**FileExists**

The ${FileExists} condition included in LogicLib will evaluate to true if what exists is actually a directory. I suggest you add the following macros to your script (right after including LogicLib.nsh):

*;FileExists is already part of LogicLib, but returns true for directories as well as files*

**!macro** \_FileExists2 \_a \_b \_t \_f

**!insertmacro** \_LOGICLIB\_TEMP

StrCpy $\_LOGICLIB\_TEMP "0"

StrCmp `${\_b}` `` +4 0 *;if path is not blank, continue to next check*

IfFileExists `${\_b}` `0` +3 *;if path exists, continue to next check (IfFileExists returns true if this is a directory)*

IfFileExists `${\_b}\\*.\*` +2 0 *;if path is not a directory, continue to confirm exists*

StrCpy $\_LOGICLIB\_TEMP "1" *;file exists*

*;now we have a definitive value - the file exists or it does not*

StrCmp $\_LOGICLIB\_TEMP "1" `${\_t}` `${\_f}`

**!macroend**

**!undef** FileExists

**!define** FileExists `"" FileExists2`

**!macro** \_DirExists \_a \_b \_t \_f

**!insertmacro** \_LOGICLIB\_TEMP

StrCpy $\_LOGICLIB\_TEMP "0"

StrCmp `${\_b}` `` +3 0 *;if path is not blank, continue to next check*

IfFileExists `${\_b}\\*.\*` 0 +2 *;if directory exists, continue to confirm exists*

StrCpy $\_LOGICLIB\_TEMP "1"

StrCmp $\_LOGICLIB\_TEMP "1" `${\_t}` `${\_f}`

**!macroend**

**!define** DirExists `"" DirExists`

For[Each]..{ExitFor|Continue|Break}..Next

Var i

${ForEach} $i 10 0 - 2

MessageBox MB\_OK $i

${Next}

If|IfNot|Unless..{ElseIf|ElseIfNot|ElseUnless}..[Else]..EndIf|EndUnless

AndIf|AndIfNot|AndUnless|OrIf|OrIfNot|OrUnless

While..{ExitWhile|Continue|Break}..EndWhile

Do[While|Until]..{ExitDo|Continue|Break}..Loop[While|Until]

FileRead $fp $line

${DoUntil} ${Errors}

MessageBox MB\_OK $line

FileRead $fp $line

${LoopUntil} 1 = 0

Select..{Case[2|3|4|5]}..[CaseElse|Default]..EndSelect

Switch..{Case|CaseElse|Default}..EndSwitch

${Abort}*; ${Errors}; ${RebootFlag}; ${Silent}*

# **Setting Environment Variables Examples**

**!define** JAVA\_HOME "d:\JDK1.6"

**!define** APP\_HOME "d:\application"

Section "Add Env Var"

ReadEnvStr $R0 "PATH"

messagebox mb\_ok '$R0'

StrCpy $R0 "$R0;${JAVA\_HOME};${APP\_HOME}"

System::Call 'Kernel32::SetEnvironmentVariableA(t, t) i("PATH", R0).r2'

ReadEnvStr $R0 "PATH"

messagebox mb\_ok '$R0'

SectionEnd

**!define** JAVA\_HOME "d:\JDK1.5"

**!define** APP\_HOME "d:\application"

**!include** WriteEnvStr.nsh

Section "Add Env Var"

**!ifdef** ALL\_USERS

**!define** ReadEnvStr\_RegKey \

'HKLM "SYSTEM\CurrentControlSet\Control\Session Manager\Environment"'

**!else**

**!define** ReadEnvStr\_RegKey 'HKCU "Environment"'

**!endif**

Push JAVA\_HOME

Push '${JAVA\_HOME}'

Call WriteEnvStr

Push APP\_HOME

Push '${APP\_HOME}'

Call WriteEnvStr

ReadEnvStr $R0 "PATH"

messagebox mb\_ok '$R0'

*;ensure that is written valid for NT only*

ReadRegStr $0 ${ReadEnvStr\_RegKey} 'JAVA\_HOME'

ReadRegStr $1 ${ReadEnvStr\_RegKey} 'APP\_HOME'

StrCpy $R0 "$R0;$0;$1"

*;or just this*

*;StrCpy $R0 "$R0;${JAVA\_HOME};${APP\_HOME}"*

System::Call 'Kernel32::SetEnvironmentVariableA(t, t) i("PATH", R0).r2'

ReadEnvStr $R0 "PATH"

messagebox mb\_ok '$R0'

writeuninstaller '$EXEDIR\uninst.exe'

SectionEnd

*# ...*

Section uninstall

*# remove the variable*

Push JAVA\_HOME

Call un.DeleteEnvStr

Push APP\_HOME

Call un.DeleteEnvStr

SectionEnd