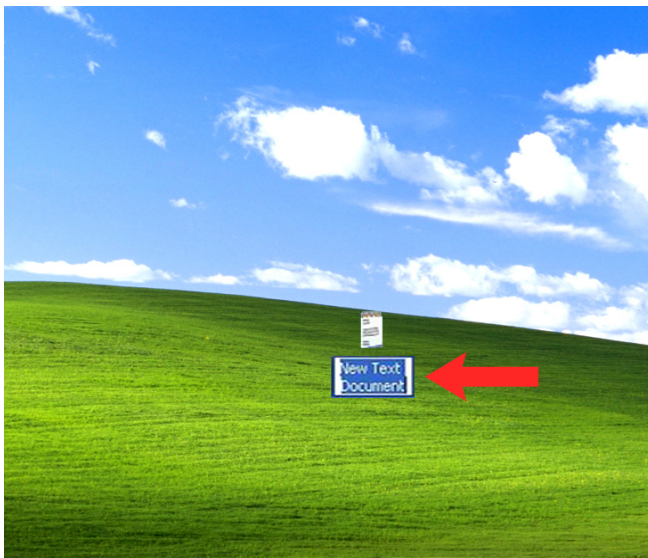
Now for some lightweight coding. Follow along and I'll get you there one step at a time!

1. **Right-click** on your desktop and create a new blank **text document**.

Note: A blank notepad document will work as well.

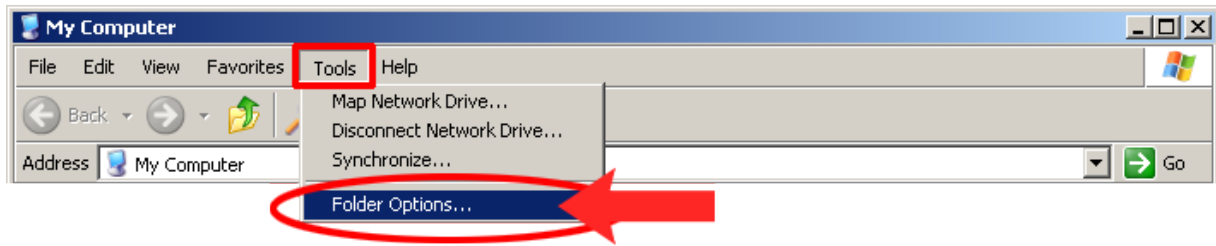


2. With the newly created text document sitting in place, your file should be displaying like this:

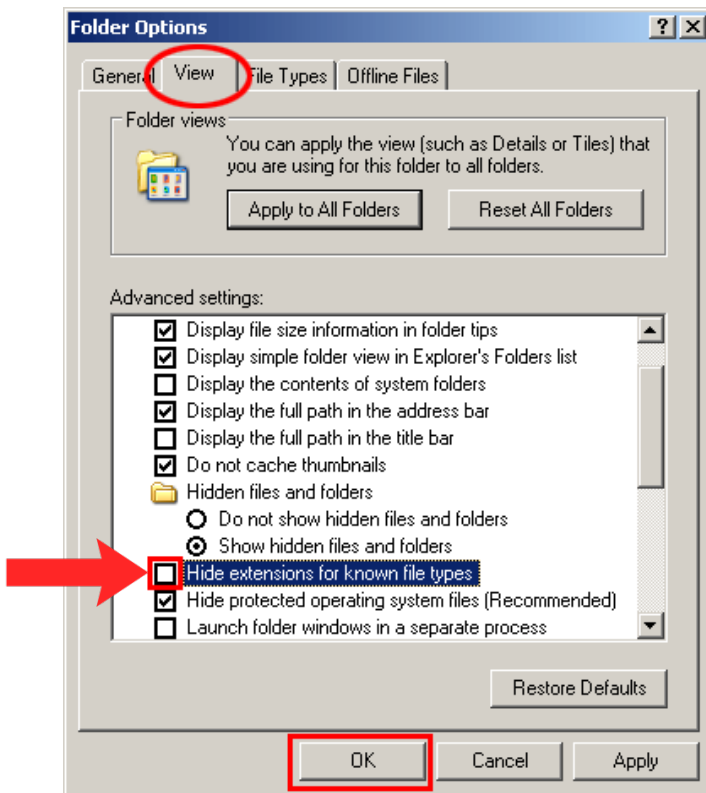


However, with no file extension displaying on my document, I will need to enable the file extension to display on the end of the filename so that I can appropriately change its file format from .txt to .bat

3. To do this, simply click on **'My Computer'** to open a new Windows Explorer window, and click the **'Tools'** button at the top, followed by **'Folder Options'**.



4. With the **Folder Options** window now open, click the **'View'** tab at the top and then scroll down and make sure the **'Hide extensions for known file types'** checkbox is deselected, and click **Ok**. This will enable file extensions displayed on every file type.



5. **Right-click** on your New Text Document, and select **rename**. Name it whatever you want; something simple like *RunBackup* and then arrow over to the right and change the file extension from **.txt** to **.bat** and when done, your new icon should change to this:



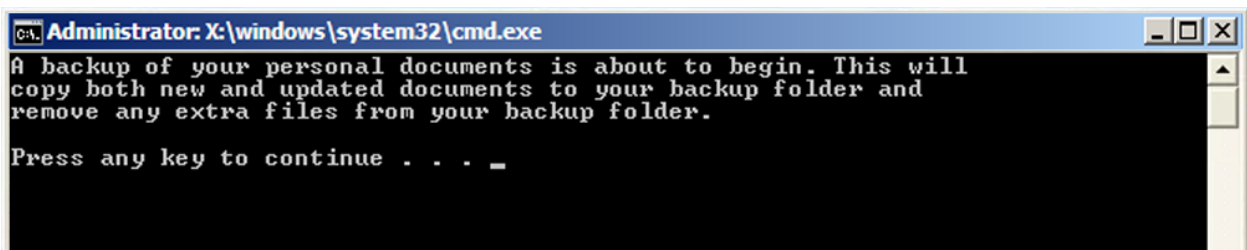
6. We're all set to write the code now! Simply **Right-click** on the batch file and select **Edit**. Once the text document is open, type the following text into the document:

```
@echo off

echo A backup of your personal documents is about to begin. This will
echo copy both new and updated documents to your backup folder and
echo remove any extra files from your backup folder.
echo.
```

Typing '**@echo off**' on the first line suppresses echoing in the specified script. In other words, we do not want every line to display (echo) the details (Backup.Batch on Administrator) of the script on every line while the script is running. We can still echo out other messages, but the unwanted details of the script will not display on each line. This is what we want, so after that enter in a space and start printing out the messages you want the program to display to the user.

In this case, I'm providing the user with the first details of the backup program. It's also important to note that the information in the dialog window will display exactly as we have it in the echo commands with the one exception of **echo.** In other words, the fourth echo command that simply has '**echo.**' with an immediate period after it, is telling the program that we want to input a space between sentences. At this point, save the file and run it. You should see the following inside of your program:



```
Administrator: X:\windows\system32\cmd.exe
A backup of your personal documents is about to begin. This will
copy both new and updated documents to your backup folder and
remove any extra files from your backup folder.

Press any key to continue . . . _
```

Per the above details, this is displaying exactly as we want to see it so far, so let's keep going. Just like before, **Right-click** the file and choose **Edit** to edit the program.

7. Let's add more instructions for the client along with a couple of new details.

```
echo The documents will be placed in your X:\BACKUP_%computername% folder.
echo All other files will be stored in your D:\Backup folder
echo You can browse to these documents at any time.
echo.
echo Please close ALL PROGRAMS before continuing with the backup.
echo.
echo.
pause
```

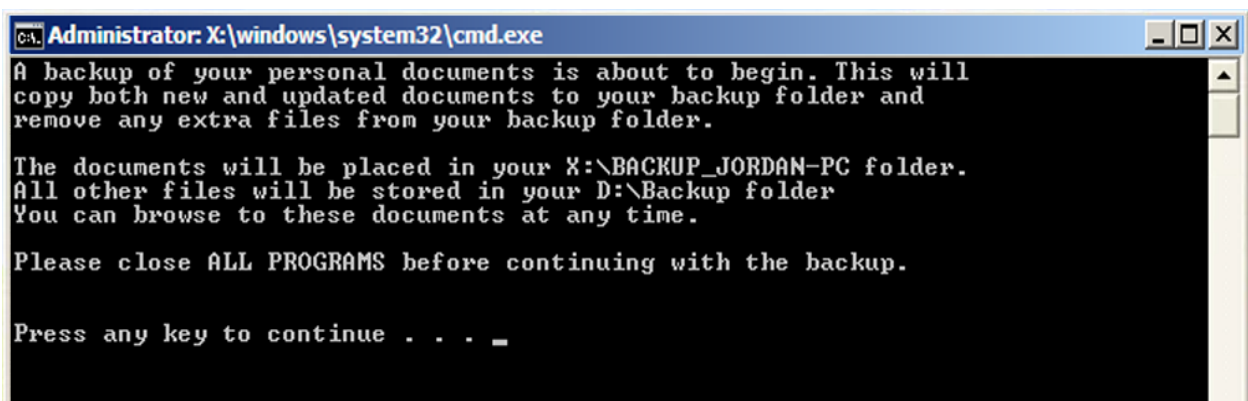
For my situation, the **X:** drive is the designated network drive for all employees at the university so I want to specify the drive letter, followed by the **BACKUP_%computername%** folder.

In Active Directory, we know that we can't have any duplicate computer names, ergo the script will read in the variable from inside the percentage symbols, and create a new destination directory at **X:\BACKUP_%computername%**. An opening and closing percentage symbol are how we specify variables in batch scripting.

We know they need to allow the computer time to run the backup without doing anything else on the machine, so we output another space, followed by a message clarifying the importance of allowing the script to run without doing anything else on the computer.

This is another reason why we put the script in the user's startup folder so at the moment of logging into their computers in the morning, the script can run their backup for them.

Then, we enter a **pause** command to suspend the batch file until the user presses a key. Let's run the program and see how it looks after adding our additional code.



```
Administrator: X:\windows\system32\cmd.exe
A backup of your personal documents is about to begin. This will
copy both new and updated documents to your backup folder and
remove any extra files from your backup folder.

The documents will be placed in your X:\BACKUP_JORDAN-PC folder.
All other files will be stored in your D:\Backup folder
You can browse to these documents at any time.

Please close ALL PROGRAMS before continuing with the backup.

Press any key to continue . . . _
```

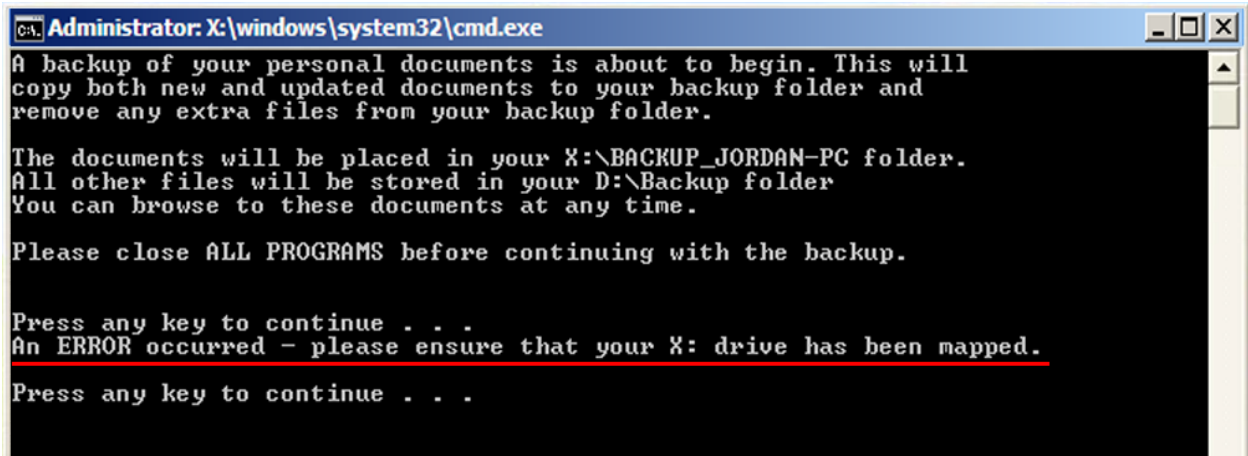
As we can see, it's recognized my computers name, so the variable, **%computername%** is properly pulling the information we want.

8. At this point, we need to put in a conditional check in the event that any drive letters are changed over time. Essentially, if the X: drive does not exist, the script will throw an error message and exit the program. This is how I implemented it into my script:

```
if exist X: goto continue
echo An error occurred – please ensure that you X: drive has been mapped.
echo.
pause
exit
```

This script is by far the easiest conditional statement I've ever written. It's essentially saying, if the X: drive exists, then go to the next section which will be marked by the **continue** command,

otherwise, give an error message, and then exit the program. The continue command is frequently used when writing conditionals and loops in batch scripting. Since I want to ensure this functionality is properly in place, I will change the drive letter to one that is not present and make sure the error is thrown. Let's see if the above script works.



```
Administrator: X:\windows\system32\cmd.exe
A backup of your personal documents is about to begin. This will
copy both new and updated documents to your backup folder and
remove any extra files from your backup folder.

The documents will be placed in your X:\BACKUP_JORDAN-PC folder.
All other files will be stored in your D:\Backup folder
You can browse to these documents at any time.

Please close ALL PROGRAMS before continuing with the backup.

Press any key to continue . . .
An ERROR occurred - please ensure that your X: drive has been mapped.
Press any key to continue . . .
```

Huzzah! The error message works! This means that I can change the drive letter back to X: and proceed with the remaining commands required for the data backup to properly function.

9. Let's consider that for security purposes we want to ensure a log is kept for the current date and time that the backups are performed for each employee on each network drive. This next section is the 'goto' continue section that I spoke of earlier.

```
:continue
echo.>>"X:\_ Backup Log File - Do Not Delete.txt"
echo Backup Version 3.2 Started For %computename%:>>"X:\_ Backup Log File - Do Not Delete.txt"
echo.|date|find"current">>"X:\_ Backup Log File - Do Not Delete.txt"
echo.|time|find"current">>"X:\_ Backup Log File - Do Not Delete.txt"
```

This code records the day and time every moment the program is run for each employee, and backs it up to their network drives on the servers for our records. The **echo. >>** command redirects the output, appends it to the file, and overwrites the data with the new data.

10. When backing up user profiles, we only need relevant directories backed up, such as:

- 1) Desktop
- 2) Documents
- 3) Downloads
- 4) Contacts
- 5) Favorites
- 6) *(Additional Media Files)*

There's a lot of other system data and installation files for applications that are needless for the data we are backing up, and therefore we only backup the directories of the data that is relevant. At this point, we use the **RoboCopy** command and pass appropriate arguments to run our backup. Example:

/MIR – Mirrors a directory tree

/B – Copies files in Backup mode

/R:1 – If a failed copy, retry once

/W:1 – Wait time between retries for 1 minute

/NJH – No job header is passed because there is no need for job setup information

/XF – Exclude files matching given names, paths, or wildcards; we do this to ignore filename extension with hyphens or periods.

If something fails, or we get an **errorlevel 16** (*usage error due to insufficient access privileges on destination directories*), we go to an error.

```
echo Copying new data to the backup server...
RoboCopy.exe "%homepath%\Documents"
    "X:\Backup_%computername%\Documents" /MIR /B /R:1 /W:1 /NJH /XF *.ost *.vmc *.vhd *.vsv
if errorlevel 16 goto error
RoboCopy.exe "%homepath%\Contacts"      "X:\Backup_%computername%\Contacts" /MIR /B /R:1 /W:1 /NJH
if errorlevel 16 goto error
RoboCopy.exe "%homepath%\Favorites"     "X:\Backup_%computername%\Favorites" /MIR /B /R:1 /W:1 /NJH
if errorlevel 16 goto error
RoboCopy.exe "%homepath%\Desktop"      "X:\Backup_%computername%\Desktop" /MIR /B /R:1 /W:1 /NJH
if errorlevel 16 goto error
```

11. Everything that is deemed secondary, gets backed up to their secondary backup drive on the user's local machine.

```
echo Copying new pictures, videos, and music to backup drive...
Robocopy.exe "%homepath%\Pictures"      "D:\Backup\Pictures" /MIR /B /R:1 /W:1 /NJH
if errorlevel 16 goto error
Robocopy.exe "%homepath%\Downloads"     "D:\Backup\Downloads" /MIR /B /R:1 /W:1 /NJH
if errorlevel 16 goto error
Robocopy.exe "%homepath%\Music"         "D:\Backup\Music" /MIR /B /R:1 /W:1 /NJH
if errorlevel 16 goto error
Robocopy.exe "%homepath%\Saved Games"   "D:\Backup\Saved Games" /MIR /B /R:1 /W:1 /NJH
if errorlevel 16 goto error
Robocopy.exe "%homepath%\Share"         "D:\Backup\Share" /MIR /B /R:1 /W:1 /NJH
if errorlevel 16 goto error
Robocopy.exe "%homepath%\Videos"       "D:\Backup\Videos" /MIR /B /R:1 /W:1 /NJH
if errorlevel 16 goto error
```

12. Once the script is done performing the backup, or failed to backup, we need to update this information to the log file for our records. Once this is done, we should have everything in place for our program to run properly. Let's test it and see how it runs.

```
echo Backup Completed: >> "X:\_ Backup Log File - Do Not Delete.txt"
echo. | date | find "current" >> "X:\_ Backup Log File - Do Not Delete.txt"
echo. | time | find "current" >> "X:\_ Backup Log File - Do Not Delete.txt"

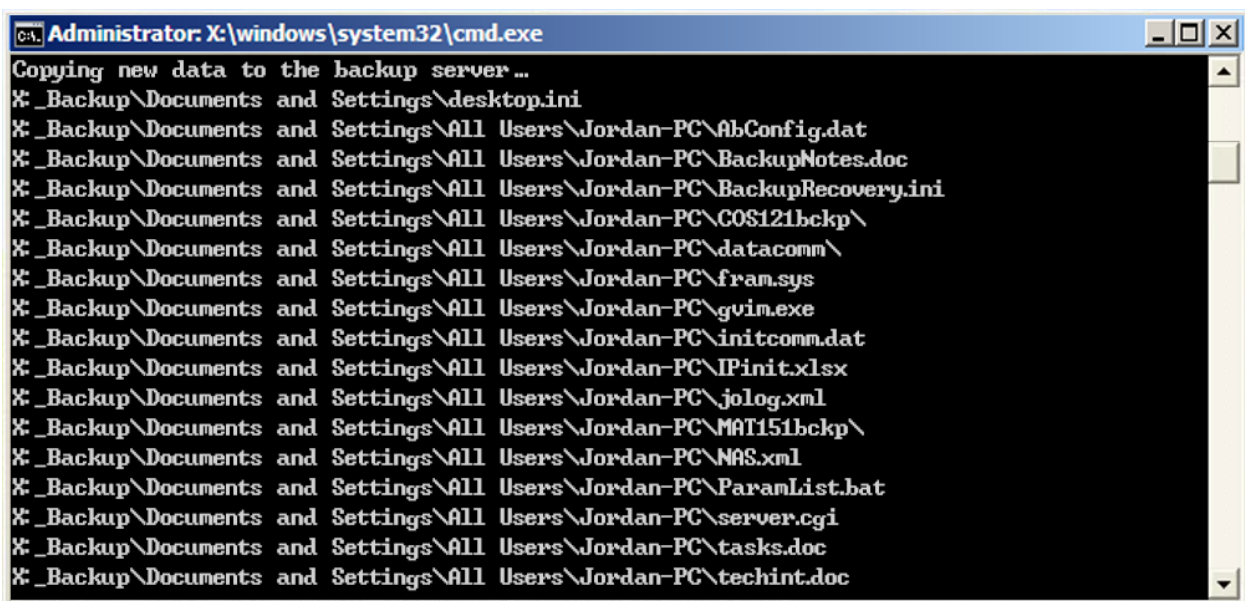
echo A backup of your personal documents has been completed.
echo.
echo Please check the contents of your X:\BACKUP_%computername% folder and your
echo D:\Backup folder to verify that all of your important documents have been
echo copied over.
echo.

pause

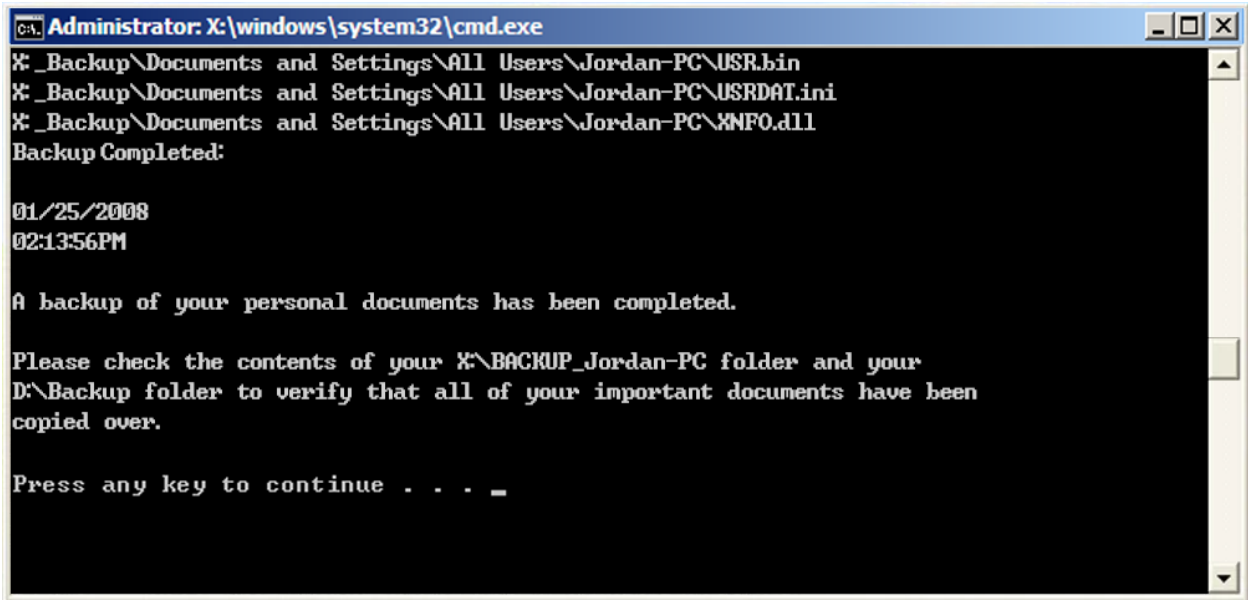
exit

:error
echo Backup Error: >> "X:\_ Backup Log File - Do Not Delete.txt"
echo. | date | find "current" >> "X:\_ Backup Log File - Do Not Delete.txt"
echo. | time | find "current" >> "X:\_ Backup Log File - Do Not Delete.txt"

echo.
echo An ERROR occurred while backing up your files. Please leave this window
echo open to see where the error occurred
echo.
pause
```



```
Administrator: X:\windows\system32\cmd.exe
Copying new data to the backup server ...
*_Backup\Documents and Settings\desktop.ini
*_Backup\Documents and Settings\All Users\Jordan-PC\AbConfig.dat
*_Backup\Documents and Settings\All Users\Jordan-PC\BackupNotes.doc
*_Backup\Documents and Settings\All Users\Jordan-PC\BackupRecovery.ini
*_Backup\Documents and Settings\All Users\Jordan-PC\COS121bckp\
*_Backup\Documents and Settings\All Users\Jordan-PC\datacomm\
*_Backup\Documents and Settings\All Users\Jordan-PC\fram.sys
*_Backup\Documents and Settings\All Users\Jordan-PC\gvim.exe
*_Backup\Documents and Settings\All Users\Jordan-PC\initcomm.dat
*_Backup\Documents and Settings\All Users\Jordan-PC\IPinit.xlsx
*_Backup\Documents and Settings\All Users\Jordan-PC\jolog.xml
*_Backup\Documents and Settings\All Users\Jordan-PC\MAT151bckp\
*_Backup\Documents and Settings\All Users\Jordan-PC\NAS.xml
*_Backup\Documents and Settings\All Users\Jordan-PC\ParamList.bat
*_Backup\Documents and Settings\All Users\Jordan-PC\server.cgi
*_Backup\Documents and Settings\All Users\Jordan-PC\tasks.doc
*_Backup\Documents and Settings\All Users\Jordan-PC\techint.doc
```



```
Administrator: X:\windows\system32\cmd.exe
X:\_Backup\Documents and Settings\All Users\Jordan-PC\USR.bin
X:\_Backup\Documents and Settings\All Users\Jordan-PC\USRDAT.ini
X:\_Backup\Documents and Settings\All Users\Jordan-PC\%NFO.dll
Backup Completed:

01/25/2008
02:13:56PM

A backup of your personal documents has been completed.

Please check the contents of your X:\BACKUP_Jordan-PC folder and your
D:\Backup folder to verify that all of your important documents have been
copied over.

Press any key to continue . . . _
```

The script works flawlessly! I've verified my backup on the servers and my local D:\Backup folder and everything including the log file is accounted for.

Conclusion

This concludes my guide for simplifying automation in IT, and how to create your own batch program for data backup. Before I pitched this whole idea to my employer, I had to make sure it worked in my own work environment at home, and when it did, I had developed a pretty great reputation in the IT department at Taylor University. My hope is that this guide is clear to help you build your own program, or at nothing else, helped you to learn some new things about automation and how it can significantly better the quality of your infrastructure.

Unless otherwise specified, all diagrams, illustrations, and code in this guide were created by me. If you have any questions about this guide or any other general inquiries, you can email me at technologicguy@gmail.com