Windows Command Help Index

Introduction

Windows Command is a command-line interface designed for Microsoft Windows. Just as Windows is not a replacement for all DOS-based programs, Windows Command is not meant to be a replacement for COMMAND.COM. It is a method of interfacing with Windows in a familiar command-line environment. It can be used to minimize the use of DOS-based command lines while in Windows, and can also be used as a replacement for shells like Program Manager or File Manager.

Windows Command is a blend of a command line and a graphical program. It features user-customizable pull-down menus with submenus containing: all the Windows Command commands (which can be "pulled down" onto the command line), all programs shipped with Microsoft Windows, (a single menu selection will execute any of the programs), and access to the Windows Command help facility. In addition, the entire menu bar can be expanded to include menus of your own design. Windows Command also features a pop-up command history dialog that allows point-and-shoot access to previously executed commands, and a File Browser that allows you to bring a complete filename to the command line by simply selecting the file in a file dialog box.

Windows Command is a powerful command-line interface. It has many features which make using the command line easier, such as multiple commands per line, command recall, filename completion, and command aliases. It also has powerful command processing functions, such as redirection of standard output, parameter substitution, user-definable-and-accessible variables and wildcard support.

Windows Command has been designed to allow keyboard access to all features of its own, and also many features of Windows, without the use of a mouse. It allows executing, activating, moving, resizing, minimizing, maximizing, and closing Windows programs from the command line. You can also find and display window and class information for any window on the desktop (including hidden and child windows), as well as a number of other Windows functions, such as arranging icons on the desktop, loading a text file or even the entire screen buffer to the clipboard, and managing file associations, all using simple commands, and without touching a mouse.

This help file assumes that you have basic knowledge of DOS and of Windows. If you are new to this system, you may want to keep your DOS and Windows manual handy for reference.

Windows Command is a user-supported program. If you have any suggestions, comments or problems concerning Windows Command, I invite and request you to write to me at the address below:

Michael B. Tierney
205 Inglewood Dr.
Pittsburgh, PA 15228
or send me electronic mail on Compuserve:
70604,1512
from Internet/Bitnet: 70604.1512@compuserve.com

Thank you for using Windows Command. I hope you find it to be a useful tool, and a productive new way to use the Microsoft Windows environment.

Using the Menu

The Windows Command menu allows access to almost every command and feature available in Windows Command. In addition to the powerful features found in the default menu, you can define portions of the menu to execute anything that can be typed at the command line.

The default menu has five main submenus: File, Edit, Commands, Execute, and Help. Each of these allows access to powerful abilities within the Windows Command environment.

File

This submenu contains **Browse**, which allows access to the Windows Command File Browser, an **Exit Windows** selection, to exit the Microsoft Windows environment, and an **Exit** selection, to quit Windows Command.

Edit

This submenu has a **Copy Screen Buffer** selection, that allows you to copy the entire contents of the screen buffer (which includes all the text you can see by using the scroll bars on the Windows Command window) to the clipboard.

Commands

This menu selection contains categories of listings of all the available commands. The command listing is separated into categories: **Batch** file commands, **File** system commands, **DOS** function commands, **Windows** environment commands, and **Windows Command** environment commands.

Selecting one of the commands from the menu will bring that command down to the command line. Thus, selecting "Dir" from the "File" category will make "Dir" appear on the command line. At that point, you can press F1 to bring up the help screen for that command, or hit <ENTER>, and the command will execute.

Execute

This menu selection contains a listing of all the Windows programs shipped with Microsoft Windows, and allows you to execute any given program by selecting it from the list.

This submenu will be removed from the Windows Command menu bar when you load a custom menu with the **Menu** command.

Help

This menu selection allows access to the Windows Command help system. It includes an **Contents** selection, which will bring up the Windows Command Help Index, a **Windows Command Basics** selection which will bring up the Introduction to Windows Command, a **Using the Keyboard** selection, which will bring up the available command keys, **Using Windows Command Help**, which will show how to use Windows Command Help, and About Windows Command..., which shows author information for Windows Command.

Customizing the Menu

The menu can be customized by using the Menu command to load a Windows Command menu template file.

When you execute a **Menu** command, the **Execute** submenu is removed from the Windows Command menu bar, and the menu defined in the menu file begins in its place. The menu file syntax is simple, but if a mistake is found, an error is returned, and the menu is restored to the default.

Note:

The Windows Command menu interfaces very closely with the command line. Because of this, all the menu selections except for Help disable during command execution. To demonstrate this for yourself, execute a Pause command, and try to access the menu.

Batch Files

A batch file is a text file which contains a listing of commands to be executed in order. You can create such a file using Notepad, which comes with Windows. Commands are entered one-per-line in the file, with a maximum command length of 255 characters. "Windows Command Batch" files must be given the file extension ".WCB".

When you first start Windows Command, it searches for and executes the file AUTOEXEC.WCB, if it is found in any directory on your DOS path.

Windows Command has several commands available specifically for batch execution. The **If** command allows conditional execution of commands, and **Goto** will branch execution to a *label*.

A batch file label consists of a colon ":", followed by the label name. For example, **:ThisLabel** is a label named ThisLabel, and the command **Goto ThisLabel** will subsequently start batch file execution at the label ThisLabel.

A sample batch file follows:

```
@echo off
echo This is a test batch file.
goto MakeTheTestFile
:EraseTheTestFile
echo Erasing the test file...
erase testfile.$$$
if not exist testfile.$$$ goto EndTheTestBatch
:MakeTheTestFile
echo Making the test file...
dir > testfile.$$$
goto EraseTheTestFile
:EndTheTestBatch
echo Ending the test batch file...
exit
```

Execution begins with **@echo off**, which will suppress the batch file commands from appearing on the screen. Next, "**This is a test batch file...**" will be echoed to the screen, then execution will go to the label **MakeTheTestFile**. "**Making the test file...**" will echo to the screen, and then the output from a **dir** command will be routed into the file **TESTFILE.$$$**. Execution will branch to **EraseTheTestFile**, "**Erasing the test file...**" will echo to the screen, then **TESTFILE.$$$** will be erased. At that point, the file **TESTFILE.$$$** will cease to exist, so that when **if not exist testfile.$$$ then goto EndTheBatchFile** executes, the batch file will branch to EndTheBatchFile

You can also use parameter substitution and Windows Command variables in batch files to send command line arguments or variables to commands. This allows you greater flexibility in the execution of batch files.

Remember that when using a batch file to set aliases with parameter substitution or variables in them that you must use double percent signs (%%) in order to get the single percent signs into your alias. This is because when the command executes from the batch file, the command is searched for percent signs that indicate a parameter substitution or variable.

Important Note:

Do not write batch files that either call themselves infinitely or use an infinite GOTO loop. This can make the batch file execution dominate your system resources and crash or halt your system. If you have problems such as this with batch files, the cause is most likely an infinite loop.

Wildcards

Wildcards provide a way to run a command on numerous files without having to type in each file name. Commands which support wildcards are: Attrib, Copy, Del, Dir, Erase, Except, Findfile, Move, Print, Rd, Ren, Rename, Rmdir, and Type, and the Filename Completion feature.

The wildcard characters are * and ?. For example, "*.DOC", references all files in the given directory that have the extension ".DOC". "??.DOC" references all files in the given directory that have the extension ".DOC", and a filename that is two characters or less long. "MYFIL?.*" references any file whose filename begins with "MYFIL", and is no more than 6 characters long, and has any file extension. "*" is any file with any filename and no extension, and "*.*" is any file with any filename and extension.

Note: "*ANY.*" will be read as "*.*", since any characters following an asterisk in either the filename or extension are ignored.

Processing the Command Line

If the command is running from an alias or batch file, the command line is first checked for parameter or variable identifiers, which are any strings inside opening and closing percent signs (%). If any are found, they are substituted with their appropriate value. If a percent sign is entered on the command line without a second percent sign to match, an error will be generated. To use a single percent sign in an alias or batch file command, you must use double percent signs (%%), and they will be translated into one percent sign.

The command line is checked for the multiple command character, the caret (^). If you need to use a caret for any other reason, put the whole expression inside back quotes. For example, to delete the file THIS^FIL.DOC, you will need to enter ERASE "THIS^FIL.DOC". Otherwise, the command will be split up and executed as ERASE THIS, and then FIL.DOC.

Next, the command line is checked for redirection, and the redirection is performed. The standard output redirection symbols are ">" to begin a new file, or ">>" to append the file.

The command is checked for a leading at sign (@), which means that the command will not be echoed to the screen, even if echo is on.

Finally, the command line is split into arguments, and the command is then executed.

Command Execution

Windows Command tries several methods of finding the executable program when processing the command line:

File Extension Completion:

If a file extension is not given when you type the program name, Windows Command will try to find the first executable on your DOS path or given path with the matching extensions .WCB, .BAT, .PIF, .EXE, and .COM, searching for files with extensions in that order. This means that if you type XYZ, and both XYZ.WCB and XYZ.COM exist, XYZ.WCB would execute. Likewise, if only XYZ.EXE and XYZ.COM existed, XYZ.EXE would execute, etc.

Command Extension Matching:

If a file extension is given when you type the program name, the extension is first checked to make sure it is an executable (with extensions .WCB, .BAT, .PIF, .EXE, .COM), and if so, the file is executed. Otherwise, the extension is checked for a file association. If an association is found, the associated program is started with the given file as the startup file. If no association is found for a non-executable file, the filename is returned as an Unknown Command.

When the command execution is complete, Windows Command will display the name and TaskID of the program executed, if it is available. A Windows Command variable LASTEXEC is then set with the Task ID of the program that was run last.

Redirection

Windows Command supports the redirection of standard output. This means that any text from any command that appears in the Windows Command window can be routed into a file. The redirection symbols are ">", which erases the file if it exists and then puts the new text into it, and ">>", which appends the new text to the file, whether it exists or not. The redirection symbol goes at the end of the command, and is followed by the file name that you want the output to go to.

For example, **DIR > DIRFILE.TXT** will create a file called DIRFILE.TXT that contains the output of the Dir command.

Please Note: Output from the Type command can not be redirected, since the effect would be the same as a Copy command.

Using Aliases

Aliases are commands defined with the Alias command, and are comprised of other commands. Once set, aliases can be run just like any other internal command. Aliases take precedence over Windows Command internal commands, so that you can use aliases to redefine the default actions of Windows Command commands. For example, defining an alias as

```
ALIAS DIR DIR /W %*%
```

will cause Dir to default to a wide display, as opposed to a single-column display. The %*% is a parameter substitution that brings all of the arguments after the command (the first argument) to the alias.

You can also use parameter substitution and Windows Command variables in aliases to send command line arguments or variables to commands. This allows you greater flexibility in the execution of aliases.

Note:

When comparing two strings with the IF command, it is required that the two parameters to be compared be inside double quotes. This means that if you have an alias that uses the **IF** command, you must use double double quotes to get the quotes into your alias. Otherwise, the quotes will be removed when you create the alias at the command line, and **IF** will not work properly.

This should also be kept in mind for other instances where quotes are necessary for commands inside an executing alias.

Important Note:

Since aliases can call one another, be careful not to create an alias that calls itself infinitely. This can cause the alias execution to dominate your system resources and crash or halt your system. If you have problems such as these with aliases, the cause is most likely an infinite loop.

Using Browser

The Windows Command File Browser allows you to bring a file's complete pathname to the command line by selecting the file from a file dialog box. To access the browser, select File-Browse from the menu or press F2.

To select a file in the Browser, first select the drive and/or directory from the "Directories:" list, then select the filename from the "Files:" list, and press "OK" or <ENTER>. The selected file will then echo to the command line.

You can also search for more specific file names by typing portions of the filename into the "Filename:" box, and then pressing <ENTER>.

If you press ESC or "Cancel", the File Browser will close without echoing anything to the command line.

Windows Command Variables

Windows Command allows you to set an unlimited number of named variables to any value. Variables are set using the Set command. Once set, they can be accessed only in aliases or batch files, but the entire variable table can be viewed using the **Set** command.

Variable names may be up to 30 characters in length, must start with a letter ('a' through 'z'), and contain only letters and numbers ('0' through '9', and 'a' through 'z').

Variables are referenced by enclosing the variable name inside percent signs (%). Some examples of variables follow:

Set an alias as follows:

```
ALIAS TEST ECHO %THISVAR%
```

now set the variable THISVAR to a value:

```
SET THISVAR `THIS IS A VALUE`
```

now execute the alias:

```
TEST
```

and the output will be

```
THIS IS A VALUE
```

The following line would produce the same results inside a batch file:

```
ECHO %THISVAR%
```

If a variable is not found, the variable reference will be removed, and nothing will be substituted in its place. Be careful not to allow any extraneous spaces or other characters between the percent signs, or else the variable will be interpreted incorrectly.

Variables work in much the same way as parameter substitution.

Multiple Commands

Windows Command allows more than one command to be executed on a line. The separational character, the caret (^), marks the beginning of a new command on a line. For example, **DIR ^ ECHO HI THERE** will execute as if you entered **DIR**, and then **ECHO HI THERE**.

When you use multiple commands, if a command fails, the remainder of the line does not execute. For example, if you entered **DIR /X ^ ECHO HI THERE**, DIR /X would exit with an error (Invalid Parameter), and ECHO HI THERE would never execute. This allows you to structure a series of commands that require successful completion to continue.

If you need to use a caret for reasons other than as a multiple command identifier, you must enclose the argument with the caret(s) in quote characters.

Parameter Substitution

Windows Command allows you to access command-line parameters within aliases and batch files. Parameters are arguments on the command line that are separated by "white space" characters, and are all numbered, starting from zero. The first argument (parameter zero) on any command line is always the command, so parameter 1 is always the the first argument passed to the command, followed by parameter 2, and so on.

Parameters are referenced by enclosing the parameter number inside percent signs (%). Some examples of parameter substitution follow:

Set an alias as follows:

```
ALIAS TEST ECHO %2% %1%
```

now execute the alias:

```
TEST THERE HI
```

and the output will be

```
HI THERE
```

The following line would produce the same results inside a batch file:

```
ECHO %2% %1%
```

Multiple Parameter Referencing

There are occasions when you are not sure how many parameters will be on the command line, but you want to use all or most of them. Windows Command allows you to reference more than one parameter using the wildcard character, the asterisk (*). To return all the parameters from parameter number 1 forward, use %*%. To return all the parameters including the command (parameter 0) forward, use %0*%. Some parameter substitution examples follow:

Set an alias as follows:

```
ALIAS TEST ECHO %*%
```

now execute the alias:

```
TEST HI THERE EVERYONE
```

and the output will be

```
HI THERE EVERYONE
```

The following line would produce the same results inside a batch file:

```
ECHO %*%
```

or set an alias as follows:

```
ALIAS TEST ECHO %2*%
```

now execute the alias:

```
TEST HI THERE EVERYONE
```

and the output will be

```
THERE EVERYONE
```

The following line would produce the same results inside a batch file:

```
ECHO %2*%
```

If a numbered parameter does not exist, the parameter reference will be removed, and nothing will be substituted in its place. Be careful not to allow any extraneous spaces or other characters between the percent signs, or else the parameter will be interpreted incorrectly.

Parameter substitution works in much the same way as Windows Command variables.

Command Options

Many of the commands have options which modify the way they execute. These are all listed in the individual commands' help screens. The help screens use the convention of the foreslash (/) to show command options, however, all commands will accept either a foreslash or a dash (-) as a command option delimiter. For example, **DIR /W** and **DIR -W** will execute exactly the same way.

Executing Windows Command

You can specify a command to execute on startup by typing the command as a parameter when you run Windows Command. For example, typing WCOMMAND DIR /W ^ CLS in the Run dialog of Program Manager or File Manager (or whatever you use to start Windows Command) will cause Windows Command to start, and execute DIR /W, and then CLS.

Normally, when you start Windows Command, it searches for and executes the AUTOEXEC.WCB batch file if it exists. However, if you specify a command on the command line (which could be another batch file), AUTOEXEC.WCB will not be executed on startup, and only the given command will be run.

Filename Completion

With filename completion, you can search for a filename and insert it into a command by scrolling through file names. First, you enter none or as much of the filename as you want to search for, including wildcards, then press **F9** or **TAB**. The filename will be inserted automatically into the command line, and you can continue scrolling through different filenames by pressing **F9**. If you go too far, you can scroll backwards through the files by pressing **F8**.

Windows Command searches for matching filenames by assuming a wildcard (asterisk) at the end of both the filename and file extension name. Therefore, typing **F.D** then **F9** will begin scrolling through all files matching the pattern **F*.D***.

For example, if you want to open Notepad to edit the file SYSTEM.INI in your Windows directory:

- if you are in another directory, type **NOTEPAD \WINDOWS\S.INI**, make sure the cursor is on the partial filename, or on the space to the right of the last character, and hit **F9**. Continue pressing **F9** (or press **F8** if you accidentally go too far) until you reach **SYSTEM.INI**. Then press **ENTER** to process the command.
- if you are in your Windows directory, type **NOTEPAD S** or just **NOTEPAD** (with a space after the last letter), make sure the cursor is on the partial filename, or on the space to the right of the last character, (or to the right of the space after "NOTEPAD") and hit **F9**. Continue pressing **F9** (or press **F8** if you accidentally go too far) until you reach **SYSTEM.INI**. Then press **ENTER** to process the command.

Command Recall

Command Recall allows you to recall previously entered commands to edit or execute them again. Pressing the **UP** arrow key will recall the last command entered. Pressing **UP** repeated times will continue back to the first command, and then wrap around to the last command again. Once **UP** has been pressed, pressing **DOWN** will recall the commands in reverse order. You can view the entire command history list by entering the History command.

Command Editing Keys

The following are special keys used by Windows Command:

Left	Move cursor one character to the left
Right	Move cursor one character to the right
Home	Move cursor to beginning of command line
End	Move cursor to end of command line
PageUp	Scroll display up one page
PageDown	Scroll display down one page
Enter	Enters the command to process
Insert	Toggles insert mode
Delete	Deletes character hilighted by cursor
Backspace	Deletes character immediately before cursor
Escape	Erases entire command line
F3	Command Recall - Forward
Up	Command Recall - Forward
Down	Command Recall - Backward
F9	Filename Completion - Forward
Tab	Filename Completion - Forward
F8	Filename Completion - Backward
Shift-Tab	Filename Completion - Backward
ALT	Accesses the Menu
F1	Help - Context-sensitive help if command line is not empty
F2	Opens the Windows Command File Browser
Ctrl-Ins	Copies the contents of the screen buffer to the clipboard
Ctrl-Break	Interrupts command activity
Alt-F4	Exit Windows Command

Using Help

There are a number of ways to access the Windows Command Help Facility: pressing F1, selecting Help options from the menu, or issuing the Help command at the command line.

When you press F1, Windows Command looks at the current command line. If the line is empty, the Help Index is displayed. If there is a command on the command line, the first word is read, and the help screen for the appropriate command will be displayed.

The Help submenu gives instant access to the Help Index, the IntroductionI screen, the Using the Keyboard screen, and this screen, Using Help.

The Help command issued at the command line can display either the Help Index or context-sensitive help.

Batch Commands

These commands are used in Windows Command Batch files:

Goto	Branch to a Label
If	Conditional command Execution
Pause	Pause Until User Presses a Key
Rem	Remark or Null Command

File Commands

These commands control files and directory operations:

Attrib	Get/Set File Attributes
Cd	Change Current Directory
Chdir	Change Current Directory
Copy	Copy a File
Del	Erase a File
Dir	Display a Directory of Files
Erase	Erase a File
Except	Execute on All Except Specified Files
Findfile	Find Files on an Entire Drive
Md	Make a New Directory
Mkdir	Make a New Directory
Move	Move a File
Pwd	Print Working (current) Directory
Rd	Remove a Directory
Ren	Rename a File
Rename	Rename a File
Rmdir	Remove a Directory
Ser	Get a disk serial number
Type	Display a file on the screen
Vol	Get a Disk Volume Label

DOS Commands

These Commands give information on, or work through the Operating System:

Date	Get/Set the System Date
Mem	Display Free Memory and System Statistics
Path	Display DOS Path
Print	Print a File
Set	Get/Set DOS/Windows Command Variables
Time	Get/Set System Time
Ver	Display System Program Versions
Verify	Get/Set DOS Verify Flag

***Windows Command* Commands**

These commands control the *Windows Command* environment:

?	Access Windows Command Help
About	Shows Windows Command author information
Alias	Set an Alias for a Command
Beep	Beep the Speaker
Break	Get/Set Break Checking Status
Cls	Clear the Screen
Color	Change Screen Colors
Echo	Echo Line, or Get/Set Echo Status
Eset	Get/Set Windows Command Environment Variables
Exit	Exit Windows Command
Help	Access Windows Command Help
History	View and Recall Command History
Inkey	Reads a user keystroke into a variable
Input	Reads user input into a variable
Menu	Set or Reset User-Definable Menus
Prompt	Set Command Prompt
Set	Get/Set DOS/Windows Command Variables
Title	Set Information Displayed in Title Bar
Unalias	Remove an Alias Name
Unset	Remove Windows Command Variables

Windows Commands

These commands control the Windows environment:

Activate	Activates a Program
Arrange	Arranges the icons on the desktop
Assoc	Get/Set file extension associations
Cdx	Change Directory and Execute
Clipload	Load a file to the Clipboard
Close	Close a Program
Maximize	Maximize a Program
Minimize	Minimize a Program
Restore	Restore a Minimized Program
Runmax	Load and Maximize a Program
Runmin	Load and Minimize a Program
Send	Send Commands/Data to Applications
Task	Display all Running Programs
Winclass	Display Window Class Information
Winexit	Exit Windows
Wininfo	Display Window Information
Winmove	Move a Program Window
Winsize	Resize a Program Window

Windows Command Command Index

This is a complete list of *Windows Command* Commands.

?	Access Windows Command Help
About	Shows Windows Command author information
Activate	Activates a Program
Alias	Set an Alias for a Command
Arrange	Arranges the icons on the desktop
Assoc	Get/Set file extension associations
Attrib	Get/Set File Attributes
Beep	Beep the Speaker
Break	Get/Set Break Checking Status
Cd	Change Current Directory
Cdx	Change Directory and Execute
Chdir	Change Current Directory
Clipload	Load a file to the Clipboard
Close	Close a Program
Cls	Clear the Screen
Color	Change Screen Colors
Copy	Copy a File
Date	Get/Set the System Date
Del	Erase a File
Dir	Display a Directory of Files
Echo	Echo Line, or Get/Set Echo Status
Erase	Erase a File
Eset	Get/Set Windows Command Environment Variables
Except	Execute on All Except Specified Files
Exit	Exit Windows Command
Findfile	Find Files on an Entire Drive
Goto	Branch to a Label
Help	Access Windows Command Help
History	View and Recall Command History
If	Conditional command Execution
Inkey	Reads a user keystroke into a variable
Input	Reads user input into a variable
Maximize	Maximize a Program
Md	Make a New Directory
Mem	Display Free Memory and System Statistics
Menu	Set or Reset User-Definable Menus
Minimize	Minimize a Program
Mkdir	Make a New Directory
Move	Move a File
Path	Display DOS Path
Pause	Pause Until User Presses a Key
Print	Print a File
Prompt	Set Command Prompt
Pwd	Print Working (current) Directory

Rd	Remove a Directory
Rem	Remark or Null Command
Ren	Rename a File
Rename	Rename a File
Restore	Restore a Minimized Program
Rmdir	Remove a Directory
Runmax	Load and Maximize a program
Runmin	Load and Minimize a program
Send	Send Commands/Data to Applications
Ser	Get a disk serial number
Set	Get/Set DOS/Windows Command Variables
Task	Display all Running Programs
Time	Get/Set System Time
Title	Set Information Displayed in Title Bar
Type	Display a file on the screen
Unalias	Remove an Alias Name
Unset	Remove Windows Command Variables
Ver	Display System Program Versions
Verify	Get/Set DOS Verify Flag
Vol	Get a Disk Volume Label
Winclass	Display Window Class Information
Winexit	Exit Windows
Wininfo	Display Window Information
Winmove	Move a Program Window
Winsize	Resize a Program Window

About

Purpose:

Shows the author and registered user information for Windows Command.

Usage:

ABOUT

Example:

ABOUT will display a dialog box containing the author and registered user information for Windows Command.

Activate

Purpose:

To activate an application's window.

Usage:

ACTIVATE <taskid>

Example:

ACTIVATE `MICROSOFT WORD` will activate the matching program.

Alias

Purpose:

Sets an alias for a command or series of commands. The alias acts as a Windows Command internal command, and will override existing commands of the same name.

Usage:

ALIAS aliasname command(s)

Example:

ALIAS MV MOVE %*%	will allow you to type MV to execute a MOVE command.
ALIAS DIR DIR /W %*%	will make DIR give a default wide display.

Arrange

Purpose:

Arranges the icons on the desktop.

Usage:

ARRANGE

Example:

ARRANGE   will arrange the icons on the desktop so that they are all in neat rows.

Assoc

Purpose:

Get or set associations between a filename extension and the program that uses the file with that given extension.

Usage:

ASSOC <LIST|extension> <program name>

<extension> is the 3-characters following the "." in the filename.

<program name> is the full pathname of the program that uses files with the extension given.

Example:

ASSOC LIST will display a listing of all the existing filename associations.

ASSOC BMP will show the current setting for the BMP extension and prompt you for whether you want to remove that association or not.

ASSOC WCB C:\WINDOWS\WCOMMAND.EXE will create an association between files with the extension ".WCB", and Windows Command. After executing this command, whenever a file with the extension ".WCB" is run from Program Manager, File Manager, Windows Command, or any other program that uses associations, Windows Command will automatically be started.

Attrib

Purpose:

To Display or set the Read-Only, System, Hidden or Archive attributes of a file or subdirectory.

Usage:

ATTRIB [+r|-r] [+s|-s] [+h|-h] [+a|-a] filename [/D]

+r/-r	Add/Remove Read Only Attribute
+s/-s	Add/Remove System File Attribute
+h/-h	Add/Remove Hidden Attribute
+a/-a	Add/Remove Archive Attribute
/D	Set or Display Subdirectory Attributes

Example:

ATTRIB *.* will display the attributes of all files in the current directory.

ATTRIB -R +S -H +A THISFILE.DOC will add System and Archive attributes to and remove Read-Only and Hidden attributes from the file THISFILE.DOC.

Beep

Purpose:

To Beep the Speaker.

Usage:

BEEP

Example:

BEEP will cause the speaker to sound a tone.

Break

Purpose:

To Display Ctrl-Break Checking Status, or turn it on or off.   The default is ON.

Usage:

BREAK [ON|OFF]

Example:

BREAK	will show whether break checking is on or off.
BREAK ON	will set break checking on.

Cdx

Purpose:

To automatically change to the directory in which a program resides when executing a program. This will allow programs which expect to be executed from the same directory as they are in to run properly.

Usage:

CDX <program>

Example:

CDX C:\123\LOTUS will execute the command LOTUS from the directory C:\123.

CDX PROGMAN will start Program Manager from your windows directory.

Chdir or Cd

Purpose:

To change the current directory on the default or specified drive.

Usage:

CHDIR <directory>

Example:

CHDIR \ 	will change the current directory to the root of the default drive.

CD A:\ 	will change the current directory of the A drive to the root directory; the current drive will remain the same.

Clipload

Purpose:

To load a file into the clipboard.   The file size must be less than 64K.

Usage:

```
CLIPLOAD <filename>
```

Example:

CLIPLOAD THISFILE.DOC will load the file THISFILE.DOC to the clipboard.

Close

Purpose:

To Close a currently active task. A message is sent to the program, telling it to close; whether or not the task actually ends will depend on the application itself. Task IDs can be found by issuing the TASK command.

Usage:

CLOSE <taskid>

Example:

CLOSE 9337 will close the task with the Task ID 9337.

Cls

Purpose:

Clears the *Windows Command* Window.

Usage:

CLS

Example:

CLS   will clear the screen.

Color

Purpose:

To change the colors of the Windows Command Window.

Usage:

COLOR <color> [ON <color>]

<color> is one of the following:

RED
GREEN
BLUE
YELLOW
MAGENTA
CYAN
BLACK
WHITE

Example:

COLOR BLACK ON WHITE will set the text color to black and the background color to white.

Copy

Purpose:

To Copy or Append a file to another file.

Usage:

COPY [/A|/B]<fromfile> [+ [/A|/B] <fromfile2> ...] <tofile>

/A	Copy remaining files in Ascii mode
/B	Copy remaining files in Binary mode

Example:

COPY THISFILE.DOC /A + THISTOO.DOC A:\THATFILE.DOC will copy THISFILE.DOC and append THISTOO.DOC to the file A:\THATFILE.DOC.

Date

Purpose:

Set the System Date.

Usage:

```
DATE <month> <day> <year>
    <month> is an integer between 1 and 12
    <day> is an integer between 1 and 31
    <year> is the full number of the year (i.e. 1960, not 60)
```

Example:

```
DATE 7 4 1991        will set the date to July 4th, 1991.
```

Dir

Purpose:

To display a directory listing of files.

Usage:

DIR [/1/2/4/W/P/B/U/N/T/S/Onedsgu/Adrsha] [<filename>]

/1	display one column of files (default)	
/2	display two columns of files	
/4	display four columns of files	
/W	display five columns of files	
/P	pause after each page of files	
/B	display only filenames, no other information	
/U	display filenames in upper case	
/N	do not display file attributes in listing	
/T	display only directory size totals and number of files	
/S	display all files in current directory and subdirectories	
/O	sort order	
	-	Reverse any of the below orders
	N	by Name
	E	by Extension
	D	by Date
	S	by Size
	G	Directories First
	U	Unsorted
/A	display only files with given attributes	
	-	display all files except with given attributes
	D	Directory
	R	Read Only
	S	System
	H	Hidden
	A	Archive

Example:

DIR /W	will display the files in the current directory in wide (5-column) format.
DIR A:\DIRECT	will display the files in the directory A:\DIRECT.

Echo

Purpose:

To display whether echo is on or off, set the echo state, or echo a line to the screen. Echo defaults to Off.

Usage:

ECHO [ON|OFF|<line to be echoed>]

Example:

ECHO	will display whether echo is on or off.
ECHO OFF	will turn echo off.
ECHO HI THERE	will echo "HI THERE" to the screen.

Erase or **Del**

Purpose:

To erase (or delete) a file.

Usage:

```
    ERASE <filename> [/P]
or  DEL <filename> [/P]
        /P      to prompt for each file delete
```

Example:

DEL *.DOC /P   will delete all files with the extension .DOC, and prompt the user for each file deleted.

Eset

Purpose:

To get or set Windows Command environment variables.

Usage:

ESET <variable name> <new value>

<variable name> is one of the following:

HistMinLen is the minimum number of characters a command must have to be saved into the command history. <new value> must be between 1 and 255. Default size is 3.

HistLines is the number of commands that will be saved into the command history. <new value> must be between 1 and 255. Default value is 30.

ScrBufLines is the number of lines the screen buffer will hold in memory. <new value> must be between 25 and 999. Default value is 60.

Beep determines whether the warning beep is heard. <new value> must be ON or OFF. Default is ON. This will not affect the Beep command.

Menu determines whether the menu is visible or not. <new value> must be ON or OFF. Default is ON.

Example:

ESET SCRBUFLINES 100 will set the screen buffer size to 100 lines and allow you to scroll back up 100 lines.

ESET MENU OFF will remove the menu from the top of the Windows Command window.

Except

Purpose:

To execute a command on files except for a series of specified files.

Usage:

EXCEPT (<filename>...) <command and arguments>

Example:

EXCEPT (*.DOC *.TXT) DEL *.*		will delete all files in the current directory except for those with extensions .DOC and .TXT.

Exit

Purpose:
To exit Windows Command.

Usage:
EXIT

Example:
EXIT   will cause Windows Command to close.

Findfile

Purpose:

To search an entire drive for files. Findfile searches for matching files recursively from the directory given in the pathname argument. If no directory is given, the entire current drive is searched, starting at the root directory.

Usage:

FINDFILE <pathname>

<pathname> an "imaginary" filename to search for. **Findfile** takes the directory portion of the pathname as the directory to begin the search in. The filename portion is what is searched recursively through the subdirectories for.

Example:

FINDFILE C:\*.BAK will search the C: drive starting at the \ (root) directory for all files with a .BAK extension

FINDFILE D:\WINDOWS\*.EXE will search the D: drive starting at the \WINDOWS directory for all files with a .EXE extension.

Goto

Purpose:

To branch to a label in a batch command. For more information on labels in batch files, read the Batch Files section.

Usage:

GOTO <label>

Example:

GOTO RALPH will go to the label ":Ralph" in the current batch file, and continue executing commands from that point in the file.

Help or **?**

Purpose:

To access the Windows Command help file.

Usage:

HELP <topic>

or **? <topic>**

Example:

HELP TYPE will bring up the help screen for the TYPE command.

History

Purpose:

To display a history of entered commands for browsing or recalling, to list or load history entries from a file, or to clear the history list.

Usage:

HISTORY [CLEAR|LIST|LOAD <filename>]

CLEAR	will erase all entries from the history list
LIST	will list the history list to the screen
LOAD	will load history entries from a file
<filename>	the file to load history entries from

To create a file of history entries, redirect the output of the HISTORY LIST command to a file.

Example:

HISTORY will create a dialog box with all the commands held in the command history list.

HISTORY LIST > HISTORY.LST will save the current history list to a file

HISTORY LOAD HISTORY.LST will load history entries from the file HISTORY.LST

If

Purpose:

Allows conditional execution of a command. Note that when comparing two strings, variables or parameters using the " == " operator, you **must** leave a space before and after the " == ".

Usage:

IF [NOT] < ERRORLEVEL <number> | EXIST <filename> > | "[<string>|<variable name>|<parameter id>]" == "[<string>|<variable name>|<parameter id>]" <command>

NOT will make the <command> execute if the case given is not true.
ERRORLEVEL will return true if the last command exited with a return code of <number>
EXIST will return true if the file <filename> exists.
<command> is the command that will execute if the conditions are correct.
<string> is any text string
<variable name> is any Windows Command variable name, inside percent signs
<parameter id> is any parameter substitution identifier

Example:

IF ERRORLEVEL 0 CLS will clear the screen only if the last command executed returned an error code of 0.

IF NOT EXIST C:\WINDOWS\WCOMMAND.EXE DIR C:\ will execute a display of the directory C:\ only if the file WCOMMAND.EXE is not found in the C:\WINDOWS directory.

IF "%THISVAR%" == "MICROSOFT" ECHO WINDOWS PARENT! will echo the line "WINDOWS PARENT!" to the screen if the variable THISVAR is set to "MICROSOFT".

Inkey

Purpose:

To prompt the user to press a key, then put that input into a variable.

Usage:

INKEY <prompt string> <variable name>

Example:

INKEY `ENTER ANY NUMBER: ` ANYNUMBER will prompt the user to ENTER ANY NUMBER: , and then will wait for the user to press a key, and will put that key into the variable ANYNUMBER.

Input

Purpose:

To prompt the user for input, then put that input into a variable.

Usage:

INPUT <prompt string> <variable name>

Example:

INPUT `ENTER ANY NUMBER: ` ANYNUMBER will prompt the user to ENTER ANY NUMBER: , and then will wait for the user to enter a string, followed by <ENTER>, and will put that input string into the variable ANYNUMBER.

Maximize

Purpose:

To maximize a task. Task IDs can be found by issuing the TASK command.

Usage:

MAXIMIZE <taskid>

Example:

MAXIMIZE 9337 will maximize the task with the task ID number 9337.

Mem

Purpose:

To display system statistics, free RAM, and free memory on the default drive.

Usage:

MEM

Example:

MEM will display the Windows mode, CPU type, screen resolution, free RAM and free disk space.

Minimize

Purpose:

To minimize a task. Task IDs can be found by issuing the TASK command.

Usage:

MINIMIZE <taskid>

Example:

MINIMIZE 9337 will minimize the task with the task ID number 9337.

Mkdir or **Md**

Purpose:

To make a new directory.

Usage:

MKDIR <directory>

or MD <directory>

Example:

MD NEWDIR will make a subdirectory named NEWDIR in the current directory.

Menu

Purpose:

To set or reset the Windows Command user-definable menus from the given file.

Usage:

MENU <filename>

Example:

MENUwill load the menu from the file WCOMMAND.MNU.

Move

Purpose:

To move a file.

Usage:

MOVE <fromfile> <tofile>

Example:

MOVE *.DOC A:\*.TXT will move all files with a .DOC extension to the A drive in files with the same name, but with the extension .TXT.

Path

Purpose:

Displays the DOS Path.

Usage:

PATH

Example:

PATH		will display the current Path set in DOS.

Pause

Purpose:

To prompt the user to press a key before continuing execution.

Usage:

PAUSE

Example:

PAUSE   will display a prompt and wait for the user to press a key.

Print

Purpose:

To print a file through the Windows print queue.

Usage:

PRINT <filename>

Example:

PRINT *.DOCwill print all files with the extension .DOC.

Prompt

Purpose:

To change the command prompt.

Usage:

PROMPT <string>

<string> is any text, which may include the following special strings:

$p	current drive and path in lower case
$P	current drive and path in upper case
$G/$g	the ">" character

Example:

PROMPT HI pg will set the prompt to "HI " plus the current directory plus the ">" character.

Pwd

Purpose:

Prints out the current working, or current, directory.

Usage:

PWD

Example:

PWD   will display the current directory.

Rem

Purpose:
A null, or "do nothing" command. REM can be used to echo a line to the screen if echo is on.

Usage:
REM <text>

Example:
REM HI THERE will do nothing if echo is off or write "REM HI THERE" to the screen if echo is on.

Rename or **Ren**

Purpose:

To rename a file or directory.

Usage:

```
    RENAME <dirname|filename> <newname>
or  REN <dirname|filename> <newname>
```

Example:

REN *.DOC *.TXT		will rename all files with a .DOC extension to a file of the same name, but with a .TXT extension.

Restore

Purpose:

To restore a currently minimized task. A message is sent to the program, telling it to restore; whether or not the task actually does will depend on the application itself. Task IDs can be found by issuing the TASK command.

Usage:

RESTORE <taskid>

Example:

RESTORE 9337 will restore the application with the task ID 9337.

Rmdir or **Rd**

Purpose:

To remove an empty directory.

Usage:

```
    RMDIR <directory>
or  RD <directory>
```

Example:

```
RD OLDDIR        will remove the directory OLDDIR from the current directory.
```

Runmax

Purpose:

To load and maximize a program.

Usage:

RUNMAX <program>

Example:

RUNMAX THISAPP.EXE will execute the program THISAPP.EXE, and initially maximize it.

Runmin

Purpose:

To load and minimize a program.

Usage:

RUNMIN <program>

Example:

RUNMIN THISAPP.EXE	will execute the program THISAPP.EXE, and initially minimize it.

Send

Purpose:

Sends a menu command sequence or character sequence to a program. Note that many applications use Multiple Document Interfaces, which means that more than one window is running inside another window. To SEND to a window inside an application, you must find the Task ID for the appropriate child window by using the Task command.

Usage:

SEND <taskid**> <string>**

<string> includes any text that you wish to send, or **[MENU,<hotkeys>]** to send a series of hotkeys to a program menu.

Example:

SEND `MICROSOFT WORD` Hi will send the string "Hi" to the window matching the name "Microsoft".

SEND `MICROSOFT WORD` [MENU,FO] will activate the menu in the given program, and select the menu items with hotkeys "F" and then "O" (for example, **F**ile-**O**pen).

Ser

Purpose:

To display a disk's DOS volume serial number.

Usage:

SER <disk>

Example:

SER will display the serial number from the default drive.

SER A: will display the serial number of the disk in drive A:.

Set

Purpose:

To display the DOS environment variables and display or set Windows Command variables. Windows Command variable names may be up to 30 characters in length.

Usage:

SET [DOS|WC|<variable name> <value>]

DOS	display only DOS variables
WC	display only Windows Command variables
<variable name>	any string up to 30 characters in length to use as a variable name
<value>	the value to set the variable to, multi-word strings must be put inside back quotes (`)

Example:

SET	will display all currently set DOS and Windows Command variables.
SET DOS	will display only DOS environment variables.
SET WC	will display only Windows Command variables.
SET THISVAR 1000	will set a Windows Command variable named "THISVAR" to the string "1000"

Task

Purpose:

To display all currently active tasks and their associated task IDs. If a Task ID is given, that window's child windows will be displayed. Task sets a Windows Command variable LASTTASK if a task ID is given.

Usage:

TASK <taskid**>**

Example:

TASK will display all the currently active tasks in a list.

TASK `MICROSOFT WORD` will display all the child windows for the matching program.

Time

Purpose:

To display or set the system time.

Usage:

TIME [<hour> <minute> <second>]

Example:

TIME	will display the current time.
TIME 0 50 30	will set the current time to 12:50AM and 30 seconds.

Title

Purpose:

To set or reset the information displayed in the Windows Command title bar.

Usage:

TITLE [CWD|UCWD|DATE|TIME|DATETIME|RESET]

The options put the following information is put into the title bar:

CWD	current "working" directory
UCWD	upper case current "working" directory
DATE	current date
TIME	current time
DATETIME	current date and time
RESET	resets the title bar to default (no information displayed)

Example:

TITLE TIME	will display the time in the Windows Command title bar.
TITLE RESET	will set the title bar to the default - program name only.

Type

Purpose:

To display a file on the screen.

Usage:

TYPE <filename>

Example:

TYPE *.DOC will display all the files with the extension .DOC on the screen.

Unalias

Purpose:

To remove a previously assigned command alias.

Usage:

UNALIAS <aliasname>

Example:

UNALIAS CDD	will remove the alias CDD from the alias list, if it was previously defined.

Unset

Purpose:

To remove a previously assigned Windows Command variable.

Usage:

UNSET <varname>

Example:

UNSET NAME will remove the variable NAME from the Windows Command environment, if it was previously defined.

Ver

Purpose:

To show the versions of DOS, Windows, and Windows Command currently running.

Usage:

VER

Example:

VER   will display the versions of DOS, Windows and Windows Command on the screen.

Verify

Purpose:

To get or set the DOS verify flag. If verify is on, all disk writes will be verified to ensure they were written correctly. The default is off. Turning verify on will slow disk writes slightly.

Usage:

VERIFY [ON|OFF]

Example:

VERIFY	will display the state of the verify flag.
VERIFY OFF	will turn the verify flag off.

Vol

Purpose:

To display the current volume label on the default or specified drive.

Usage:

VOL <disk>

Example:

VOL A:	will display the volume label on the disk in drive A.

Winexit

Purpose:

To exit Windows.

Usage:

WINEXIT

Example:

WINEXIT will prompt the user to verify whether or not to exit windows. If the user presses "Y", all currently running applications will be closed, and Windows will terminate.

Winclass

Purpose:

To display an window's class information. This information includes the class name, class module handle, module file name, class window function, class icon handle, class cursor handle, class background brush, window information extra bytes, class information extra bytes, class style, and class style names. Where appropriate, the cooresponding numbers are given in both hexadecimal (followed by an 'x'), and decimal (followed by a 'd') (function addresses are all in hexadecimal).

Usage:

WINCLASS <taskid>

Example:

WINCLASS `MICROSOFT WORD`will display the window class information on the matching program window.

Wininfo

Purpose:

To display information on a window. This information includes the window text (title bar text), window handle (the "Task ID" number), parent window handle (if this number is not 0, the window is a child window), application instance handle, executable file name, window function address, menu handle, window rectangle with respect to the screen, window rectangle with respect to its parent window (only if it is a child window), window client rectangle, window style, window style names. Where appropriate, the cooresponding numbers are given in both hexadecimal (followed by an 'x'), and decimal (followed by a 'd') (function addresses are all in hexadecimal).

Usage:

WININFO <taskid>

Example:

WININFO `MICROSOFT WORD` will display information on the matching program window.

Winmove

Purpose:

To move an application's window.

Usage:

WINMOVE <taskid>

Example:

WINMOVE `MICROSOFT WORD` will initiate a keyboard-based move command on the matching program.

Winsize

Purpose:

To resize an application window.

Usage:

WINSIZE <taskid>

Example:

WINSIZE `MICROSOFT WORD`   will initiate a keyboard-based resize command on the matching program.

Error Message Codes

Error Code	Error Message
10	No Closing Quote
15	Interrupt Key Pressed
20	Insufficient Memory
21	Invalid Command Name
22	Unknown Command
23	Launch Failed
30	Invalid Command Usage
31	Invalid Parameter
40	Invalid Drive
41	Insufficient Disk Space For
50	Path Not Found
51	Invalid Path
60	Cannot Change To Directory
70	File Not Found
71	File Exists
72	File Cannot Be Copied To Itself
79	Cannot Create Directory
80	Cannot Remove Current Directory
81	Error Removing
82	Error Renaming
83	Error Moving File
90	Access Denied
100	Unable To Exit Windows
110	Invalid File Size
120	Invalid Window Task Id
121	Task Not Found
130	Printer Initialization Failed
200	Label Not Found
210	Error In Menu File
220	Invalid Parameter Substitution

Menu File Syntax

Menu files allow you to customize the Windows Command menu. You can edit a menu file by using Notepad, or any other text editor. The file itself should have an .MNU file extension, and the file is loaded using the Menu command.

The menu file syntax consists of structures for submenus, executed and non-executed menu items, separating lines, columns, menu hot keys, and comments.

Submenus

All menu files must start with a submenu declaration. This submenu will be the top level menu that will take the place of the **Execute** default submenu. The syntax for submenus is as follows:

```
Submenu Name{
	Submenu Name 2{
	}
}
```

Any leading spaces or tabs will be ignored when reading the submenu name, and the name will end with the open bracket ({). The submenu will end with the closing bracket (}), and submenus can be nested inside of each other

Menu Items

A menu may have as many as 2000 total menu items. After this, additional menu items will be ignored. The syntax for menu items is as follows:

```
Menu Item Name=Command Line
```

or

```
Menu Item Name 2=;Command Line 2
```

Any leading spaces or tabs will be ignored when reading the menu item name, and the name will end with the equals sign (=). The command line is considered to be all the text after the equals sign, including trailing spaces. It is this command that will be typed and executed at the command line when the menu item is selected.

If you precede the command line with a semicolon (;), the command will only be echoed to the command line and not executed when you select the menu item. You must be sure there are no spaces between the equals sign and the semicolon.

Separators, and Columns

The following are special menu item names that create various types of separations in menus:

```
@COL@=
@COLBAR@=
@LINE@=
```

@COL@ will create a new column for menu items. @COLBAR@ will create a new column separated by a vertical line. @LINE@ will create a separating horizontal line.

Comments

Any line with a semicolon (;) in the first character position is considered a comment. Be careful not to allow any spaces or tabs before the semicolon.

Menu Hot Keys

To create a hot key to a menu item or submenu, put an ampersand (&) before the character in the item name that you want to use as the hot key. When the menu is loaded, you will see an underline below the character that you selected, and if you press that letter when the menu is active, it will be selected. Be careful to select unique letters within each submenu. Submenus with items that have more than one of the same letter selected as a hot key, may not allow you to select the correct menu item with the letter keys.

Arguments

Windows Command splits up the command line into arguments by scanning the line for text separated by "White space" characters and command option delimiters.

"White space" characters include semicolons (;), commas (,), spaces, and tabs.

Command option delimiters include the foreslash (/), and the dash (-), since any command option in Windows Command can be set apart by either a foreslash or a dash.

To process a string containing any of the above characters as a single argument, you must enclose the string in back quotes (`).

Using Quotes

Windows Command uses back quotes (`) and double quotes (") to allow the literal translation of strings. Back quotes are removed from the command line when sending the command, and double quotes are left in, and sent to the command. For example,

"DIR STRING"

would be sent to the command processor as "DIR STRING", and

`DIR STRING`

would be sent to the command processor as DIR STRING.

Any argument which contains "white space" characters, command-option delimiters, multiple command identifiers, or any other reserved character, must be enclosed in double quotes to be translated properly. Double back quotes (``) or double double quotes ("") that are not part of another quoted argument are translated as a single quote in the final string. For example,

"HI""THERE" or `HI``THERE`

would be translated as first argument "HI", second "THERE".

HI""THERE or HI``THERE

would be translated as one argument HI"THERE or HI`THERE.

Quoted arguments appear in the echoed command if echo is on, but are removed when the command is broken into arguments.

Task IDs

Any command with a TaskID argument will accept either a Task ID number, or a title or portion of a title displayed in a program's title bar. Task ID numbers and program titles can be found using the Task command.

If a title string is used, the first matching top-level program (no child window names will be searched), searched in the same order shown by the **Task** command, will be acted upon. For example, if the Task command shows:

```
  ID      Name
7900   MICROSOFT WORD
8000   MICROSOFT EXCEL
```

then typing `MICROSOFT` as a Task ID will act on Task 7900. To act upon Task 8000, you must enter at least `MICROSOFT E`.

If you need to enter a window title with more than one word, you will have to enclose the title in back quotes, to ensure that the title is interpreted as one argument.

Windows Command License Agreement

This document contains information on the legal and ethical use of all versions of Windows Command. Using Windows Command implies agreement with all the terms described below.

1) COPYRIGHT INFORMATION

Windows Command is a copyrighted program, protected under United States Copyright Law and provisions in international treaties. These laws prohibit unauthorized copying, and other unlawful incorporation of any portion of this program into any other work without express permission from the author. Violation of the terms of this license agreement is unlawful.

Under no circumstances may any version of Windows Command be reverse assembled, reverse compiled, or translated in any way.

2) LIMITATION OF LIABILITY

Windows Command is a powerful command-line interface for Windows, and it provides access to many low-level operating system functions. Because of this, you should use Windows Command carefully. You assume full responsibility for all results brought about by commands issued intentionally or accidentally at the Windows Command command line.

The limititation of liability for Windows Command is as follows: in no event will I be held liable for any consequential or incidental losses of profits, business interruption, loss of records or data or any other loss arising from the use of Windows Command, even if I have been forewarned of such damages. I will also not be held liable for consequenses arising from incompatibilities of Windows Command with third-party products.

3) SHAREWARE VERSION

Evaluation of the shareware version of Windows Command takes place in a 21-day period after the first installation on your machine. This period of time should be enough for you to decide whether or not the program will meet your needs, and whether you would like to register. AFTER THIS EVALUATION PERIOD EXPIRES, YOU MUST EITHER REGISTER, OR STOP USING WINDOWS COMMAND. COPYING OF THE SHAREWARE VERSION IS PERMITTED ONLY IF ALL THE ORIGINAL FILES CONTAINED IN THE ORIGINAL SHAREWARE RELEASE MADE BY MICHAEL B. TIERNEY ARE LEFT TOGETHER, UNCHANGED IN ANY WAY, AND ARE TRANSFERRED AS A COMPLETE PACKAGE.

4) REGISTERED VERSION

A user is registered when the complete registration package, including the required information and registration fee is received at the address listed below. The registered copy of Windows Command will be mailed as soon as possible. Once you mail your complete registration package, you may continue to use the shareware version of Windows Command until your registered version arrives. Registered users of Windows Command are granted a license to use the program within the rights provided in this document. The registered user will own the magnetic media that Windows Command is recorded on, but the entire code recorded on that medium remains the property of Michael B. Tierney; the title to the program itself is not transferred upon registration. This license is not a sale of the original code or any copy. It is a sale of the right to execute the code in a normal fashion under Microsoft Windows.

COPYING OF THE REGISTERED VERSION IS PERMITTED ONLY BY THE USER TO WHOM THE PARTICULAR COPY OF WINDOWS COMMAND IS REGISTERED, AND MAY BE DONE SO FOR USE ONLY BY THE REGISTERED USER, AND FOR NO OTHER INDIVIDUAL. A registered user may not modify the Windows Command program, or incorporate any portion of the program into any other work, without first obtaining express written consent from Michael B. Tierney. MODIFYING WINDOWS COMMAND, SELLING OR GIVING AWAY OF A COPY OF THE REGISTERED VERSION OF WINDOWS COMMAND TO ANY THIRD PARTY, OR USE OF A REGISTERED VERSION OF WINDOWS COMMAND BY AN INDIVIDUAL OTHER THAN THE REGISTERED USER IS A VIOLATION OF THIS LICENSE AND THE ABOVE-MENTIONED FEDERAL COPYRIGHT LAWS. The registered user for each registered copy of Windows Command will be displayed in the "About" dialog box, which can be viewed by issuing the "About" command at the Windows Command command-line, or by selecting Help-About from the menu.

When updates of Windows Command are released, registered users will be given the option of a free or reduced-price upgrade. If an upgrade is purchased, the original license will cover the upgrade as well as the original product.

REGISTERED USERS MAY NOT SUBLICENSE, ASSIGN, OR TRANSFER THE LICENSE FOR

WINDOWS COMMAND, NOR CAN THEY SELL, LEASE OR RENT THE PROGRAM. If a registered user fails to comply with all provisions of their license, the license will be considered terminated. When the license is terminated by either user request, or by violation of the license agreement, all copies of the software must be deleted.

5) USE RESTRICTIONS

The shareware version of Windows Command may be used on as many computers as is needed so long as the rules above are strictly followed. The same is true for the registered version, however, only one machine may be running a copy of the program at any one time, and the program may not be transferred electronically over a network to another computer. After 21 days, the shareware version must be either registered or discarded, and only the registered user may use his/her registered copy of Windows Command.

6) NETWORK REGISTRATION

Windows Command may be registered for a network or multi-user machine by purchasing a discounted multi-user license. When using Windows Command on a network or multi-user machine, the system's administrator must understand that use of Windows Command by a non-registered user (any user other than users of the registered network or machine) is illegal, and the administrator will be held legally responsible for enforcing the policies of this license agreement on their system, and insuring that only authorized users have access to Windows Command.

To register Windows Command for use on a network or multi-user machine, please write to me at the address below. Please include in your letter the number of users on your system, how you would like to provide access to Windows Command (whether on a server, or on individual workstations), the number of copies of program disks you would like to receive, and an address, telephone or FAX number where you can be contacted. You will then receive further instructions on pricing, and how to place your order.

7) QUESTIONS OR PROBLEMS

While I have carefully tested Windows Command and all its features for proper function, I cannot guarantee that there will be absolutely no bugs or other program errors (no software designer can). If you detect a problem that can be reproduced consistently, I ask that you please contact me as soon as possible, and I will do my best to correct the problem. My address is listed below.

If you have any questions regarding any of the above information, please feel free to contact me at either of the below addresses:

Michael B. Tierney
205 Inglewood Dr.
Pittsburgh, PA 15228 USA

or

Compuserve: 70604,1512
Internet/Bitnet: 70604.1512@compuserve.com