WINCMD: A Windows Command Language

BY DOUGLAS BOLING

Remember the DOS batch files with which you once proudly automated all your routine tasks? If you've joined the mad rush to Windows, you've simply had to learn to do without this useful and convenient facility. The DOS batch file language may have been limited, but Windows supported nothing at all in its place—until now.

WINCMD.EXE, this issue's free utility, is a Windows-based command file interpreter that lets you create simple Windows programs that perform the repetitive functions that were once the domain of DOS batch files. WINCMD goes far beyond the simple DOS batch file language, however. Its language includes looping and branching statements along with user-defined subroutines and functions. Both numeric and string variables are supported, as is a full suite of arithmetic and logical operators.

You can use any ASCII editor, such as Notepad, to create WINCMD programs. Typically, these programs will carry a .WCM extension, though this is not required unless you want the program to be callable from another WINCMD program. A single .WCM file can be up to 64K in size, though its executable parts (those not made up of comment lines) are limited to about 45K.

Both the WINCMD program and its C source files can be downloaded from PC MagNet as WINCMD.ZIP. See the "Utilities by Modem" sidebar for instructions. You can also get the files by sending a postcard with your name, address, and disk size preference (360K, 720K, 1.2MB, or 1.44M) to the attention of Katherine West, *PC Magazine,* One Park Ave., New York, NY 10016-5802. No phone calls, please!

USING WINCMD Before discussing how to use WINCMD, I must start with some terminology. WINCMD.EXE is the Windows program I've written. It interprets programs—which are ASCII files—that are written in the WINCMD language. To keep the two kinds of programs separate, I'll use the full name and extension, WINCMD.EXE, when referring to the utility itself. I'll refer to the ASCII files that WINCMD.EXE interprets as WINCMD programs.

To install WINCMD.EXE, just copy it, along with WINCMD.ICO (its icon file) and any .WCM files you have (two

WNDCMD.EXE, a Windows counterpart to DOS's batch language, lets you automate tasks in the environment and even goes beyond its ancestor's capabilities.

are supplied with the program), into a directory on your path. Select the File | New menu item in the Program Manager and choose Program Item. Enter the short title you want to appear under the icon as the Description, and enter the complete path and WINCMD.EXE on the command line.

To make it easy to run WINCMD programs, you should then also use the File Manager to associate the .WCM file extension with WINCMD.EXE. To do this, start the Windows File Manager and select the File | Associate menu item. When the dialog box appears enter WCM, the WINCMD default extensio in the edit box. In the Combo Box th lists the programs, enter the path to a filename of WINCMD.EXE. Click o the OK button and you're set!

From now on, any time you click o a WINCMD file in the File Manager o Program Manager, Windows will automatically start WINCMD.EXE and pas it the name of your WINCMD program When WINCMD.EXE is started with a filename on the command line, it will immediately start executing that file. When WINCMD.EXE has finished executing the WINCMD program, it will gracefully terminate.

Since debugging is important even for WINCMD programs, WINCMD.EXE provides help in this area. If WINCMD.EXE is started with an /L command switch *before* the WINCMD program name, WINCMD.EXE will display a window with your WINCMD program loaded into memory, ready to execute. Be sure not to place the /L after the WINCMD program name, or it will be considered an argument to the command file. When WINCMD.EXE starts, it performs some preprocessing on your WINCMD program. If it finds any lines it does not understand, you'll get appropriate print error messages with line numbers to help you find the problem.

Error messages are displayed for such faults as lines being too long or numbers too big. When an error message is printed, WINCMD.EXE changes to display two windows. The listing window displays the WINCMD program with the line highlighted that is to be executed next. The output window displays the last 100 lines printed by the WINCMD program as well as the error messages.

WINCMD.EXE allows you to run a WINCMD program from the window i two different ways. The program can b

WINCMD Statements and Functions

Statement	Description
IF{expression} [THEN] {statement} [ELSE] {statement}]	Branching statement
DO {statement} {statement} . . . {statement}END	Block statement
WHILE {condition} {statement}	Loop statement
EXIT [return code]	Terminates a WINCMD program
RETURN [return code]	Terminates a subroutine
LEAVE	Exits from a loop
PRINT {expression}	Displays a line of text to the user
SAY {expression}	Displays a line of text to the user
REM {text}	Comment, remainder of text on the line ignored
// {text}	Comment, remainder of text on the line ignored
' {text}	Comment, remainder of text on the line ignored

Function	Description
LENGTH(string)	Returns the length of a string.
UCASE(string)	Returns a copy of the string in upper case characters.
SUBSTR(string, start, length)	Returns a part of a string.
ARG(number)	Returns an argument to program or a subroutine.
DELAY(number)	Halts program execution for a number of milliseconds.
APPACTIVATE(win title text)	Activates the window with the matching title text.
GETAPPACTIVE (no parameters)	Returns the title text of the currently active window.
GETAPPEXE(win title text)	Returns the EXE file name for a window.
SENDKEYS(send key string)	Sends a series of keys to the active application.
MSGBOX(message, title, flags)	Displays a message box.
ASKBOX(text, default answer)	Displays a dialog box so that the user can enter a value.
TICKS(no parameters)	Returns the number of milliseconds since Windows was started.

***Figure 1:* These statements and functions are supported in the WINCMD language.**

run normally either by pressing the Run button or by selecting the Run | Run Program menu selection. The WINCMD.EXE window will then change to display only the output window and run the WINCMD program. The Run button text will change to Stop.

Pressing the Stop button will cause WINCMD.EXE to stop executing the WINCMD program and to display both the output and listing windows. Again, the line in the program that would have been executed next is highlighted. The program can then be restarted from this point simply by clicking on the Run button. If you want to start the program from the beginning, first click on the Reset button to reset the program and then hit the Run button to start the program on its way. Note that when WINCMD resets a program, it rereads the program from the disk. This means that you can make changes to the program and then you can have WINCMD load those changes by pressing the Reset button.

The other way to run a WINCMD program is by stepping through it one line at a time. Single stepping is very handy to see exactly how a WINCMD program is being executed. Each press of the Step button will cause WINCMD.EXE to execute one line of the WINCMD program. The listing window will highlight the next line to be executed.

WINCMD VARIABLES AND ARITHMETIC The WINCMD syntax combines the power of a high-level language with the simple command line operation of a batch language. Before discussing the different WINCMD statements in detail, I should explain how WINCMD programs use variables. The WINCMD language makes no distinction between string variables and numeric variables. That is, a variable that has been assigned a string value can later be assigned a number and vice versa. Variables do not have to be declared before they are used. As you would expect, variables are assigned using the equal sign.

The following three statements are examples of valid WINCMD assignments.

```
var = 1
var = magazine
var = "I like PC Magazine"
```

In the first assignment above, the variable VAR is given the value 1. The second assignment has the variable assigned the string "magazine." In the third assignment, VAR is assigned the string "I like PC Magazine." The quotes allow spaces to be included in the string assigned to VAR. If the quotes were not used, WINCMD would attempt to assign the variable VAR with the string "I", but would then find the additional characters on the line. WINCMD would then stop execution and print the error message "Extra characters on line."

Variable names may contain numbers, letters, or the characters _, $, %, and #, but the first character in a variable must always be either a letter or underscore (_). Variable names are not case-sensitive; the names BOB, Bob, and bob all refer to the same variable. Variable names can be up to 253 characters long. Variables can be used anywhere strings or numbers can be used in the program.

The numbers you use may range from -2,147,483,648 to 2,147,483,647, and can be expressed in binary, octal, decimal, or hexadecimal format. (In entering numbers, do *not* type in the commas shown above, however.) The WINCMD default is base 10. To enter a nondecimal number, start the number with a zero, then use b, o, or x as the second character, indicating a binary, octal, or hexadecimal number, respectively. The following four lines demonstrate different ways to write the decimal value 100.

```
var = 100
var = 0x64
var = 0o144
var = 0b1100100
```

WINCMD uses the standard set of arithmetic operators for addition (+), subtraction (-), multiplication (*), and division (/). Arithmetic comparisons can be performed for greater than (>), less than (<), greater than or equal to (>=), less than or equal to (<=), and not equal (<>) operators. Comparison operators return a nonzero value if the condition is true and a zero if the condition is false.

As in BASIC, the equal sign (=) performs double duty in WINCMD. When placed alone on a line to the right of a variable, it indicates an assignment, as in var = 5. In any other location, however, it indicates a test for equality. WINCMD also supports double equals (==) as an equality operator. The equality test works for both numbers and strings. For example, to test if the variable MONTH is "February", simply use the expression

```
month = "February"
```

WINCMD also includes a set of logical and bitwise operators. The logical operators AND, OR, XOR, and NOT act on the logical value. If a variable is 0, its logical value is FALSE; otherwise, its logical value is TRUE. For example, the numbers 5 and -3 are TRUE values while 0 is a FALSE value. The logical operators allow you to combine other tests into one condition. For example, the expression

```
DAY = "Monday" AND
 MONTH <> "February"
```

will evaluate to TRUE only if both DAY is equal to Monday and MONTH is not "February".

The bitwise operators ANDB, ORB, XORB, and NOTB perform logical operations on each bit in a number. For example, 5 ANDB 20 returns a value of 4. (00101 ANDB 10100 = 00100).

When placed between two strings, the plus sign (+) indicates that the strings should be concatenated. For example,

```
CITYSTATE = "Knoxville " +
 "Tennessee"
```

will assign the variable CITYSTATE to the string "Knoxville Tennessee".

WINCMD STATEMENTS A list of the statements and functions in the WINCMD language is shown in Figure 1. To cause WINCMD to launch a program or to execute a DOS command, simply include that program or command on a separate line. For example, to start the Windows Notepad editor, simply type the name of the program on a line, as in

```
NOTEPAD
```

If no extension for the program to b launched is specified, WINCMD wi look for each of the extensions listed i the Programs key in the [Windows] section of WIN.INI until the particular program is found. If no path is specifie WINCMD follows the same search procedure Windows uses when starting a program from the Program Manage What this means is that WINCMD firs looks in the current directory, followe by the Windows and Windows Syste directories, then the WINCMD directory, the directories listed in the PATH statement, and finally any directories tha are mapped on a network.

Executing DOS internal commands i as easy as launching programs. To cop the file WINCMD.C to WINCMD.BAK just include the command on the line a you would if you were typing it in fro the DOS command line:

■ Utilities by Modem

The *PC Magazine* utilities are available by modem from PC MagNet, a ZiffNet service hosted by CompuServe.

To find the phone number nearest you, set your communications software to 300, 1,200, 2,400, or 9,600 bits per second, 7 data bits, even parity, 1 stop bit, and full duplex, then dial 800-346-3247 with your modem. When the modem connects, press Enter. At the HOST NAME prompt, enter PHONES. Follow the menus and note the number closest to you. Or you can call 800-635-6225 (voice) and follow the instructions and note the number.

To obtain the current issue's utility free of charge: Dial the local number; at the HOST NAME prompt, type CIS; and at the USER ID prompt, enter 60116,1. Then at the PASSWORD prompt, enter PCMAG UTIL.

To join ZiffNet: At the USER ID prompt, type 177000,5000. Then, at the PASSWORD prompt, enter PC*MAGNET. Finally, at the ENTER AGREEMENT NUMBER prompt, type PCMAG93.

Register your name and enter your American Express, MasterCard, or Visa number. (If you'd like to have your company billed instead, call CompuServe at 800-848-8990.) Your personal user ID and a temporary password will be displayed. A new password will arrive in the mail within ten days to confirm your subscription.

ZiffNet membership costs $2.50 per month. This includes access to *PC Magazine* Editors' Choice awards, Product Reviews Index, Weekly News from *PC Week*, Buyers' Market, ZiffNet Highlights, and the Support Forum (which also includes the current utility). CompuServe members can join by entering GO PCMAG at any CompuServe ! prompt.

Outside of these areas, PC MagNet costs $6.30 per hour for 300-bps service; $12.80 for 1,200 or 2,400 bps; and $22.80 for 9,600 bps. Billing is based on 1-minute increments.

These programs can be copied but are copyrighted. You may make copies for others as long as no charge is involved, but making copies for any commercial purpose is strictly prohibited.

WinCmd MsgBox Functions

Value	Produces
0x0000	OK Button
0x0001	OK and Cancel Buttons
0x0002	Abort, retry and Ignore Buttons
0x0003	Yes, No, Cancel Buttons
0x0004	Yes and No Buttons
0x0005	Retry and Cancel Buttons
0x0010	Hand Icon
0x0020	Question mark Icon
0x0030	Exclamation Icon
0x0040	Asterisk Icon
0x0000	Sets default button to button 1
0x0100	Sets default button to button 2
0x0200	Sets default button to button 3
0x1000	Creates System modal message box
0x2000	Creates Application modal message box

Return Value	Meaning
1	OK button pressed
2	Cancel button pressed
3	Abort button pressed
4	Retry button pressed
5	Ignore button pressed
6	Yes button pressed
7	No button pressed

***Figure 2:* These are the values that can be passed to the MsgBox function and the meaning of the values returned.**

```
COPY WINCMD.C WINCMD.BAK
```

When WINCMD sees a DOS internal command, such as COPY or DEL, it launches the command processor listed in the COMSPEC statement in the WINCMD environment.

If Windows is running in Enhanced mode and your _DEFAULT.PIF file is configured to run DOS programs in a window, DOS internal commands will be executed in the background. Since the WINCMD language uses some of the same special characters—such as the asterisk (*)—as the DOS command line does, you will need to enclose any filenames in quotation marks. For example, to copy all files with a .C extension to .BAK, just enclose the filenames in quotation marks as follows.

```
COPY "*.C D:\BACKUP"
```

One major difference between the action of a WINCMD file and a DOS batch file is that WINCMD programs continue to execute after a program has been launched. In DOS, if a batch file starts a program, it waits until that program terminates before continuing. In the example that follows, however, a WINCMD program launches three programs: Notepad, Word for Windows, and WordPerfect for DOS.

```
NOTEPAD
WINWORD
WP
```

Since WINCMD programs do not wait for a program to finish before continuing to the next line, all three programs will start at approximately the same time. WINCMD programs continue until there are no more lines in a program or an EXIT statement is encountered.

The EXIT statement is used to stop execution of a WINCMD program. In the following example, therefore, Notepad and WordPerfect for DOS will be executed, but Word for Windows will not, since its line comes after the EXIT statement.

```
NOTEPAD
WP
EXIT
WINWORD
```

Comments can be included in WINCMD files by using the REM statement. WINCMD ignores any characters from the REM statement to the end of the line. To provide a syntax similar to that of BASIC and C, comments can also be indicated by the single quote (') or by double forward slashes (//):

```
rem This is a comment line in WINCMD
// This is too.
'  Don't forget me!
```

While I will later discuss a more elegant way for WINCMD programs to ta
to the user, the simplest way is to use th
SAY and PRINT statements. (SAY an
PRINT perform the same functio
While the SAY/PRINT function is sim-
lar in the function of the SAY stateme
in REXX, I have included the PRIN
alias for all those users familiar wi
BASIC.

The SAY statement displays a lin
that appears in the output windo
of the WINCMD.EXE window. I
WINCMD.EXE is minimized when a
SAY statement is encountered, the win-

WinCmd Key Aliases

Key	Alias
BACKSPACE	{bksp}
BREAK	{break}
CAPS LOCK	{capslock}
CLEAR	{clear}
DEL	{delete}
DOWN ARROW	{down}
END	{end}
ENTER	{enter}
ESC	{esc}
HELP	{help}
HOME	{home}
INS	{insert}
LEFT ARROW	{left}
NUM LOCK	{numlock}
PAGE DOWN	{pgdn}
PAGE UP	{pgup}
PRINT SCREEN	{prtsc}
RIGHT ARROW	{right}
TAB	{tab}
UP ARROW	{up}
F1	{F1}
F2	{F2}
F3	{F3}
F4	{F4}
F5	{F5}
F6	{F6}
F7	{F7}
F8	{F8}
F9	{F9}
F10	{F10}
F11	{F11}
F12	{F12}
To shift a key precede the key with one of the fo- lowing three characters.	
SHIFT	+ (plus sign)
ALT	% (percent sign)
CTRL	^ (caret)

***Figure 3:* The aliases above are used by the SENDKEYS() function to send "special key" keystrokes to applications.**

SEARCH.WCM

Complete Listing

```
//---------------------------------------------------------
// WinCmd program that uses file manager to
// search a disk for a set of files.
//---------------------------------------------------------
//
// Ask user for file spec.  Default to all files
//
ans = AskBox ("Enter the file name to look for", "*.*")
//
// If user pressed the Cancel button, quit
//
if ans+a == a
   exit
//
// If the file manager isn't running, start it
//
if (AppActivate ("File Manager") = 0)
   winfile
//
// Send keys to File Manager to search disk.
//
sendkeys ("%-n")          //Minimize directory tree window
sendkeys ("%fh")          //Open file search dialog
sendkeys (ans)            //Enter file spec
sendkeys ("{tab}")        //Tab to directory window
sendkeys ("c:\")          //Enter hard drive root
sendkeys ("{enter}")      //Start search
sendkeys ("%-x")          //Maximize search results window
exit
```

PC MAGNET

Figure 4: **SEARCH.WCM is a WinCmd program that uses the File Manager to search for files on a disk.**

dow will be restored so that the user can read the text of the message. The SAY statement first evaluates whatever is on the line beyond the SAY statement, and then it displays the results in the output window. The following two statements will provide an example of SAY and PRINT:

```
SAY I think that five times 3 is 5 *
 3 "and that six divided by two
 is " 6/2
PRINT I think that five times 3 is 5
 *3 "and that six divided by two
 is " 6/2
```

When either of the above two lines is executed, the line:

```
I THINK THAT FIVE TIMES 3 IS 15
 and that six divided by two is 3
```

will appear. Notice that the words outside of the quotation marks are capitalized, but that the string enclosed in quotation marks is lowercase. Unlike BASIC's PRINT statement, the different expressions do not have to be separated by commas or semicolons.

Branching is performed with the IF statement. The IF statement will execute the next line if the condition included in the IF statement is true (or non-zero). While not required, the THEN statement can be used either on the same line as the IF or on the next line. If the optional ELSE keyword is used and the expression evaluates to zero, the first statement beyond the ELSE keyword is executed. The syntax for the IF statement is as follows:

```
IF {expression}
[THEN]
   {statement}
[ELSE
   {statement}]
```

The square and curly brackets in the diagram above are not part of the syntax—you don't enter them in a WINCMD program. The IF, THEN, and ELSE statements are capitalized and the {statements} are indented only for readability. The {expression} is to be replaced with any variable or equation, and {statement} represents any valid WINCMD line.

As a concrete example of the IF statement at work, consider the following:

```
IF BOB > 5
THEN
   BOB = BOB + 2
```

Here, if the variable BOB is greater than 5, BOB will be set to its current value plus 2. If you added an ELSE statement, the example might become

```
IF BOB > 5
THEN
   BOB = BOB + 2
ELSE
   BOB = BOB + 1
```

In this case, if the variable BOB is greater than 5, BOB will be increased by 2; otherwise BOB will be increased by 1. Since the THEN statement is optional, this could equivalently have been written as

```
IF BOB > 5
   BOB = BOB + 2
ELSE
   BOB = BOB + 1
```

Note that the IF statement will only execute one statement line for a THEN or ELSE clause. To get around this, you can use the DO and END statements, which permit grouping multiple statements as if written on a single line. DO...END blocks of statements can be included in both the THEN and ELSE clauses of the IF statement. The DO and END statement syntax is as follows:

```
DO
   {statement}
   {statement}
        .
        .
        .
   {statement}
END
```

As an example of using DO and END to include more than the one statement within the IF-THEN-ELSE structure, consider

```
IF BOB > 5 AND DAY = "Monday"
   DO
      BOB = BOB + 2
      FRED = BOB - 2
      JOE = BOB + FRED
   END
```

```
ELSE
   DO
      COPY WINCMD.C WINCMD.BAK
      JOE = BOB/2 + 2
    END
```

Here, DO END statements enclose the three statements of the THEN clause and the two statements of the ELSE clause. Note also that a more complex condition is used. In this example, the THEN clause will be executed only if BOB is greater than 5 *and* DAY equals the string "Monday."

Loops are performed by the WHILE statement, whose syntax is

```
WHILE {condition}
   {statement}
```

The WHILE statement evaluates the condition and, if the condition is non-zero (that is, if it is TRUE), the line indicated by {statement} is executed. The WHILE statement then re-tests the condition and, if the result is still non-zero, it executes the statement again. WHILE will thus continue to test the condition and execute the statement until the condition evaluates to zero. Like the IF statement, the WHILE statement works only on the statement line that immediately follows the WHILE, but this statement line can be the start of a DO...END block. Thus, multiple statements can be executed in the loop.

The example below sets the variable COUNT to 10, and then enters a WHILE loop that decrements COUNT each time through until COUNT equals zero.

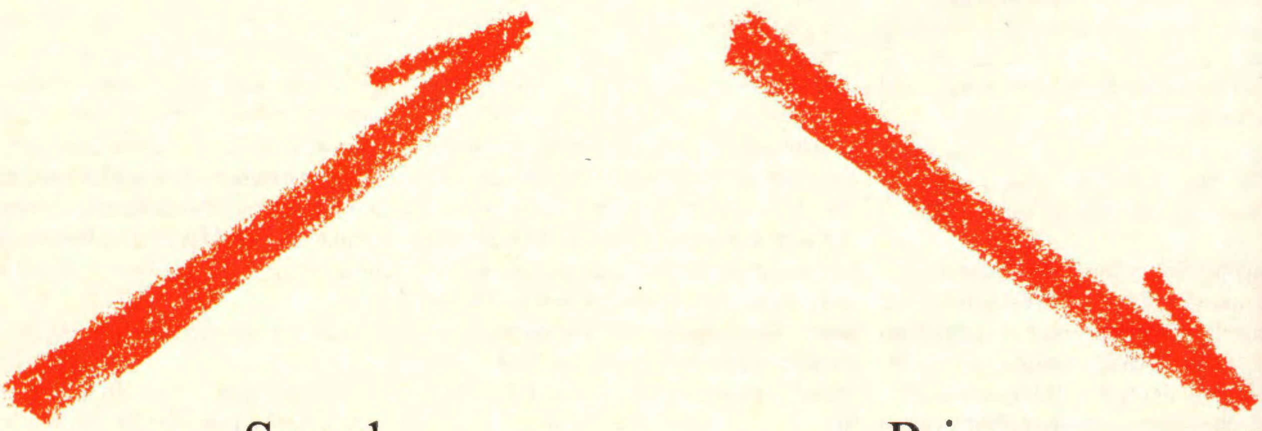
```
COUNT = 1Ø
WHILE COUNT > Ø
DO
     COUNT = COUNT - 1
     FRED = FRED * FRED
END
```

If you want to leave a WHILE loop before its condition evaluates to false, you can use the LEAVE statement. When WINCMD.EXE sees a LEAVE statement, the loop is immediately exited, even if there are other statements after it. The next statement to execut will be the first one beyond the loop. I the next example, the loop terminates i the variable ERROR becomes non-zer

```
WHILE FRED <> BOB
DO
  BOB = BOB + 1

  REM
  REM INSERT OTHER LINES HERE
  REM

  IF ERROR
       LEAVE

   REM
   REM ANY STATEMENTS AFTER THE
   REM LEAVE WILL NOT BE EXECUTED IF
   REM ERROR NOT Ø
   REM
END
```

SUBROUTINES AND FUNCTIONS Th WINCMD language allows you to crea subroutines and functions. Actual WINCMD makes no distinction betw

Introducing three new
You'll find the



Speed

Price

functions and subroutines. If a user-defined routine is called while evaluating an expression, the routine is treated as a function and the value returned by the routine replaces the function name. If the user-defined routine is called on a separate line, it is treated as a subroutine and any value returned by the routine is simply ignored.

The start of a subroutine is defined by a *label,* which is quite simply the name of the subroutine. A label must be a valid variable name; that is, it must start with a letter or underscore and it must be less than 253 characters in length. The last character of a label must be a colon (:). A subroutine is terminated by a RETURN statement. There is no limit to the number of statements that can be used in a subroutine.

In the following example, the subroutine WASTETIME is called to waste a little time in the program.

```
BOB = BOB + 1
WASTETIME
BOB = BOB + 4
EXIT

WASTETIME:
    COUNT = 1000
    WHILE COUNT > 0
        COUNT = COUNT - 1
RETURN
```

In this example the variable BOB is incremented, after which the subroutine is called. The loop is executed in the subroutine and the subroutine returns, upon which BOB is increased by 4. The EXIT statement then prevents the program from continuing down through the WASTETIME subroutine again. You must always EXIT a subroutine.

You can pass parameters to subroutines by enclosing the parameters in parentheses after the variable name. Multiple parameters are separated by commas. Below, I've modified the former example to pass the number 1000 and the variable BOB to the WASTETIME routine.

```
BOB = BOB + 1
WASTETIME (1000, BOB)
BOB = BOB + 4
EXIT

WASTETIME:
    COUNT = ARG(1)
    WHILE COUNT > ARG (2)
        COUNT = COUNT - 1
RETURN
```

Subroutines retrieve their arguments by using the built-in ARG() function. ARG() returns the subroutine parameter indicated by the number passed to it; that is, ARG(1) returns the first parameter, ARG(2) the second, and so on. If a subroutine attempts to read a parameter that was not passed, ARG() returns a null string. If ARG() is called with no arguments, it returns the number of parameters passed to the subroutine. ARG (0) returns the name of the routine that was called.

If the ARG() function is called outside a subroutine or function, it returns the command line arguments to the program. Just as the %1 and %2 return the first and second arguments in a DOS

Parallel Tasking adapters.
lines familiar.

Reaction

When we introduced EtherLink® III, the world's fastest Ethernet adapter, your response was equally quick: You bought it in record numbers.

Now EtherLink III's unique Parallel Tasking™ technology includes a 32-bit EISA and 16/32-bit Micro Channel® for twisted-pair, thin, or thick coax, and a 16-bit ISA combo with all three media on the same board.

You get the same pipelining design, the same software drivers, and the same automatic self-configuration that made EtherLink III a runaway success. Plus, complete SNMP manageability with LinkWatch.™ And like all 3Com® adapters, the new EtherLink III family comes with a lifetime warranty.

To learn more about this remarkable family of Ethernet adapters, and its surprisingly low price, call **1-800-NET-3Com.** Or see your nearest 3Com reseller. Because now there's even more to smile about.

3Com®

Networks That Go the Distance™

CIRCLE 293 ON READER SERVICE CARD

batch file, ARG(1) and ARG(2) will return those parameters for a WINCMD file. ARG(0) returns the name of the WINCMD program, and ARG() returns the number of parameters on the command line.

In the example below, the user-defined function MAX returns the value of the largest argument passed to it:

```
BOB = MAX (300, FRED, SAM, JOE) * 3
EXIT

MAX:
COUNT = ARG()
MAXVAL = ARG(1)
WHILE COUNT > 0
DO
    IF  MAXVAL < ARG(COUNT)
        MAXVAL = ARG (COUNT)
END
RETURN MAXVAL
```

In the code above, MAX uses ARG() to determine how many arguments have been passed to it. It then compares each argument and returns the value of MAXVAL on the RETURN line. When MAX has finished, the variable BOB will be assigned the highest value of all the arguments passed, multiplied by 3.

WINCMD variables are global: If a variable is modified in a subroutine, the change will also be reflected in the variable when the subroutine is finished. WINCMD does not support any form of local variables, nor does it currently let you get rid of a variable that has been defined.

WINCMD BUILT-IN FUNCTIONS WINCMD has a number of built-in functions that make it easier to create useful programs. These functions can be divided into two types: general programming functions and functions designed to control Windows applications. The LENGTH() function returns the number of characters in a string. For example, in the line

```
BOB = LENGTH ("HolstonHills")
```

BOB would be assigned 13.

The SUBSTR() function, which is similar to the like-named functions in BASIC and REXX, returns a portion of a string. SUBSTR() is called with three parameters, which determine the portion of the string returned. The first argument is the original string; the second argument is the starting character; and the last argument is the number of characters to return. In the following example, the variable FRED is assigned the string "bcdef", which represents the second through the seventh characters of the string "abcdefghi."

```
FRED = SUBSTR ("abcdefghi", 2, 5)
```

The UCASE() function returns an uppercase version of the string passed as its parameter. The string passed is not changed. In the next example, BOB is assigned "BCDEF" while FRED remains "bcdef".

```
FRED = "BCDEF"
BOB = UCASE (FRED)
```

The DELAY() function holds up the execution of the next line in the program for a specified number of milliseconds. While the delay time can range from 1 to 65,355 ms, in practice numbers less than 100 are not useful. Using the DELAY() function is preferrable to simple looping as a way of killing time, because during a DELAY() functio WINCMD.EXE is idle and does not tak any processor time away from other applications.

The MSGBOX() function lets a WINCMD program display a standar Windows Message Box. The functio takes three parameters: the text of th message box; the title of the message bo and a configuration parameter that tel Windows which buttons and icon to display in the box. The usable configuratio parameters are shown in Figure 2. Th various values listed can be combined b using the ORB operator. Th MSGBOX() function also returns a val that depends on which button you pres Figure 2 also shows what these returne values mean.

The following example displays a ve disturbing message box.

```
REPLY = MsgBox ("Your disk is
  trashed", "Disk Error", 2 ORB 0x30
  ORB 0x1000)
```

This message box displays "Your disk trashed" in a message box titled "Di Error," together with an Exclamati icon. The box will have three butto Abort, Retry, and Ignore, and will be sy

▪ PC Magazine Utilities Updates

As with all good software, the programs presented in *PC Magazine* are upgraded and improved. Here's a partial list of the programs on PC MagNet that have been updated. To download these files from the Utilities/Tips Forum, type GO ZNT:TIPS, type LIB or select Libraries from the menu, then select Library 2 PCMAG UTILS. Type DOW and the filename listed below (for example, DOW ANSI.COM), or select Download a File from the menu.

ADDIT.COM, Version 1.1
ANSI.COM, Version 1.3
BAT2EXEC.COM, Version 1.5
BCOPY, Version 1.2
CARDFILE.COM, Version 1.1
CHKFRAG.EXE, Version 1.7
CONFIG.CTL, Version 3.0
DIRMATCH.COM, Version 3.1
EMMA.COM, Version 2.2
EMS40.SYS, Version 1.1
PCACCESS.EXE, Version 1.1
RN.COM, Version 3.0
SLICE.COM, Version 1.3
SNIPPER.COM, Version 1.2
ZCOPY.COM, Version 1.4

For a list of the programs that are available on PC MagNet, download the file PCMCAT.ZIP from Library (General Info) of the Utilities/Tips Forum.

A downloadable index to *PC Magazine*'s product reviews is also available in Library 1 (General Info). PCM.EXE is a self-extracting file containing the Computer Library *PC Magazine* Reviews Index for Januar 1988 through December 1991. It requires the search files that are in PCSRCH.EXE.

tem-modal. The value in the variable REPLY depends on which button the user presses.

The ASKBOX() function displays a dialog box with a message and an entry field. This function allows WINCMD programs to ask the user for input. The two parameters to the function are the question to ask the user and the default answer for the entry field. ASKBOX will display a dialog box with OK and Cancel buttons. If the user presses OK, the function returns the text in the entry field. If the user presses the Cancel button, the function returns a null string.

The GETAPPACTIVE() function returns the text of the active window on the Windows desktop. (This is the window on the desktop that has the active colored title bar.) The text returned is the text from the title bar of this window. For example, if Notepad has just been started and has the input focus, the text returned by GETAPPACTIVE() will be "Notepad - (Untitled)." The GETAPPACTIVE() function has no input paramters.

The APPACTIVATE() function activates the window whose title bar text matches that supplied as the single parameter to the function. For example, to set the input focus to the Program Manager, the function would look like this:

```
APPACTIVATE("Program Manager")
```

The function returns the text of the previously active window if the function was successful, otherwise it returns 0.

In using APPACTIVATE you must bear in mind that many programs add things to their window titles. Notepad, for example, adds the current filename; Program Manager adds the name of a program group that's maximized, and so on. Unfortunately, APPACTIVATE can't recognize these expanded titles.

The SENDKEYS() function provides a way for WINCMD programs to control Windows applications by sending keystrokes to the application that has the current input focus. SENDKEYS takes only one string argument, but that string can simulate every key on the keyboard, including the Ctrl, Alt, and Shift keys.

The format of the SENDKEYS() input string is similar to the strings used by the SENDKEYS functions in WordBASIC and Visual BASIC. Unlike WordBASIC and Visual BASIC, however, in a WINCMD program you set the active application *before* sending the keys, not after. Most keys can be sent to an application simply by including the letter in the input string. For example, to send the three keys `a`, `b`, and `c` to the Program Manager, just use the following two lines in your WINCMD program:

```
APPACTIVATE("Program Manager")
SENDKEYS ("abc")
```

To send a shifted key, simply precede the letter with ^, %, or + character to simulate the Ctrl, Alt, or Shift key, respectively. The shifting prefixes can be combined, so you can create combinations such as Ctrl-Shift-Alt-W.

If you want to hold the shifting key down for a number of keys, enclose the keys in parentheses. For example, to send the `f` and `x` keys while holding down the Alt key, use the command

```
SENDKEYS ("%(fx)")
```

Special keys, such as Enter, Escape, and Tab, can be sent using an *alias* for the key, enclosed in brackets {}. A list of the recognized key aliases is shown in Figure 3. For example, to send Enter and Tab keys to an application, use the command

```
SENDKEYS({enter}{tab})
```

To send the such keys as +, ^, and %, enclose the key in brackets thus:

```
SENDKEYS("{+}")
```

Note that I have enclosed most of the SENDKEY strings in quotes. This is because many of the characters in a SENDKEY string may be characters that WINCMD would otherwise interpet for its own use. The plus key and the parenthesis keys are examples. To avoid having WINCMD do something you do not want, simply enclose the string in quotes. Since there is no harm in using the quotes, I recommend you use them consistently.

2CLIP.WCM

Complete Listing

```
//---------------------------------------------------
// WinCmd program that copies a text file into the
// Windows clipboard
//---------------------------------------------------
//
// Ask user for file name.  Default to blank
//
ans = AskBox ("Enter the name of the file to copy into the clipboard"," ")
//
// If user pressed the Cancel button, quit
//
if ans+a == a
   exit
//
// Start NotePad
//
notepad

//
// Send keys to notepad to read in the file
//
sendkeys ("%fo")          //Display the file open dialog box
sendkeys (ans)            //Enter file spec
sendkeys ("{enter}")      //Open the file.
```

```
//
// See if notepad read the file.  If it didn't, the warning box
// for the "Open" dialog will be active, and GetAppActive() will
// return "Open".  In that case, close notepad and tell the user
//
winact = Getappactive()
if (winact = "Open") do
   sendkeys ("{enter}")  //Close the warning box
   sendkeys ("{esc}")    //Close the file open box
   ok = fail
end
else do
   sendkeys ("%ea")      //Select all the text
   sendkeys ("%ec")      //Copy the data into the clipboard
   ok = pass
end
sendkeys ("%fx")         //Close notepad
if (ok = fail)
  msgbox ("Notepad could not access the file you requested.
     Nothing was copied into the clipboard", "Error!", Ø)

exit
```

***Figure 5:* 2CLIP.WCM uses the NOTEPAD to copy a file onto the Windows Clipboard.**

One additional caution is that SENDKEYS() works by placing simulated keystrokes in Windows' system event queue. Since this queue can become filled if you send too many characters at once, it's a good idea to break up the keys that you send into blocks of fewer than five keys. Since WINCMD.EXE yields to other applications between each line of a WINCMD program, sending keys on consecutive lines will not present a problem.

The GETAPPEXE() function returns the name of the program that created the window with the matching title text. GETAPPEXE() is handy for determining whether a particular program is running. For example,

```
GETAPPEXE("Program Manager")
```

will return PROGMAN.EXE. Note again, however, that the match to the window title text must be exact.

The final WINCMD function, TICKS() returns the number of milliseconds since Windows was started. TICKS() takes no arguments.

WINCMD EXAMPLES The power of WINCMD is best demonstrated by example. SEARCH.WCM, the first of the two sample programs I've included with this utility, uses the File Manager to search a disk for matching files. The WINCMD program is listed in Figure 4. The second program, 2CLIP.WCM, uses Notepad to copy a file into the Windows Clipboard. Both programs use features of the standard Accessories bundled with Windows to help perform the function of the WINCMD program. The programs also demonstrate the power of the SENDKEYS function in controlling other programs.

WINCMD.EXE allows you to write command files with which you can control your favorite Windows applications. However, WINCMD.EXE is not the last that you will hear about the WINCMD language. In the next installment of this column, I will present a DLL that hooks into WINCMD.EXE to provide dozens of helpful functions for your WINCMD programs, so be sure to stay tuned! □

DOUGLAS BOLING IS A CONTRIBUTING EDITOR TO *PC MAGAZINE*.