WINIMAN Version 1.10

A Windows INI File Manager

1. Introduction

MS-DOS(2) 6.0 gives users the ability to create multiple configurations of CONFIG.SYS and AUTOEXEC.BAT. While this is very handy, one problem is that Windows(2) does not give this same flexibility.

WINIMAN(1) gives you the ability to change your WIN.INI & SYSTEM.INI files on the fly based on any DOS environment variable including your boot-up configuration variable (config) from MS-DOS if using version 6.0 or later. You can use **WINIMAN** even if you are not using MS-DOS 6.0 as it only depends on DOS environment variables and not the new multiple configuration feature. It will work on any file that interprets a semi-colon as first character of a comment statement. (Let's hope that a future release of IBM's, Microsoft's or Novell's DOS supports comments beginning with a semi-colon in batch files!).

This program does not keep multiple copies of your WIN.INI and SYSTEM.INI files, but rather modifies your existing INI files directly (by taking a copy which is renamed to the original name if successfully processed).

This program is designed to be run before you run windows - for example in your AUTOEXEC.BAT.

2. Command Invocation

WINIMAN is driven by a control file containing the names of the files you require it to process. The default name of the control file is WINIMAN.DAT in the current directory. This may be overridden on the command line as shown below. The file names specified in the control file are not case sensitive but must begin in column 1 (left justified).

WINIMAN {control file name} {/q}

For example, WINIMAN.DAT control file might contain 3 lines:-

C:\Windows\System.Ini
C:\Windows\Win.Ini
C:\Winapps\Desk\BackMenu.Ini

Registered users can specify the '/q' switch for quiet mode to prevent display of status messages. This switch can be specified before or after the optional control file name.

3. **WINIMAN** Processing Statements Syntax

In the following **WINIMAN** statements, these definitions apply:-

variable - an environment variable defined by the DOS SET command (including the DOS 6.0 'config' variable if multiple configuration feature is used). **See Note (iii) below**.

<op> - the logical operator for the comparison (**case insensitive**) :-
either
'==' or 'EQ' **(**meaning "equal to")
or
'!=' , '<>' or 'NE' (meaning "not equal to")

value - the value that the variable is compared against (**case sensitive**). The special value of **NULL** is used to test if the variable has **not** been set.

activated - leading semi-colon (if present) removed so that the statement is active in the INI file.

deactivated - leading semi-colon (if not already present) added so that the statement is treated as a comment in the INI file.

The syntax of the **WINIMAN** processing statements in the INI files is one of the following:-

a. **WINIMAN** SINGLE IF statement syntax:-

```
;{%variable%<op>'value'}
```

The statement following this is activated if the test succeeds and deactivated if the test fails.

The SINGLE IF statement form only affects the line following which **must not** be another **WINIMAN** processing statement.

b. **WINIMAN** BLOCK IF statement syntax:-

```
;{%variable%<op>'value'}start
.
.
.
;{%variable%}end
```

All statements within the defined block are activated if the test succeeds and deactivated if the test fails.

c. **WINIMAN** TEST CASES statement syntax:-

```
;{test(%variable%)}start
;{case('value')}
.
.
;{case('value')}
.
.               .
;{otherwise}
.
.
;{test(%variable%)}end
```

All statements following the CASE statement that specifies the current value of the variable in the TEST statement up to the next CASE statement, OTHERWISE statement or TEST END statement are activated. All other statements (ie following the other CASE statements) are deactivated.

The statements following the OTHERWISE statement are activated if no previous CASE statements were found to be true and deactivated if one was. Only **one** OTHERWISE statement may appear in a TEST CASES block and it must be the last **WINIMAN** statement before the TEST END statement.

TEST CASE example in the [boot] section of SYSTEM.INI:-

```
;{test(%Name%)}start
;{case('David')}
;;  David prefers BackMenu as his shell
shell=Backmenu.Exe

;{case('John')}
;;  John prefers Norton Desktop for Windows as his shell
shell=NDW.Exe

;{otherwise}
;;  Others will have the standard Program Manager
shell=Progman.Exe
```

;{test(%Name%)}end

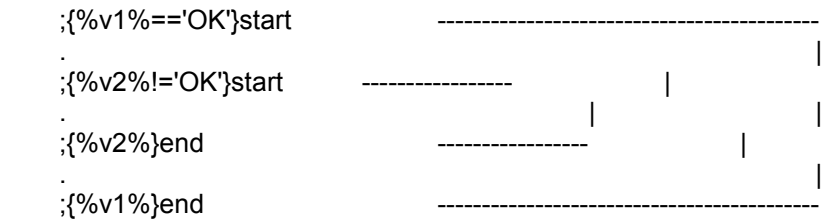
Other options that could be controlled in this manner are the restrictions you can place on the default Program Manager File Menu. These are controlled by settings under the [Restrictions] heading in PROGMAN.INI. For example, you may want to stop everyone else renaming or deleting Program Manager Groups and Items when sharing a PC by using the EditLevel=3 option.

The meanings of these features are as follows (all supplied defaults are 0 - meaning false):-

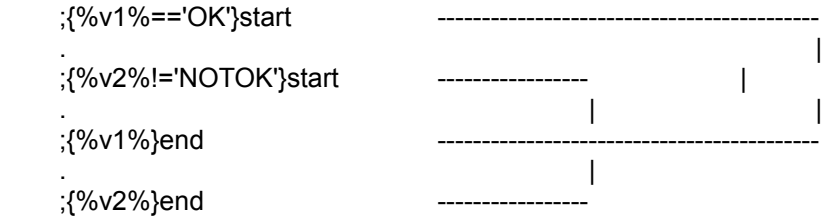
[Restrictions]
NoRun=0 or 1 (0 allows the Run entry on the File Menu)
NoClose=0 or 1 (0 allows Windows to be closed/exited)
NoSaveSettings=0 or 1 (0 allows changes to Program Manager to be saved)
NoFileMenu=0 or 1 (0 allows access to Program Manager File Menu)
EditLevel=0, 1, 2, 3 or 4

0: Lets the user make any modifications to the Program Manager. These can be saved if NoSaveSettings=0
1: Prevents the user from renaming, deleting or creating Program Manager Groups
2: As Level 1 plus the user cannot remove or create program items
3: As Level 2 plus the user cannot change the command line for any program
4: As Level 3 plus the user cannot change any information for existing program items.

Any number of **WINIMAN** blocks may be nested inside another **WINIMAN** block and any number of **WINIMAN** single statements can be placed inside a block (with the above provisos). All nested blocks must be wholly contained within the next outer block. For example:-



is valid, whilst:-



is not.

4. MS-DOS 6.0 Configuration Example

CONFIG.SYS

```
[MENU]
menuitem=SAlone, Configure for Stand Alone Laptop
menuitem=Docked, Configure for Docking Station
.
.
.
```

AUTOEXEC.BAT

```
.
goto %config%
.
```

This above example gives you 2 ways to boot your system (say a Laptop or NoteBook PC that can be stand alone or in the docking station at your desk). Your CONFIG.SYS & AUTOEXEC.BAT can process this at runtime. You have a Sound Card in your docking station and you only need the corresponding driver loaded in the [386enh] section of SYSTEM.INI when you are docked. For sake of argument, let's also say that the card cannot be enabled when not installed. By adding the **WINIMAN** SINGLE IF statement one line above where it is needed in the WIN.INI or SYSTEM.INI we can allow **WINIMAN** to configure these options on the fly:

```
[386enh]
device=anydriver.386
;{%config%=='Docked'}
device=sound.386
.
.
.
```

What **WINIMAN** will do, if the user picks any menu option other than Docked, the line "device=sound.386" will be changed to ";device=sound.386" but if the user picks Docked, **WINIMAN** will make sure that there is no semi-colon in front of the option.

This type of processing may also be important when drivers are added for other devices (eg tape drives). Some programs (for example Norton's pcAnywhere) add their own screen drivers to only send updates to the remote PC. These may conflict with certain graphics cards and should not be loaded for normal use.

It is important that you put the **WINIMAN** statements inside curly braces {} NOT square brackets [].

5. SETVAR(1) Program

This companion program prompts input from the user and creates SETVAR.BAT in the directory specified by the DOS environment variable TEMP or in the current directory if this is variable has not been set. If this file already exists, it will be overwritten.

This batch file sets 2 DOS variables SETVAR1 containing the whole line entered and SETVAR2 containing the first word of the input line.

SETVAR.BAT should then be executed using the DOS CALL command and can then be deleted. Don't forget to free the DOS SETVAR1 & SETVAR2 environment variables after use.

SETVAR can take an optional switch /u which will cause the variable values to be converted to upper case.

For example, your AUTOEXEC.BAT might contain:-

```
@Echo off
.
Echo Please enter your name:-
SETVAR
IF "%TEMP%" == "" Goto No_Temp_Var

Call %TEMP%\SETVAR.BAT
Del %TEMP%\SETVAR.BAT
Set Name=%SETVAR2%
Goto Clear_SETVAR_Variables

:No_Temp_Var
Call SETVAR.BAT
Del SETVAR.BAT
Set Name=%SETVAR2%

:Clear_SETVAR_Variables
Set SETVAR1=
Set SETVAR2=
```

```
:Process_Winiman
Winiman
```

6. Notes

i. All **WINIMAN** statements within a block in which the comparison failed will not be processed (ie they are also treated as comments).

ii. Comments within a block must begin with 2 semi-colons ';;' in order not to be processed by **WINIMAN**.

iii. **Both the variable name and the value are case sensitive.** The DOS SET command capitalises the variable name it sets but **not** the value it gives it. Environment variable names set by programs **are case sensitive**. In order to provide some consistency, **WINIMAN** will first try to find a match for the variable **EXACTLY** as specified on the statement. If it cannot find it, it will then search for a match on the upper case version of the variable name. If it still cannot find the variable, **the variable is assumed to have the special value of 'NULL'**. This can be used to provide a 'default setting' feature. If you appear to have difficulty, use the DOS SET command without any arguments to display the variables set and their values.

iv. The DOS environment variable and the value may not contain the special characters used by WINIMAN - semi-colon, curly braces, round brackets, per-cent sign and single quotation mark - ;{}()%'.

7. Disclaimer

The **WINIMAN** and **SETVAR** are provided AS IS. Neither David Kelvin nor Absolute Zero Software Services will in any way be responsible, in financial or any other terms, for damages (both consequential and incidental) resulting from the use or misuse of the **WINIMAN** and **SETVAR** programs.

As a precaution, it is suggested you backup all the files specified in the **WINIMAN** control file.

8. Registration

When you register **WINIMAN**, you will receive a serial number and a Registration Code. In order to get **WINIMAN** to recognize this and allow you to use the '/q' switch and suppress the title, Copyright and Shareware/Registered banner, you must create the file WINIMAN.INI in the same directory as WINIMAN.EXE. This file is in the same format as all Windows INI files and should contain the following 4 lines:-

```
[WINIMAN]
RegistrationName=<The EXACT name as requested on the Registration form>
RegistrationCode=<The returned Registration Code>
SerialNumber=<The returned Serial Number of your copy of WINIMAN>
```

For example:-

```
[WINIMAN]
RegistrationName=John Smith
RegistrationCode=1234567890
SerialNumber=123456
```

Installation consists of 5 files:-

README.TXT	Size: 510	Creation Date & Time: 13/10/93 01:10:00
REGISTER.WRI	Size: 6016	Creation Date & Time: 13/10/93 01:10:00
SETVAR.EXE	Size: 10376	Creation Date & Time: 13/10/93 01:10:00
WINIMAN.EXE	Size: 34768	Creation Date & Time: 13/10/93 01:10:00
WINIMAN.WRI	Size: 23424	Creation Date & Time: 13/10/93 01:10:00

9. Version History

All versions prior to 1.10 were never formally released.

Version	Comments
1.10	Initial release

10. Acknowledgements

Thanks go to the author of WIN6UTIL[3] for reminding me of the need for this type of utility. Unfortunately, good as WIN6UTIL is, it did not provide all that I needed and it's statements confused the other programs I used which did not understand lines beginning with open curly braces in their INI files!

Absolute Zero Software Services, PO Box 260, Edgware, Middx, ENGLAND, HA8 8EY