

# Inside<sup>™</sup> Microsoft<sup>®</sup> Windows<sup>®</sup> 95

Tips and techniques for Windows 95 enthusiasts

## Using VBScript to automate the DOS-based Subst command

DOWNLOAD

[ftp.zdjournal.com/w95/jun00.zip](http://ftp.zdjournal.com/w95/jun00.zip)

If you store a lot of data on your hard drive, you probably use a very detailed folder structure in order to keep your data organized and to make it easy to find what you're looking for. For example, you might have a structure that has a main folder for your work, which contains other nested folders for each project category and months. Such a structure might look similar to the example shown in **Figure A**.

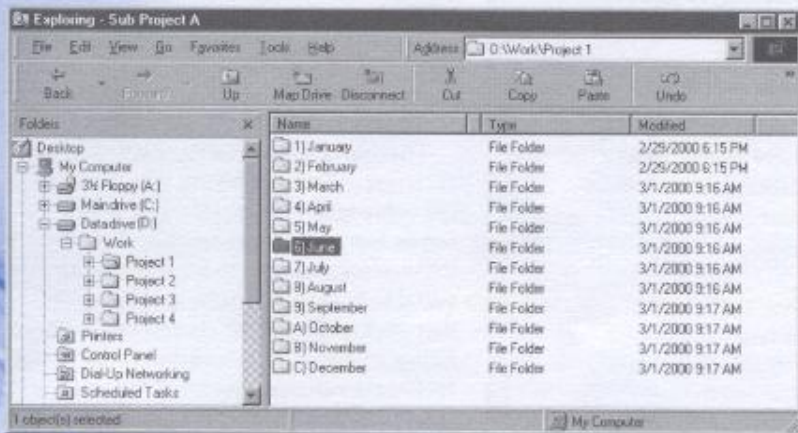
While this type of folder structure keeps your data very organized, it does require that you spend a lot of time opening and closing folders in Windows Explorer and your application's Open and Save As dialog boxes as you work. All this opening and closing of folders can be time consuming and frustrating.

Wouldn't it be nice if you could reference the path to a specific set of data folders with a single drive letter? If you're familiar with the old DOS commands, you

know that you can do so quite easily with the Subst command.

For example, if you're working with the files in the Project 1\Sub Project A\June folder, you can use the Subst command to assign that path to drive letter W. Any time you need to access files in that folder, you can just access drive W. Of course, that means that each time you want to assign a specific path to a drive letter, you have to open an MS-DOS Prompt window and manually type the commands to create the drive. This, too, can be time consuming and frustrating.

Fortunately, we discovered a way to automate the Subst command with the Windows Scripting Host and two scripts written in the VBScript language. These scripts will allow you to dynamically assign a drive letter for your nested data folders and delete that drive letter when you've finished. Once you begin using these



**Figure A:** Accessing your nested folders can mean that you spend a lot of time opening and closing folders.

### In this issue

Using VBScript to automate the DOS-based Subst command

Creating your own sound scheme

Recording sounds from audio CDs

Creating an hourly chime in Task Scheduler

*Shareware Corner:* Completely remove files from your hard drive with Eraser

### Tips & Tricks:

- Sort your files by size
- Quick scrolling
- Two ways to close a window
- Finding similar fonts
- ...and more!

### Letters to the editor:

- Using minimized windows to facilitate file transfers
- More on the modem speed report setting
- Configuring Outlook Express 5.0 to work offline



handy scripts, you'll wonder how you ever lived without them.

In this article, we'll show you how to create two scripts in VBScript that will automate the procedure of using the `Subst` command to create a drive letter for your nested folders and delete it just as easily. As we do, we'll explain each step in detail.

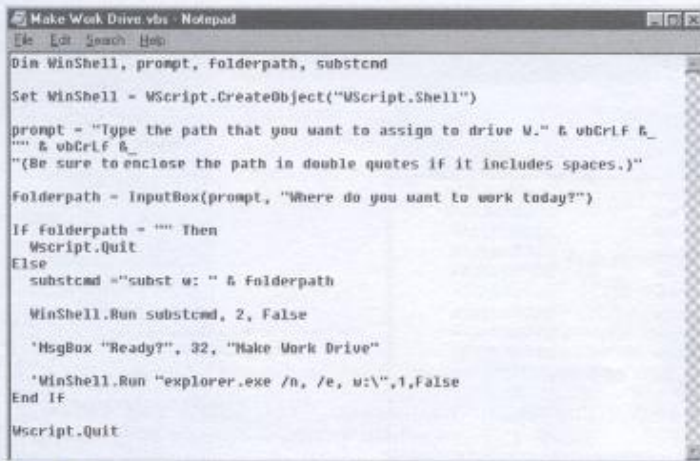
## Getting started

As we mentioned, our scripts will make use of the `Subst` command to create a drive letter for your nested folders as well as delete the drive letter. If you're not familiar with the `Subst` command, we recommend that you take a look at the sidebar "Understanding the `Subst` command." Doing so will help you understand and appreciate what the scripts are designed to automate.

Furthermore, if typing in the commands to create the scripts isn't your cup of tea, don't worry—the scripts are available for download from our ftp site at [ftp.zdjournal.com/w95/jun00.zip](http://ftp.zdjournal.com/w95/jun00.zip).

## Creating the Make Work Drive script

Creating the Make Work Drive script is easy. To begin, launch Notepad and construct the Make Work Drive script shown in **Figure B**. Be sure that you type all the commands exactly as shown. If you don't, the script won't work correctly. When you've finished, save the file as *Make Work Drive.vbs*.



```
Dim WinShell, prompt, folderpath, substcmd
Set WinShell = WScript.CreateObject("WScript.Shell")
prompt = "Type the path that you want to assign to drive U:" & vbCrLf & _
"" & vbCrLf & _
""(Be sure to enclose the path in double quotes if it includes spaces.)"
folderpath = InputBox(prompt, "Where do you want to work today?")
If folderpath = "" Then
WScript.Quit
Else
substcmd = "subst u: " & folderpath
WinShell.Run substcmd, 2, False
MsgBox "Ready?", 32, "Make Work Drive"
WinShell.Run "explorer.exe /n, /e, u:\",1,False
End If
WScript.Quit
```

**Figure B:** The Make Work Drive script automates using the DOS-based `Subst` command.

## Studying the Make Work Drive script

Now that you've created the Make Work Drive script, let's take a step-by-step look at how the script works. Doing so now will give you a basic understanding of how the VBScript commands work in the script function and will make it easier for you to customize the script later.

The first line in the script defines the four variables, or placeholders, that the script uses to store information. We've used names that represent the information that the script stores in the variables. For example, the `WinShell` variable will hold special information about the Windows operating system, the `prompt` variable will hold the prompt that will appear in the dialog box, the `folderpath` variable will hold the path to the folders you want to work with, and the `substcmd` variable will hold the complete `Subst` command.

The second line activates the `WinShell` variable and provides it with access to some of the Windows operating system objects that the Windows Scripting Host makes available to the scripting languages. In this script, we'll take advantage of the objects that launch applications and display dialog boxes.

The third line assigns the text that will be the prompt in the dialog box to the `prompt` variable. The `vbCrLf` command at the end of each line of text is equivalent to pressing [Enter] after each line of text. This makes it possible to format the contents of the dialog box.

The next command line uses the `InputBox` command to create the dialog box, which waits for input from the user. When the user types a path in the text box and clicks OK or leaves the text box blank and clicks Cancel, the result is assigned to the `folderpath` variable.

The next eight lines make up the `If...Then...Else` statement, which analyzes the value in the `folderpath` variable and performs one of two actions. The first part of the statement checks to see if the `folderpath` variable is empty. If it is, then the user either clicked Cancel or clicked OK without filling in the text box. Either way, if the `folderpath` variable is empty, the `WScript.Quit` command terminates the script.

If the `folderpath` variable contains text, the script jumps to the commands that fol-



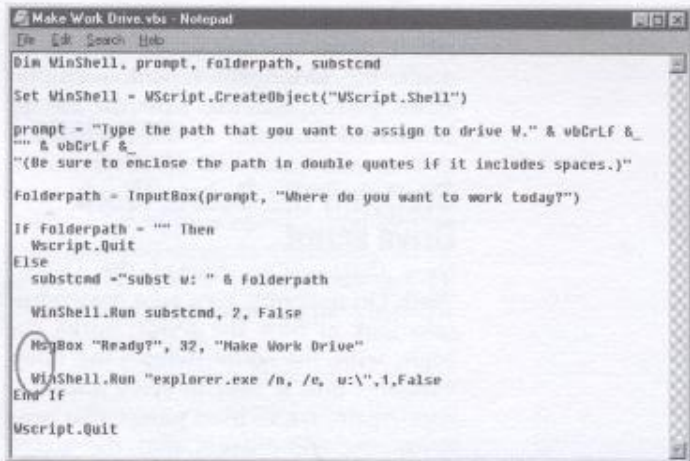
low the Else statement. There, the `substcmd` variable is assigned a text string that includes the command `subst w:` as well as the path that's stored as a text string in the `folderpath` variable.

The next line then uses the `WinShell.Run` command to run the DOS-based `Subst` command, which assigns the path to the drive letter `W`. As soon as the `Subst` command creates the drive, the script opens a My Computer style window and displays the contents of drive `W`.

The next two lines contain *optional* commands that allow the script to display the contents of drive `W` in a Windows Explorer style window. These lines are disabled by virtue of the apostrophe (') at the beginning of each line. If you're satisfied with a My Computer style window, you can leave these commands disabled.

However, if you'd prefer to see the contents of drive `W` in a Windows Explorer style window, all you have to do is delete the apostrophe marks, as shown in **Figure C**. When you do, the `MsgBox` command will immediately display a prompt and prevent the script from opening a My Computer style window. When you click OK, the following `WinShell.Run` command displays the contents of drive `W` in a Windows Explorer style window.

Regardless of which window style you decide to go with, the `End If` command



**Figure C:** If you'd prefer to use a Windows Explorer style window, simply delete the apostrophe marks in front of the two disabled lines.

closes the `If...Then...Else` statement. Then, the `Wscript.Quit` command terminates the script.

## Creating the Delete Work Drive script

Once you've run the Make Work Drive script, you'll need a way to delete the work drive should you want to assign the drive letter to another path. To do so, you'll use the Delete Work Drive script.

To create the Delete Work Drive script, launch Notepad and construct the script

## Understanding the Subst command

Since the `Subst` command is the workhorse for our scripts, let's take a few moments to learn how it works. Basically, the `Subst` command, which is a DOS command, allows you to assign a drive letter to a specific path. If you want to assign the letter `W` to the `Work\Project 1\6) June` folder, you'd open a MS-DOS Prompt window and type the command

```
Subst W: "D:\Work\Project 1\6) June"
```

(In this case, we have to enclose the path in quotation marks because it contains spaces.)

You could then access the `Work\Project 1\6) June` folder from Windows Explorer, My Computer, or any File Open or Save As dialog box, simply by accessing drive `W`. No more clicking multiple folder icons to access the June folder.

Keep in mind that the new drive letter will be available until you delete it or reboot your system.

If you want to assign the drive letter to another path, you first have to delete the existing drive. To do so, open an MS-DOS Prompt window and use the command

```
Subst W: /d
```

Another thing to keep in mind here is that we chose to use `W` for the drive letter as it makes it easy to remember that the drive is assigned to the `Work` folder. However, you can use any letter that you want, as long as it's not already being used on the system.

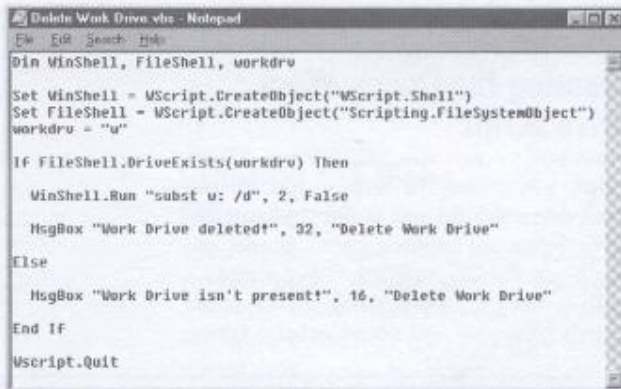
Now that you understand how the `Subst` command works, you'll appreciate the automation that the Make Work Drive and Delete Work Drive scripts provide. Once you begin using them, you'll wonder how you lived without them.



shown in **Figure D**. Be sure that you type all the commands exactly as shown. If you don't, the script won't work correctly. When you've finished, save the file as *Delete Work Drive.vbs*.

## Studying the Delete Work Drive script

Now that you've created the Delete Work Drive script, let's take a step-by-step look at how the script works. To begin with, the script defines the three variables that it uses to store information. Again, we've used names that represent the information that the script stores in the variables. Just like in the Make Work Drive script, the `WinShell` variable will hold *special* information about the Windows operating system.



```
Dim WinShell, FileShell, workdrv

Set WinShell = WScript.CreateObject("WScript.Shell")
Set FileShell = WScript.CreateObject("Scripting.FileSystemObject")
workdrv = "w"

If FileShell.DriveExists(workdrv) Then

    WinShell.Run "subst u: /d", 2, False

    MsgBox "Work Drive deleted!", 32, "Delete Work Drive"

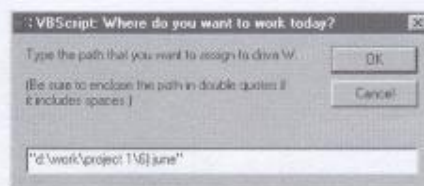
Else

    MsgBox "Work Drive isn't present!", 16, "Delete Work Drive"

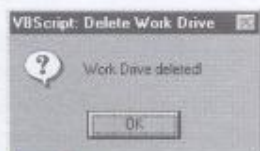
End If

WScript.Quit
```

**Figure D:** This script will delete the work drive.



**Figure E:** You'll type the path to the folder that you want to work with in the text box.



**Figure F:** When you double-click on the Delete Work Drive shortcut, the script informs you that the operation was successful.



**Figure G:** If the work drive isn't present when you double-click on the Delete Work Drive shortcut, you'll see this dialog box.

The `FileShell` variable will hold *special* information about file management and the `workdrv` will hold the drive letter assigned to the work drive.

The second line activates the `WinShell` variable and provides it with access to some of the Windows operating system objects just like it did in the Make Work Drive script. It will allow access to the objects that launch applications and display dialog boxes.

The third line activates the `FileShell` variable and provides it with access to some of the file management objects the Windows Scripting Host makes available to the scripting languages. In the Delete Work Drive script, we'll take advantage of the object that allows us to examine drive letters.

The fourth line simply assigns the `workdrv` variable the letter assigned to the work drive. In this case, the work drive is assigned to the letter *w*.

The next six lines make up an `If...Then...Else` statement, which determines if the drive letter *W* is present and performs one of two actions. If drive *W* is present, the first part of the statement uses the `WinShell.Run` command to launch the `subst /d` command to delete the drive letter. Then, the `MsgBox` command displays a message letting the user know that the drive has been deleted.

If drive *W* isn't present, it either hasn't been created yet or it's already been deleted. In either case, the following `MsgBox` command displays a message letting the user know that the drive isn't present.

The `End If` command then closes the `If...Then...Else` statement. Finally, the `Wscript.Quit` command terminates the script.

## Using the scripts

Using the scripts is easy. To make using the scripts as convenient as possible, you can create shortcuts to the scripts and then put the shortcuts on the desktop, the Start menu, the Quick Launch toolbar, or on a custom toolbar—wherever you think they would be most convenient for you.

Now, when you double-click on the Work Drive script shortcut, you'll see the Where Do You Want To Work Today? dialog box. You'll then type the path to the folder that you want to work with in the text box, as shown in **Figure E**, and click OK. Keep in mind that you must



enclose the path in double quotes if it includes spaces.

If you decide that you don't want to create drive W at this point, you can just click Cancel. When you do, the Where Do You Want To Work Today? dialog box disappears.

If after you create your work drive, you want to delete it, just double-click on the Delete Work Drive shortcut. When you do, you'll see the dialog box shown in **Figure F**. If you forget that you previously delet-

ed the work drive and double-click on the Delete Work Drive shortcut, you'll see the dialog box shown in **Figure G**.

## Conclusion

If you use a detailed folder structure to keep your data organized, you probably spend a lot of time opening and closing folders as you work. In this article, we've shown you how to simplify this task by using VBScript to automate the use of the `Subst` command. ■

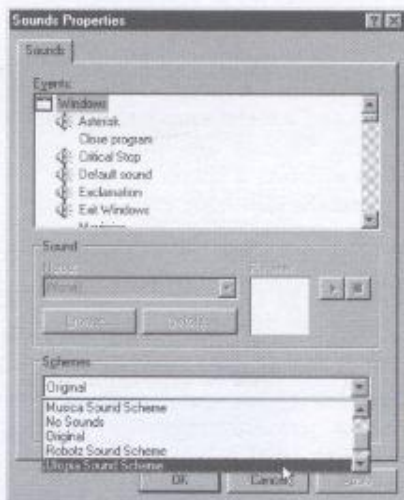
# Creating your own sound scheme

As you know, Windows 95 uses sounds to alert you to all kinds of events that occur as you're using your system. For example, you'll hear sounds when you start the system, select items from the Start menu, minimize or maximize windows, and much more. These sounds are designed to help you use the operating system and make using Windows fun.

If you enjoy having sounds alert you to events, you know that Windows 95 comes with several sound schemes from which to choose. To select from the available sound schemes, open Control Panel and double-click on the Sounds icon. When you do, you'll see the Sounds Properties dialog box. Click the Schemes dropdown list, as shown in **Figure A**, and select the sound scheme that you want to use. To apply the sound scheme, you either click the Apply button or click OK.

If you get tired of the sound schemes that come with Windows 95, you can find thousands of sounds on the Internet and use them to create your sound schemes. However, chances are that you've wondered if you could create your own unique sound schemes. Fortunately, the answer is yes and the procedure is a lot of fun! You could make it a family project on a rainy day.

Windows 95 provides you with Sound Recorder, which comes with all the tools



**Figure A:** Windows 95 comes with several sound schemes from which you can choose.

you need to record your own sounds and save them as WAV files. Sound recorder also provides you with the ability to edit, enhance, and even add special effects to your sounds. You can then use the Sounds tool in Control Panel to assemble a sound scheme. With a little ingenuity and some special techniques, you'll find yourself creating wonderful sound schemes.

In this article, we'll show you how to create unique sounds by using common items that you can find in your home. As we do, we'll show you how to use Sound Recorder to record, edit, manipulate, and save the sounds as WAV files. Then, we'll show you how to use the Sounds tool in