Sub FRedit()

' Paul Beverley - Version 21.03.25

' Scripted find and replace

' ©2009-2025 Paul Beverley

Set nowDoc = ActiveDocument

promptForSelectedText = True

doFinalBeep = True

showTime = True

useReverseStrikeFeature = False

debugging = False

' debugging = True

funnyCode = "Blank"

' used as the code to mean "Leave the Find/Replace box blank"

isMacro = "DoMacro"

' Used as the code to mean "Do the following macro"

caseCode = ChrW(172)

' The 'bent pipe' character (horizontal line with a bent end)

maxLines = 2000

' the maximum number of F&R lines in your list

myScreenOff = True

oldColour = Options.DefaultHighlightColorIndex

CR = vbCr: CR2 = CR & CR

Set workFile = ActiveDocument

Set rng = workFile.Content

justEditStrikeText = False

editAll = vbNo

If useReverseStrikeFeature = True And \_

rng.Font.StrikeThrough = 9999999 Then

editAll = MsgBox("Edit all text?", \_

vbQuestion + vbYesNoCancel, "FRedit")

If editAll = vbCancel Then Exit Sub

If editAll = vbNo Then

myPrompt = "Edit ONLY strikethrough text (= Yes)" & CR2

myPrompt = myPrompt & "DON'T edit strikethrough text (= No)" & CR2

myPrompt = myPrompt & "Panic! (= Cancel)"

myResponse = MsgBox(myPrompt, vbQuestion + vbYesNoCancel, "FRedit")

If myResponse = vbCancel Then Exit Sub

If myResponse = vbYes Then justEditStrikeText = True

End If

End If

ReDim findText(maxLines) As String, ReplaceText(maxLines) As String

ReDim fHlight(maxLines) As Integer, rHlight(maxLines) As Integer

ReDim fTxtCol(maxLines) As Long, rTxtCol(maxLines) As Long

ReDim fFontSize(maxLines) As Integer, rFontSize(maxLines) As Integer

ReDim styleArray(maxLines, 4) As String, funct(maxLines, 18) As Boolean

timeStart = Timer

For i = 1 To 30

mySpaces = mySpaces & "> "

Next i

myBaseStyle = workFile.Styles(wdStyleNormal)

' In case there's an error

Set thisRng = Selection.Range.Duplicate

' FRedit the selected text only?

Set rng = workFile.Content

If Len(Selection) = Len(rng) Then Selection.HomeKey Unit:=wdStory

Set rngSel = Selection.Range.Duplicate

If Selection.End <> Selection.start Then

If promptForSelectedText = True Then

myResponse = MsgBox("Work on selected text only?", \_

vbQuestion + vbYesNoCancel, "FRedit")

If myResponse = vbCancel Then Exit Sub

Else

myResponse = vbYes

End If

wasSelectedText = (myResponse = vbYes)

Else

wasSelectedText = False

End If

If debugging = False Then On Error GoTo ReportIt

If myScreenOff = True Then Application.ScreenUpdating = False

' Assume cursor is in the file to be edited

Set workFile = ActiveDocument

myTrack = workFile.TrackRevisions

' Find the FRedit list file

gottaList = 0

For Each myDoc In Application.Documents

DoEvents

pNum = myDoc.Paragraphs.count

myNum = 5

If pNum < 5 Then myNum = pNum

Set rng = myDoc.Paragraphs(myNum).Range

rng.start = 0

myTest = Replace(LCase(rng), " ", "")

If InStr(myTest, ChrW(124) & "fredit") Then

gottaList = gottaList + 1

If gottaList = 1 Then

myDoc.Activate

Set theList = myDoc

End If

End If

' Debug.Print myDoc.Name

Next myDoc

myWarning = "Please ensure that your FRedit list starts with: | FRedit" \_

& CR & CR2 & "Then place the cursor in the text to be edited, and rerun FRedit."

If gottaList = 0 Then

Beep

myResponse = MsgBox("Can't find a FRedit LIST." & CR2 & \_

myWarning, vbExclamation + vbOKOnly, "FRedit")

Exit Sub

End If

If gottaList > 1 Then

Beep

myTime = Timer

Do

Loop Until Timer > myTime + 0.2

Beep

myResponse = MsgBox("There are TWO or more FRedit LISTs open!" & CR2 & \_

"Is this the list you want to use?", vbQuestion + vbYesNo, "FRedit")

If myResponse <> vbYes Then

myResponse = MsgBox("Please close the list(s) you don't want to use," & \_

CR & "and then rerun FRedit", vbExclamation + vbOKOnly, "FRedit")

Exit Sub

End If

End If

' Debug.Print workFile.Name, theList.Name

If workFile = theList Then

Beep

myResponse = MsgBox("The CURSOR seems to be in the FRedit LIST." & CR2 & \_

"Place the cursor in the TEXT TO BE EDITED, and rerun FRedit.", \_

vbExclamation + vbOKOnly, "FRedit")

Exit Sub

End If

Set rng = theList.Content

' Check final line in the list is a paragraph marker

If Right(rng, 2) <> CR2 Then rng.InsertAfter Text:=CR

' Check for rogue manual line breaks in the list

mlb = InStr(rng.Text, Chr(11))

If mlb > 0 Then

Selection.start = mlb - 1

Selection.MoveLeft , 1

Selection.End = mlb + 1

Beep

MsgBox "Beware: FRedit list contains manual line breaks!"

Application.Run macroName:="ShowFormatting"

MsgBox "I've turned Show Formatting on, so you can see."

Application.ScreenUpdating = True

Exit Sub

End If

' Create the list of F&Rs

Selection.HomeKey Unit:=wdStory

FRitem = 0

fNotes = False: eNotes = False: Etext = True: BoxText = False

' FRitem is used to count the actual lines that are F&R commands

lastLine = theList.Paragraphs.count

For i = 1 To lastLine

' Look through the list for things that aren't actual F&Rs

Do

Do

Set rng = theList.Paragraphs(i).Range

StatusBar = mySpaces & "Reading line: " & Str(FRitem)

DoEvents

rng.End = rng.End - 1

allLine = rng

i = i + 1

' Keep going until you find a non-blank line

Loop Until Len(allLine) > 0 Or i > lastLine

' Check if it's a comment line, i.e. starting with a pad

firstChar = Left(allLine, 1)

If firstChar = "|" Then

If InStr(allLine, "rack =") > 0 Then

' Check for | Track command

Select Case LCase(Right(allLine, 3))

Case " no"

If myTrack Then

MsgBox "Please switch track changes off!", \_

vbOKOnly + vbExclamation, "FRedit"

workFile.Activate

Application.ScreenUpdating = True

Exit Sub

End If

Case "yes"

If myTrack = False Then

MsgBox "Please switch track changes on!", \_

vbOKOnly + vbExclamation, "FRedit"

workFile.Activate

Application.ScreenUpdating = True

Exit Sub

End If

Case Else

myError = 4: GoTo myErrorReport

End Select

End If

If InStr(allLine, "ootnotes =") > 0 Then

' Check for | Footnote command

If InStr(LCase(allLine), "yes") > 0 Then

fNotes = True

Else

fNotes = False

End If

End If

If InStr(allLine, "ndnotes =") > 0 Then

' Check for | Endnote command

If InStr(LCase(allLine), "yes") > 0 Then

eNotes = True

Else

eNotes = False

End If

End If

If InStr(allLine, "ext =") > 0 Then

' Check for | Text command

If InStr(LCase(allLine), "yes") > 0 Then

Etext = True

Else

Etext = False

End If

End If

If InStr(allLine, "box =") > 0 Or InStr(allLine, \_

"boxes =") > 0 Then

' Check for | Textboxes command

If InStr(LCase(allLine), "yes") > 0 Then

BoxText = True

Else

BoxText = False

End If

End If

End If

Loop Until firstChar <> "|" Or i > lastLine

i = i - 1

' If you find hashes, stop looking for F&R lines

If Left(allLine, 1) = "#" Then Exit For

' Debug.Print allLine

' It's an F&R line, so check for highlighting

If Len(allLine) > 1 Then

lineStart = rng.start

lineEnd = rng.End

' Has it got a vertical bar in it?

padPosition = InStr(allLine, "|")

' If not, it's a two-line F&R

If padPosition = 0 Then

' We've got two lines

fText = rng

' Clip off MatchCase and Wildcard indicators

mchWild = False

mchCase = True

If Left(fText, 1) = caseCode Then

fText = Right(fText, Len(fText) - 1)

mchCase = False

rng.start = rng.start + 1

End If

If Left(fText, 1) = "~" Then

fText = Right(fText, Len(fText) - 1)

mchWild = True

rng.start = rng.start + 1

End If

' What style is the Find in?

fStyle = rng.Style

If fStyle = myBaseStyle Then fStyle = ""

rng.End = lineStart + 1

' Check format & type colour of first char of Find

fItalic = rng.Italic

fBold = rng.Bold

fSuper = rng.Font.Superscript

fSub = rng.Font.Subscript

fUline = rng.Underline

fSmall = rng.Font.SmallCaps

fAllcaps = rng.Font.AllCaps

fDstrike = rng.Font.DoubleStrikeThrough

fFont = rng.Font.Name

fSize = rng.Font.Size

fHiColour = rng.HighlightColorIndex

fTxtColour = rng.Font.Color

i = i + 1

Set rng = theList.Paragraphs(i).Range

rng.End = rng.End - 1

lineEnd = rng.End

rText = rng

padPosition = InStr(rText, "|")

If padPosition > 0 Or Len(rng) = 0 Then myError = 2: GoTo myErrorReport

' What style is the Replace in?

rstyle = rng.Style

If rstyle = myBaseStyle Then rstyle = ""

rng.End = rng.start + 1

' Check format & type colour of first char of Replace

rItalic = rng.Italic

rBold = rng.Bold

rSuper = rng.Font.Superscript

rSub = rng.Font.Subscript

rUline = rng.Underline

rSmall = rng.Font.SmallCaps

rAllcaps = rng.Font.AllCaps

rDstrike = rng.Font.DoubleStrikeThrough

rFont = rng.Font.Name

rSize = rng.Font.Size

rHiColour = rng.HighlightColorIndex

rTxtColour = rng.Font.Color

Else

' It's all on one line, so no style change

' unless the style is different from Normal

' What style is the Find in?

fStyle = ""

rstyle = rng.Style

If rstyle = myBaseStyle Then rstyle = ""

' Chop up the line into F and R

fText = Left(allLine, padPosition - 1)

rText = Right(allLine, Len(allLine) - padPosition)

' Clip off MatchCase and Wildcard indicators

mchWild = False

mchCase = True

If Left(fText, 1) = caseCode Then

fText = Right(fText, Len(fText) - 1)

mchCase = False

rng.start = rng.start + 1

End If

If Left(fText, 1) = "~" Then

fText = Right(fText, Len(fText) - 1)

mchWild = True

rng.start = rng.start + 1

End If

rng.End = lineStart + 1

' Check format & type colour of first char of Find

If rng.Text = " " Then

rng.start = rng.End

rng.End = rng.End + 1

End If

fItalic = rng.Italic

fBold = rng.Bold

fSuper = rng.Font.Superscript

fSub = rng.Font.Subscript

fUline = rng.Underline

fSmall = rng.Font.SmallCaps

fAllcaps = rng.Font.AllCaps

fDstrike = rng.Font.DoubleStrikeThrough

fFont = rng.Font.Name

fSize = rng.Font.Size

fHiColour = rng.HighlightColorIndex

fTxtColour = rng.Font.Color

rng.End = lineStart + padPosition + 1

rng.start = lineStart + padPosition

' Check format & type colour of first char of Replace

rItalic = rng.Italic

rBold = rng.Bold

rSuper = rng.Font.Superscript

rSub = rng.Font.Subscript

rUline = rng.Underline

rSmall = rng.Font.SmallCaps

rAllcaps = rng.Font.AllCaps

rDstrike = rng.Font.DoubleStrikeThrough

rFont = rng.Font.Name

rSize = rng.Font.Size

rHiColour = rng.HighlightColorIndex

rTxtColour = rng.Font.Color

trackit = Not (rng.Font.StrikeThrough)

End If

FRitem = FRitem + 1

' Save all the F&R info in arrays

findText(FRitem) = fText

ReplaceText(FRitem) = rText

styleArray(FRitem, 1) = fStyle

styleArray(FRitem, 2) = rstyle

styleArray(FRitem, 3) = fFont

styleArray(FRitem, 4) = rFont

fHlight(FRitem) = fHiColour

rHlight(FRitem) = rHiColour

fTxtCol(FRitem) = fTxtColour

rTxtCol(FRitem) = rTxtColour

fFontSize(FRitem) = fSize

rFontSize(FRitem) = rSize

funct(FRitem, 1) = mchWild

funct(FRitem, 2) = mchCase

funct(FRitem, 3) = fBold

funct(FRitem, 4) = rBold

funct(FRitem, 5) = fItalic

funct(FRitem, 6) = rItalic

funct(FRitem, 7) = fSuper

funct(FRitem, 8) = rSuper

funct(FRitem, 9) = fSub

funct(FRitem, 10) = rSub

funct(FRitem, 11) = fUline

funct(FRitem, 12) = rUline

funct(FRitem, 13) = fSmall

funct(FRitem, 14) = rSmall

funct(FRitem, 15) = fAllcaps

funct(FRitem, 16) = rAllcaps

funct(FRitem, 17) = trackit

funct(FRitem, 18) = fDstrike

' ^p is not allowed in wildcard searches!

If mchWild And InStr(fText, "^" & "p") > 0 Then

myError = 5: GoTo myErrorReport

End If

If InStr(fText, "^" & "{") > 0 Then

myError = 95: GoTo myErrorReport

End If

' You can't do case insensitive AND wildcard

If mchWild And mchCase = False Then

myError = 6: GoTo myErrorReport

End If

End If

Next i

lastItem = FRitem

' Check the Normal font

normalSize = theList.Styles(myBaseStyle).Font.Size

normalFont = theList.Styles(myBaseStyle).Font.Name

workFile.Activate

' Remember if TC is on or off

trackNow = workFile.TrackRevisions

' Remember current cursor position and move cursor to top, for speed

Set thisRng = Selection.Range.Duplicate

Selection.HomeKey Unit:=wdStory

' Define the ranges

' Get the data out of the arrays

For FRitem = 1 To lastItem

fText = findText(FRitem)

rText = ReplaceText(FRitem)

fStyle = styleArray(FRitem, 1)

rstyle = styleArray(FRitem, 2)

fFont = styleArray(FRitem, 3)

rFont = styleArray(FRitem, 4)

fHiColour = fHlight(FRitem)

rHiColour = rHlight(FRitem)

fSize = fFontSize(FRitem)

rSize = rFontSize(FRitem)

fTxtColour = fTxtCol(FRitem)

rTxtColour = rTxtCol(FRitem)

mchWild = funct(FRitem, 1)

mchCase = funct(FRitem, 2)

fBold = funct(FRitem, 3)

rBold = funct(FRitem, 4)

fItalic = funct(FRitem, 5)

rItalic = funct(FRitem, 6)

fSuper = funct(FRitem, 7)

rSuper = funct(FRitem, 8)

fSub = funct(FRitem, 9)

rSub = funct(FRitem, 10)

fUline = funct(FRitem, 11)

rUline = funct(FRitem, 12)

fSmall = funct(FRitem, 13)

rSmall = funct(FRitem, 14)

fAllcaps = funct(FRitem, 15)

rAllcaps = funct(FRitem, 16)

trackit = funct(FRitem, 17)

fDstrike = funct(FRitem, 18)

If trackNow = True Then workFile.TrackRevisions = trackit

If fText = isMacro Then

Application.Run macroName:=rText

Else

' funnyCode means fText should be blank

If InStr(fText, funnyCode) > 0 Then fText = ""

BlankIt = False

If InStr(rText, funnyCode) > 0 Then rText = "": BlankIt = True

If fText = "<Symbol>" Then

fText = "": BlankIt = True

fFont = "Symbol"

rFont = normalFont

End If

' Replace hex code strings with codes

codePos = InStr(fText, "<&H")

Do While codePos > 0

codeLen = InStr(fText, ">") - codePos

uText = Mid(fText, codePos, codeLen)

uCode = Val(Right(uText, codeLen - 1))

uChar = ChrW(uCode)

fText = Replace(fText, uText & ">", uChar)

codePos = InStr(fText, "<&H")

Loop

codePos = InStr(rText, "<&H")

Do While codePos > 0

codeLen = InStr(rText, ">") - codePos

uText = Mid(rText, codePos, codeLen)

uCode = Val(Right(uText, codeLen - 1))

uChar = ChrW(uCode)

rText = Replace(rText, uText & ">", uChar)

codePos = InStr(rText, "<&H")

Loop

For hit = 1 To 4

If hit = 1 Then

If Not (fNotes = True And workFile.Footnotes.count \_

> 0) Then hit = 2

End If

If hit = 2 Then

If Not (eNotes = True And workFile.Endnotes.count \_

> 0) Then hit = 3

End If

If hit = 3 Then

If Etext = False Then hit = 4

End If

goes = 1

If hit = 4 Then

If BoxText = True Then

goes = workFile.Shapes.count

Else

hit = 5

End If

End If

If hit < 5 Then

For myGo = 1 To goes

If hit = 1 Then Set rng = \_

workFile.StoryRanges(wdFootnotesStory)

If hit = 2 Then Set rng = \_

workFile.StoryRanges(wdEndnotesStory)

If hit = 3 Then

If wasSelectedText Then

Set rng = rngSel.Duplicate

Else

Set rng = workFile.Content

End If

End If

someText = True

If hit = 4 Then

Do

someText = False

If workFile.Shapes(myGo).Type <> 24 \_

And workFile.Shapes(myGo).Type <> 3 Then

someText = workFile.Shapes(myGo).TextFrame.HasText

End If

If someText Then

Set rng = workFile.Shapes(myGo).TextFrame.TextRange

Else

myGo = myGo + 1

End If

Loop Until someText Or myGo > goes

End If

If someText = True Then

' Now do the F&R with the appropriate conditions set

Options.DefaultHighlightColorIndex = rHiColour

If (rHiColour <> fHiColour) And (fHiColour <> 0) Then

' But first emboss all text in fHiColour

Set rngNow = rng.Duplicate

For Each myPar In rngNow.Paragraphs

If myPar.Range.HighlightColorIndex > 9999 Then

For Each wd In myPar.Range.Words

If wd.HighlightColorIndex > 9999 Then

For Each ch In wd.Characters

If ch.HighlightColorIndex = fHiColour Then

ch.Font.Emboss = True

End If

Next ch

Else

If wd.HighlightColorIndex = fHiColour Then

wd.Font.Emboss = True

End If

End If

Next wd

Else

If myPar.Range.HighlightColorIndex = fHiColour Then

If Len(myPar.Range.Text) > 1 Then myPar.Range.Font.Emboss = True

End If

End If

DoEvents

Next myPar

End If

StatusBar = mySpaces & "F&Ring line: " & Str(FRitem) & \_

" of " & Str(lastItem) & " > > > " & fText & \_

"|" & rText

Debug.Print Str(FRitem) & " of " & Str(lastItem) & \_

" > " & fText & " | " & rText

' Now do the F&R

Set rngNow = rng.Duplicate

With rngNow.Find

.ClearFormatting

.Replacement.ClearFormatting

.Format = False

If wasSelectedText Then

.Wrap = False

Else

.Wrap = wdFindContinue

End If

.Text = fText

.Replacement.Text = rText

.MatchWildcards = mchWild

.MatchCase = mchCase

DoEvents

If fStyle > "" Or rstyle > "" Then

If rstyle = "" Then rstyle = myBaseStyle

If fStyle > "" Then .Style = fStyle

If rstyle > "" Then .Replacement.Style = rstyle

Else

' N.B. If changing styles, don't try to

' change bold, italic, etc, etc.

If fBold <> rBold Then

.Font.Bold = fBold

.Replacement.Font.Bold = rBold

End If

If rBold Then .Replacement.Font.Bold = True

If fItalic <> rItalic Then

.Font.Italic = fItalic

.Replacement.Font.Italic = rItalic

End If

If rItalic Then .Replacement.Font.Italic = True

If fSuper <> rSuper Then

.Font.Superscript = fSuper

.Replacement.Font.Superscript = rSuper

End If

If rSuper Then .Replacement.Font.Superscript = True

If fSub <> rSub Then

.Font.Subscript = fSub

.Replacement.Font.Subscript = rSub

End If

If rSub Then .Replacement.Font.Subscript = True

If fUline <> rUline Then

.Font.Underline = fUline

.Replacement.Font.Underline = rUline

End If

If rUline Then .Replacement.Font.Underline = True

If fSmall <> rSmall Then

.Font.SmallCaps = fSmall

.Replacement.Font.SmallCaps = rSmall

End If

If rSmall Then .Replacement.Font.SmallCaps = True

If fAllcaps <> rAllcaps Then

.MatchCase = fAllcaps

.Font.AllCaps = rAllcaps

.Replacement.Font.AllCaps = False

End If

If rAllcaps Then

.MatchCase = False

.Replacement.Font.AllCaps = True

End If

If rDstrike Then .Replacement.Font.StrikeThrough = True

If fStyle = rstyle Then

If fFont = rFont And fFont <> normalFont Then

.Replacement.Font.Name = rFont

End If

If fFont <> rFont Then

.Font.Name = fFont

.Replacement.Font.Name = rFont

End If

If fSize = rSize And fSize <> normalSize Then

.Replacement.Font.Size = rSize

End If

If fSize <> rSize Then

.Font.Size = fSize

.Replacement.Font.Size = rSize

End If

End If

End If

If rHiColour <> fHiColour And fHiColour <> 0 And rText > "" And \_

fHiColour > 0 Then .Font.Emboss = True

If rText > "" And (rHiColour > 0 Or fHiColour > 0) Then

.Replacement.Highlight = True

If rHiColour = 0 Then .Replacement.Highlight = False

End If

If rText > "" Then

If rTxtColour = fTxtColour Then

If fTxtColour > 0 Then \_

.Replacement.Font.Color = fTxtColour

Else

If rTxtColour > 0 Or fTxtColour > 0 Then

.Font.Color = fTxtColour

If BlankIt = False Then .Replacement.Font.Color = \_

rTxtColour

End If

End If

End If

If justEditStrikeText = True Then

.Font.StrikeThrough = True

Else

If editAll = vbNo Then .Font.StrikeThrough = False

End If

.Font.DoubleStrikeThrough = False

.Font.Hidden = False

.Execute Replace:=wdReplaceAll

End With

DoEvents

If rHiColour <> fHiColour Then rng.Font.Emboss = False

End If

Next myGo

End If

Next hit

End If

Next FRitem

StatusBar = ""

' Restore highlight colour to normal

Options.DefaultHighlightColorIndex = oldColour

If myError < 7 Or myError > 16 Then thisRng.Select

If wasSelectedText Then rngSel.Select

totTime = Timer - timeStart

If showTime = True And totTime > 60 Then

MsgBox ((Int(10 \* totTime / 60) / 10) & \_

" minutes")

Else

If doFinalBeep = True Then

Beep

myTime = Timer

Do

Loop Until Timer > myTime + 0.2

Beep

End If

End If

workFile.TrackRevisions = myTrack

Application.ScreenUpdating = True

Exit Sub

' Warn the user about problems that the macro has detected

myErrorReport:

If myError <= 6 Or (myError > 11 And myError < 17) \_

Or myError = 95 Then

rng.Select

Selection.Expand wdParagraph

End If

Select Case myError

Case 2: myPrompt = "No matching replace text"

Selection.MoveStart Unit:=wdParagraph, count:=-1

' Selection.Expand wdParagraph

Case 4: myPrompt = "A 'Count =' line should say 'yes' or 'no'."

Case 5: myPrompt = "Sorry, Word can't use ^p in a wildcard search." \_

& CR2 & "On Word for Mac, try [^13]." & CR2 \_

& "On Word for Windows, try ^13."

Case 6: myPrompt = "Sorry, Word can't do case insensitive " & \_

"searches with wildcards."

Case 13: myPrompt = "A 'Footnotes =' line should say 'yes' or 'no'."

Case 14: myPrompt = "An 'Endnotes =' line should say 'yes' or 'no'."

Case 15: myPrompt = "A 'Text =' line should say 'yes' or 'no'."

Case 16: myPrompt = "A 'Textboxes =' line should say 'yes' or 'no'."

Case 17: myPrompt = "You have used an unacceptable ^(something) in a search."

Case 19: myPrompt = "Unacceptable pattern match in this F&R line."

Case 21: myPrompt = "Can't find a macro called: " & rText

Case 22: myPrompt = "More Replace groups than Find groups in wildcard F&R."

Case 23: myPrompt = "^? is not a valid special character in wildcard F&R."

Case 24: myPrompt = "^ on its own is not a valid character in Replace."

Case 25: myPrompt = "The style you are trying to apply does not exist." & CR2 \_

& fText & ChrW(124) & rText

Case 95: myPrompt = "Sorry, Word can't use ^{ in a search." \_

& CR2 & "Use ^94{ instead."

Case Else: myPrompt = "Progam error; please inform Paul Bev."

End Select

thisRng.Select

MsgBox myPrompt, vbOKOnly + vbExclamation, "FRedit"

Options.DefaultHighlightColorIndex = oldColour

workFile.TrackRevisions = myTrack

theList.Activate

Application.ScreenUpdating = True

Exit Sub

' Errors that Word generates end up here

ReportIt:

Application.ScreenUpdating = True

' DoMacro call to unknown macro

If Err.Number < 0 Then myError = 21: GoTo myErrorReport

' Can't find the files it needs

' If Err.Number = 4248 Then myError = 11: GoTo myErrorReport

' Trying to run FRedit from the Zip file

' If Err.Number = 5941 Then myError = 12: GoTo myErrorReport

' Non-existent style

If Err.Number = 5834 Then myError = 25: GoTo myErrorReport

errNow = Err.Number

' If we've found the list, select it...

theList.Activate

Set rng = theList.Content

' ... and look for the current line in the list

' which is probably where the problem lies.

gottit = False

Dim myLine As Single

myLine = 1

For j = 1 To theList.Paragraphs.count

myTxt = theList.Paragraphs(j)

If Asc(myTxt) <> Asc("|") And Len(myTxt) > 2 Then

If InStr(myTxt, "|") > 0 Then

myLine = myLine + 1

Else

myLine = myLine + 0.5

End If

End If

If myLine > FRitem Then Exit For

Next j

theList.Paragraphs(j).Range.Select

If Err.Number = 5625 Or Err.Number = 5692 Then

myError = 17

GoTo myErrorReport

End If

If errNow = 5560 Then

myError = 19

GoTo myErrorReport

End If

If errNow = 5560 Or errNow = 5590 Then

myError = 19

GoTo myErrorReport

End If

' Wildcard error - too many groups

If errNow = 5623 Then myError = 22: GoTo myErrorReport

If errNow = 5692 Then myError = 23: GoTo myErrorReport

If errNow = 5624 Then myError = 24: GoTo myErrorReport

' Display Word's error message

Application.ScreenUpdating = True

On Error GoTo 0

Resume

End Sub