```
//*********************************************************
//      MACRO:      LONGNAME.WCM
//      PURPOSE:    Converts the name of selected documents from
//                  DOS (8.3) filenames to long filenames using the
//                  description from the Document Summary.
//*********************************************************
Application (A1; "WordPerfect"; Default; "EN")
DllCall Prototype SendMessageNN ("USER32"; "SendMessageA"; DWord; {hWnd; Msg;
WParam; LParam})

Constant (Loop := 0; OK := 1; Cancel := 2)
Constant (LogFileName := "longname.log")
Global Dlg := "LongNameDlg"
Global FilePath := ?PathDocument
Global Selections := ""
Global DlgStatus := Loop
Global hWndList

If (?NumberOpenDocuments > 8)
        MessageBox (; "WordPerfect for Windows"; "Cannot open a new document.  Please close
        a document and play the macro again."; IconStop!)
        Quit
EndIf

// Exit all substructures.
While (?Substructure)
        SubDoc := ?CurrentSubDoc
        SubstructureExit ()
        If ((SubDoc = 10) Or (SubDoc = 11))
                BoxEnd (Save!)
        EndIf
EndWhile

If (FilePath = "")
        FilePath := ?PathCurrent
EndIf

DefineDialog ()
DialogLoad (Dlg; "WordPerfect")
hWndList := RegionGetHandle (Dlg + ".ListBox")

FillList (FilePath)
RegionSetWindowText (Dlg + ".FileBox"; FilePath)
RegionSetModified (Dlg + ".FileBox"; NotModified!)
RegionEnableWindow (Dlg + ".OKBttn"; Disable!)
DialogShow (Dlg; "WordPerfect"; CB; "FileBox")
```

```
MainLoop ()

DialogDismiss (Dlg; "OKBttn")

If (DlgStatus = OK)
	// Get count of selected items in the listbox
	TotalCount := SendMessageNN (hWndList; 0190x; 0; 0)	// LB_GETSELCOUNT

	If (TotalCount != 0)
		ConvertNames (TotalCount)
	EndIf
EndIf

DialogDestroy (Dlg)
Quit

//****************************************
//	PROCEDURE: MainLoop
//	PURPOSE: Main idle loop.
//****************************************
PROCEDURE MainLoop ()
ButtonDisabled := True
Repeat
If (ButtonDisabled)
	If (RegionGetModified (Dlg + ".FileBox"))
		RegionEnableWindow (Dlg + ".OKBttn"; Enable!)
		ButtonDisabled := False
	Else
		Selections := RegionGetSelectedText (Dlg + ".ListBox")
		If (Selections != "")
			RegionEnableWindow (Dlg + ".OKBttn"; Enable!)
			ButtonDisabled := False
		EndIf
	EndIf
Else
	If (RegionGetModified (Dlg + ".FileBox"))
		RegionEnableWindow (Dlg + ".OKBttn"; Enable!)
		ButtonDisabled := False
		// Get count of items in the listbox
		Count := RegionGetListCount (Dlg + ".ListBox")
		// Unselect everything in the listbox
		SendMessageNN (hWndList; 0183x; Count; 0) // LB_SELITEMRANGEEX
	Else
		Selections := RegionGetSelectedText (Dlg + ".ListBox")
		If ((Selections = "") And (Not RegionGetModified (Dlg + ".FileBox")))
```

```
                RegionEnableWindow (Dlg + ".OKBttn"; Disable!)
                ButtonDisabled := True
            EndIf
        EndIf
EndIf
Until (DlgStatus != Loop)
ENDPROC

//*************************************
//      PROCEDURE: CB
//      PURPOSE: Callback routine for main dialog.
//*************************************
PROCEDURE CB ()
Switch (CB[3])
    CaseOf "":
        Switch (CB[5])
            CaseOf 274:    // WM_SYSCOMMAND
                DlgStatus := Cancel
            CaseOf 6:      // WM_ACTIVATE message
                If (RegionGetModified (Dlg + ".FileBox"))
                    FilePath := RegionGetWindowText (Dlg + ".FileBox")
                    FillList (FilePath)
                EndIf
        EndSwitch
    CaseOf "OKBttn":
        If (RegionGetModified (Dlg + ".FileBox"))
            FilePath := RegionGetWindowText (Dlg + ".FileBox")
            FillList (FilePath)
            Return
        EndIf
        Selections := RegionGetSelectedText (Dlg + ".ListBox")
        If (Selections != "")
            DlgStatus := OK
        EndIf
    CaseOf "CancelBttn":
        DlgStatus := Cancel
EndSwitch
ENDPROC

//*************************************
//      PROCEDURE: FillList
//      PURPOSE: Fills the listbox with files.
//*************************************
PROCEDURE FillList (PathSpec)
DLLCall Prototype SendMessageNS ("USER32"; "SendMessageA"; Integer; {hwndList; Msg;
wParam; AnsiString (LParam)})
```

```
DLLCall Prototype GetShortPathName ("KERNEL32"; "GetShortPathNameA"; DWord;
{AnsiString (LongPath); &AnsiString (ShortPath); Len})

OldPath := PathSpec
NewPath := PathSpec

// Get a short version of the path, just in case there are long filenames in the path.
GetShortPathName (PathSpec; PathSpec; 1024)
// Did anything change?
If (ToUpper (PathSpec) = ToUpper (OldPath))
        // No! Preserve the original case.
        PathSpec := OldPath
EndIf

// Is there there a * or ? in the path?
Loc1 := StrPos (PathSpec; "*")
Loc2 := StrPos (PathSpec; "?")
If ((Loc1 = 0) And (Loc2 = 0))
        // No! Assume the path given is a folder. Make sure the path ends in a slash.
        PathSpec := EnsureSlash (PathSpec)
        // Since this must be a folder, add the filespec of "*.*" onto the end.
        NewPath := PathSpec + "*.*"
EndIf
If (OldPath != PathSpec)
        RegionSetWindowText (Dlg + ".FileBox"; PathSpec)
        RegionSetModified (Dlg + ".FileBox"; NotModified!)
EndIf

RegionResetList (Dlg + ".ListBox")
SendMessageNS (hWndList; 018Dx; 0; NewPath)   // Do LB_DIR
RegionSetModified (Dlg + ".FileBox"; NotModified!)
ENDPROC

//**************************************
//      PROCEDURE: DefineDialog
//      PURPOSE: Defines the main dialog.
//**************************************
PROCEDURE DefineDialog ()
If (Not DoesDialogExist (Dlg))
        DialogDefine (Dlg; 50; 50; 174; 196; Percent!; "Convert to Long Filenames")
        DialogAddText (Dlg; 0; 8; 8; 100; 10; 0; "&Show files in:")
        DialogAddFilenameBox (Dlg; "FileBox"; 8; 19; 100; 14; DirOnly!; Folder; FilePath;
        "*.*")
        DialogAddText (Dlg; 0; 8; 37; 100; 10; 0; "Select &files to rename:")
        DialogAddListBox (Dlg; "ListBox"; 8; 48; 100; 100; Sorted! + NameSearch! +
        MultipleSelection! + ExtendedSelection!; Filename)
```

```
        DialogAddHLine (Dlg; 0; 8; 146; 100)
        DialogAddText (Dlg; 0; 8; 152; 100; 40; 0; "Selected files that have a descriptive name in
        the document summary will be converted to a long filename.")
        DialogAddPushButton (Dlg; "OKBttn"; 116; 8; 50; 14; OKBttn! + DefaultBttn!; "OK")
        DialogAddPushButton (Dlg; "CancelBttn"; 116; 26; 50; 14; CancelBttn!; "Cancel")
        DialogSave (Dlg)
EndIf
ENDPROC

//*************************************
//      PROCEDURE: ConvertNames
//      PURPOSE: Converts the selected files to long filenames based on their descriptive
names.
//*************************************
PROCEDURE ConvertNames (TotalCount)
SomeFailed := False
Declare Outfile[4]

WaitMsgInit ()
WaitMsgDisplay ("Converting files...")
OutFile[] := PathSplit (EnsureSlash (FilePath))
Folder := EnsureSlash (OutFile[1] + OutFile[2])
Count := 0
OpenLog (Folder)

Repeat
        RegionSetProgressPercent ("WaitMsg.Progress"; Count/TotalCount * 100)
        NameEnd := StrPos (Selections; ";")
        Filename := StrLeft (Selections; NameEnd - 1)
        Selections := SubStr (Selections; NameEnd + 1; StrLen (Selections) - NameEnd)
        If (Not Convert1Doc (Folder; Filename))
                SomeFailed := True
        EndIf
        Count := Count + 1
Until (NameEnd = 0)

CloseLog ()
WaitMsgHide ()
WaitMsgDestroy ()

If (hLog)
        MessageBox (; "WordPerfect for Windows"; "Conversion complete.  The current
        document contains a record of the changes that were made."; IconInformation!)
Else
        If (SomeFailed)
                MessageBox (; "WordPerfect for Windows"; "Conversion complete.  Some
```

```
            documents were not renamed."; IconInformation!)
        Else
            MessageBox (; "WordPerfect for Windows"; "Conversion complete.";
            IconInformation!)
        EndIf
EndIf
ENDPROC

//*************************************
//      PROCEDURE: Convert1Doc
//      PURPOSE: Converts a specific document to a long filename.
//*************************************
FUNCTION Convert1Doc (Dir; Filename)
Declare Outfile[4]
RetVal := False

OnError (GotError)
FileOpen (Dir + Filename)
OnError ()
GetData (Description; SummaryTag!; Data!; CurrentDoc!; DescriptiveName!)
CloseNoSave ()
// Remove any invalid characters from the name
Description := StrToChars (Description; Remove!; "\/:*?""<>|")
If (Description != "")
        // Get the current extension so that we can reuse it.
        OutFile[] := PathSplit (Filename)
        Extension := OutFile[4]

        // Make sure that the filename is not longer than 255 characters
        Description := StrLeft (Description; 255 - StrLen (Extension) - StrLen (Dir))
        // Rename this file
        OnError (GotError)
        If (Not RenameFile (Dir + Filename; Dir + Description + Extension))
            Go (GotError)
        EndIf
        OnError ()
        WriteLog (Filename; "renamed to "; Description + Extension)
Else
        WriteLog (Filename; "does not have a descriptive name."; "")
EndIf
Return (RetVal)

GotError:
OnError ()
WriteLog (Filename; "could not be renamed because of an error."; "")
Return (RetVal)
```

```
ENDFUNC

//****************************************
//      PROCEDURE: OpenLog ()
//      PURPOSE: Opens a log file.
//****************************************
PROCEDURE OpenLog (Path)
Global LogFile := Path + LogFileName
Global hLog := OpenFile (LogFile; WriteNew!; Exclusive!; AnsiText!)
If (hLog < 1)
        hLog := 0
        Return
EndIf
FileWrite (hLog; "Files in ^0:"; NewLine!; Path)
ENDPROC

//****************************************
//      PROCEDURE: CloseLog ()
//      PURPOSE: Closes a log file and displays it to the user.
//****************************************
PROCEDURE CloseLog ()
If (hLog)
        CloseFile (hLog)
        If (Not ?DocBlank)
                FileNew ()
        EndIf
        FontSize (10p)
        Font ("Courier")
        TabSet (Relative!; 2.5"; TabLeft!)
        FileInsert (LogFile; Yes!; Insert!)
        DeleteFile (LogFile; NoPrompts!)
        SearchString ("[Left Tab]")
        ReplaceString ("[Hd Left Ind]")
        ReplaceAll (Regular!)
        PosDocTop ()
EndIf
ENDPROC

//****************************************
//      PROCEDURE: WriteLog ()
//      PURPOSE: Writes a new entry to the log file.
//****************************************
PROCEDURE WriteLog (Filename; Status; NewName)
If (hLog)
        If (NewName != "")
                NewName := """" + NewName + """"
```

```
        EndIf
        FileWrite (hLog; "^0^1^2^3"; NewLine!; Filename; NToC (0F909x); Status; NewName)
EndIf
ENDPROC

//*************************************
//      FUNCTION: PathSplit
//      PURPOSE: Splits a path into 4 elements:
//              1: Drive Letter
//              2: File Path
//              3: Filename
//              4: File Extension
//*************************************
FUNCTION PathSplit (Filename)
DLLCall Prototype WfsPathSplit ("Pfit110"; "WfsPathSplit"; Void; {AnsiString (Filename);
&AnsiString (A); &AnsiString (B); &AnsiString (C); &AnsiString (D)})
Declare Out[] := {""; ""; ""; ""}

WfsPathSplit (Filename; Out[1]; Out[2]; Out[3]; Out[4])
Return (Out[])
ENDFUNC

//*************************************
//      PROCEDURE: EnsureSlash ()
//      PURPOSE: Puts a backslash on a directory if one is not already there.
//*************************************
FUNCTION EnsureSlash (PathSpec)
// Is there already a slash?
If ("\" != StrRight (PathSpec; 1))
        // No! Add it.
        PathSpec := PathSpec + "\"
EndIf
Return (PathSpec)
ENDFUNC

//*************************************
//      PROCEDURE: WaitMsgInit ()
//      PURPOSE: Initializes macro wait messages.
//*************************************
PROCEDURE WaitMsgInit ()
If (Not DoesDialogExist ("WaitMsg"))
        DialogDefine ("WaitMsg"; 50; 50; 150; 42; Percent!; "Please Wait")
        DialogAddText ("WaitMsg"; "WaitText"; 8; 8; 134; 10; Left!; "")
        DialogAddProgress ("WaitMsg"; "Progress"; 8; 19; 134; 15)
        DialogSave ("WaitMsg")
EndIf
```

```
DialogLoad ("WaitMsg"; "WordPerfect")
Return
ENDPROC

//*************************************
//      PROCEDURE: WaitMsgDisplay (Text)
//      PURPOSE: Displays a previously defined wait message dialog.
//*************************************
PROCEDURE WaitMsgDisplay (Text)
RegionSetWindowText ("WaitMsg" + ".WaitText"; Text)
DialogShow ("WaitMsg"; "WordPerfect"; WaitDlgCallBack)
ENDPROC

//*************************************
//      PROCEDURE: WaitDlgCallBack
//      PURPOSE: Callback function for the wait message.
//*************************************
PROCEDURE WaitDlgCallBack ()
If (WaitDlgCallBack[5] = 274 And WaitDlgCallBack[6] = 61536)
      Assert (CancelCondition!)
EndIf
ENDPROC

//*************************************
//      PROCEDURE: WaitMsgHide
//      PURPOSE: Hides a previously defined wait message dialog.
//*************************************
PROCEDURE WaitMsgHide ()
DialogDismiss ("WaitMsg"; "WaitText")
ENDPROC

//*************************************
//      PROCEDURE: WaitMsgDestroy
//      PURPOSE: Destroys a previously defined wait message dialog.
//*************************************
PROCEDURE WaitMsgDestroy ()
DialogDestroy ("WaitMsg")
ENDPROC
```