Hints and tips mine

This is just a random collection that I've made over the years (most recent at the top) of 'useful stuff': all sorts of ideas, hints, web links, explanations etc. Much of it is macro code segments, but if you search the file for a word/short phrase of interest, you might find something helpful. Hope it helps!

PB (21/12/22)

##

Sub TemplatesFolderAddress()
' Paul Beverley - Version 18.12.22
' Types the address of the current templates foler

full = NormalTemplate.FullName
nm = NormalTemplate.Name

Selection.InsertAfter Text:=Replace(full, nm, "")
Selection.Copy
WordBasic.EditUndo
End Sub

##

Finding the location (pathname) of the Normal template
Word 365 for Mac
Open any Word file. Run the below NTaddress macro.

Sub NTaddress()
' Version 26.06.20
' Locates your Normal template folder

Documents.Add
Selection.TypeText Text:=NormalTemplate.FullName
End Sub

This macro will create a new Word file containing the pathname for where the Normal template is located on your computer. It should look something like this:

/Users/[YourName]/Library/Group Containers/UBF8T346G9.Office/User Content.localized/Templates.localized/Normal.dotm

Now jump to the Displaying the location of the Normal template in Finder section below.

##

Somehow, my trust center settings decided to check the boxes that enabled
Protected View for
files originating from the Internet
located in potentially unsafe locations
for Outlook attachments (I don't even use Outlook).

I unchecked them and it solved the problem! Thanks to all who tried to help.

I found this site to be helpful
https://www.easeus.com/resource/word-experienced-an-error-trying-to-open-the-file.html#part1

##########

```
'  With ActiveWindow
'    .Top = 0
'    .Left = 1440
'    .Width = 1400
'    .Height = 800
'  End With
```

##########

"Application.Move Left:=1440, Top:=2" command is not available.
Does anyone have any tricks up their sleeve for a Mac running Monterey and up-to-date Word 365?
Thanks.
To which I add -- I found something that works, although it places the window about a thumb's width from the top of the screen, for some reason. But it's definite progress, and quite usable. Hurray!
Here's the macro I came up with:

```
Sub FullScreen()
   With ActiveWindow
      .Top = 0
      .Left = 0
      .Width = 1400
      .Height = 800
   ActiveWindow.ActivePane.View.Zoom.PageFit = wdPageFitBestFit
End With
End Sub
```

##########

Select the final actual line of text in the file!

```
Selection.EndKey Unit:=wdStory
' Select bottom line
Selection.MoveStart wdSentence, -1
```

##########

```
Sub CycleWindows()
' Jumps through the windows, activating them one by one, each time you run it

   If Application.ActiveWindow.Index < Application.Windows.Count Then
      Application.Windows(Application.ActiveWindow.Index + 1).Activate
   Else
      Application.Windows(1).Activate
   End If
```

```
    Debug.Print "Current Window is.....: " & Application.ActiveWindow.Index
End Sub
```

Show all information about open windows on different documents:

```
Sub DocumentsAndWindows()

    Dim vDocument As Document
    Dim vWindow As Window

    Debug.Print "Number of Documents.....: " & Application.Documents.count
    For Each vDocument In Application.Documents
        Debug.Print "   Document.............:" & vDocument.Name
    Next
    Debug.Print

    Debug.Print "Number of Windows.....: " & Application.Windows.count
    For Each vWindow In Application.Windows
        Debug.Print "   Window.............: " & vWindow.Index & " - " & vWindow.Document.Name
    Next
    Debug.Print "   Current Window.....: " & Application.ActiveWindow.WindowNumber
    Debug.Print

End Sub
```

##

```
Call Wait(3)
SendKeys "{TAB}"
SendKeys "{TAB}"
SendKeys "{TAB}"
SendKeys "{TAB}"
SendKeys "{TAB}"
SendKeys "{TAB}"
SendKeys "{TAB}"
SendKeys "{TAB}"
SendKeys "{TAB}"
SendKeys "^v"
SendKeys "{ENTER}"

Sub Wait(n As Long)
'from https://social.msdn.microsoft.com/Forums/en-US/08128481-8006-4f1d-935d-0a960b47aa74/wait-macro-in-word-2010?forum=worddev

Dim t As Date
t = Now
Do
DoEvents
Loop Until Now >= DateAdd("s", n, t)
End Sub
```

```
#####################################################

Options.PasteSmartCutPaste = False
Options.SmartParaSelection = False
Options.SmartCursoring = False
Options.AutoWordSelection = False

Exit Sub
Options.AutoWordSelection = True
Options.SmartParaSelection = True
Options.SmartCursoring = True
Options.PasteSmartCutPaste = True

  Exit Sub

myOptWS = Options.AutoWordSelection
Options.AutoWordSelection = False

myOptCP = Options.PasteSmartCutPaste
Options.PasteSmartCutPaste = False

myOptSC = Options.SmartCursoring
Options.SmartCursoring = False

myOptPS = Options.SmartParaSelection
Options.SmartParaSelection = False


myOptWS = Options.AutoWordSelection
Options.AutoWordSelection = False
Options.AutoWordSelection = myOptWS

Options.PasteSmartCutPaste = myOptCP
Options.SmartCursoring = myOptSC
Options.SmartParaSelection = myOptPS


End Sub

Sub SmartOptsOff()
  Options.AutoWordSelection = False
Exit Sub
  Options.PasteSmartCutPaste = False
  Options.SmartParaSelection = False
  Options.SmartCursoring = False


#####################################################
  ActiveDocument.Range.ParagraphFormat.Alignment = wdAlignParagraphJustify
  ActiveDocument.Range.ParagraphFormat.Alignment = wdAlignParagraphJustifyHi
  ActiveDocument.Range.ParagraphFormat.Alignment = wdAlignParagraphJustifyMed
  ActiveDocument.Range.ParagraphFormat.Alignment = wdAlignParagraphJustifyLow
  ActiveDocument.Range.ParagraphFormat.Alignment = wdAlignParagraphDistribute
  ActiveDocument.Range.ParagraphFormat.Alignment = wdAlignParagraphThaiJustify
```

##

Dear Mr Beverley,

I came across your MS Word macros for proofreading last year, and I would like to thank you for putting such a huge body of work in the public domain. Although I have been writing macros in Excel for about 20 years, I had never written one in MS Word. The discovery of your macros inspired me to write my first MS Word macro a few months ago.

I had always found it frustrating to have to manually format subscripts and superscripts in chemical formulae, i.e. changing H2O to H2O. One scientific report that I reviewed recently contained over 600 chemical terms! Therefore, I wrote a macro called ProofChem, which is a one-click macro for detecting and formatting all chemical formulae in a manuscript. ProofChem can automatically recognise and format quite complex chemical terms, such as:

· [Ag(S2O3)2]3- --> [Ag(S2O3)2]3-

· K342K[Fe(14CN)6] --> K342K[Fe(14CN)6]

Other tools I found on the internet required the user to choose and apply a format for each chemical term individually, and often were not able to discern real chemical formulae from similar looking terms that should not be formatted. I'm not a chemist, so on the advice of a colleague, I designed my macro to align where possible with the conventions in the Nomenclature of Inorganic Chemistry published by IUPAC (https://iupac.org/what-we-do/books/redbook). Chemical formulae can be quite obscure in specialist scientific articles, so the macro is not infallible. Nevertheless, my chemist colleague told me that in his view it covered 95% of everyday terms used by a normal chemist in, say, technical reports.

Although I wrote the macro on "company time", my employer has approved the distribution of the macro with an open-source license, and the macro is now freely available on github. I think you mentioned in one of your videos that your proofread many scientific manuscripts, so I thought you might find ProofChem useful and I just wanted to draw your attention to it.

You can find a short video demo of the macro on YouTube: https://youtu.be/r5G0IRT8YlU, and the macro itself can be downloaded from here: https://github.com/ec-jrc/jrc-proofreading

Thank you again for sharing your magnum opus of macros for the proofreading community, which I - and I'm sure many others - found most useful and inspiring.

With very best wishes,

##
I am continuing "fine-tuning" the appearance of our Liturgical (worship) Books at the Monastery. The text is set Aligned Left, Ragged Right; formatting which is the current usage in most Anglican Prayer Books. There was a style of formatting used by a company named Cogent-Elliot, in which the text was set Aligned Left, but then adjusted by expanding and condensing Letter and Inter-word spacing line by line or a paragraph at a time, to even out the right edge.

I have several macros adapted from code I've collected over the years, such as:

```
Sub ExpandLetterSpacing()
  Dim LSpacing
  LSpacing = Selection.Font.Spacing
   Selection.Font.Spacing = LSpacing + 0.05
```

```
' How to limit adjustment to a maximum of 0.25 points?

End Sub
```

to adjust the text. We use Sabon MT set 16/18, and I have found that the text can be adjusted +/- .25 points, without it becoming obvious that the text and or spaces have been adjusted. How can I modify the code to limit adjustment to a maximum of +/- 0.25 points?

##########
Word frequency program:

http://neon.niederlandistik.fu-berlin.de/textstat/

##########

```
Sub MergeDocs()
    Dim rng As Range
    Dim MainDoc As Document
    Dim strFile As String
    Const strFolder = "P:\Doc files\New folder\New folder\" 'change to suit
    Set MainDoc = Documents.Add
    strFile = Dir$(strFolder & "*.doc") ' can change to .docx
    Do Until strFile = ""
        Set rng = MainDoc.Range
        rng.Collapse wdCollapseEnd
        rng.InsertFile strFolder & strFile
        strFile = Dir$()
    Loop
    MsgBox ("Files are merged")

End Sub
```

##########

```
Sub SaveEachSectionAsADoc()
  Dim objDocAdded As Document
  Dim objDoc As Document
  Dim nSectionNum As Integer
  Dim strFolder As String

  Dim dlgFile As FileDialog

  ' Initialization
  Set objDoc = ActiveDocument

  Set dlgFile = Application.FileDialog(msoFileDialogFolderPicker)

  ' Pick a location to keep new files.
  With dlgFile
    If .Show = -1 Then
      strFolder = .SelectedItems(1) & "\"
    Else
      MsgBox "Select a folder first!"
```

```
      Exit Sub
    End If
  End With
  Debug.Print strFolder

  ' Step through each section in current document, copy and paste each to a new one.
  For nSectionNum = 1 To ActiveDocument.Sections.Count
    Selection.GoTo What:=wdGoToSection, Which:=wdGoToNext, Name:=nSectionNum
    ActiveDocument.Sections(nSectionNum).Range.Copy

    Set objDocAdded = Documents.Add
    Selection.Paste

    ' Save and close new documents.
    objDocAdded.SaveAs FileName:=strFolder & "Section " & nSectionNum & ".docx"
    objDocAdded.Close
  Next nSectionNum
End Sub
```

##

Select line, select page, select section

https://docs.microsoft.com/en-us/office/vba/word/concepts/miscellaneous/predefined-bookmarks

```
Sub whatever()

Selection.Bookmarks("\line").Range.Select

'Set myrange = Selection.Bookmarks("\section").Range

    Set rng = Selection.Bookmarks("\page").Range
    rng.Select
rng.Collapse wdCollapseStart
    Set rng2 = Selection.Bookmarks("\line").Range
rng2.Select
```

##

Could this help in nudging the display up, to get an extra line on screen?
For use with HighlightFindDown/Up

```
Sub SelectionToTop()
    Dim pLeft As Long
    Dim pTop As Long, loopTop As Long, windowTop As Long
    Dim pWidth As Long
    Dim pHeight As Long, windowHeight As Long
    Dim Direction As Integer

    windowHeight = PixelsToPoints(ActiveWindow.Height, True)
    ActiveWindow.GetPoint pLeft, windowTop, pWidth, pHeight, ActiveWindow
    ActiveWindow.GetPoint pLeft, pTop, pWidth, pHeight, Selection.Range

    'Direction, defines scroll direction -1 0 or 1
    Direction = Sgn((pTop + pHeight / 2) - (windowTop + pHeight))
```

```
    Do While Sgn((pTop + pHeight / 2) - (windowTop + pHeight)) = Direction And (loopTop <> pTop)
      ActiveWindow.SmallScroll Direction
      loopTop = pTop
      ActiveWindow.GetPoint pLeft, pTop, pWidth, pHeight, Selection.Range
    Loop
End Sub
```

##########

Extracting Text to a new document

```
Public Sub PrayerCondense()
Dim P As Long

ActiveDocument.SaveAs2 "Prayer 2 Abridged"

P = 1

With ActiveDocument.Paragraphs
  While P <= .Count

    Select Case .Item(P).Style
      Case "Service Title", "Prayer 2 Title", "Prayer 2"
        'Do nothing
        P = P + 1
      Case Else
        .Item(P).Range.Delete
    End Select
  Wend
End With
End Sub
```

-----Original Message-----
Subject: Extracting Text to a new document

I hope this finds you all well. I am working on my Communities Worship materials. Each service has several prayers, which for the sake of illustration I will refer to as Prayer 1 - Prayer 4. I would like to extract the following texts, (which has the following styles applied to them) Service Title, Prayer 2 Title, Prayer 2 Body to a new document. The original document is several hundred pages long.

What would be the best way to do this? A search? A find and replace? Some sort of an array?

Thank you for any assistance.
Brother Jeremy, CSJW

##########

Mac things

```
    System.Cursor = wdCursorWait

    System.Cursor = wdCursorNormal

horz = System.HorizontalResolution
vert = System.VerticalResolution
```

```
Sub TestSystem()
' Version 27.12.20
' Test OS reporting

Documents.Add
Selection.TypeText "PCode = " & Application.ProductCode & vbCr
Selection.TypeText "Version = " & Application.Version & vbCr
Selection.TypeText "System OS = " & Application.System.OperatingSystem & vbCr
Selection.TypeText "System version = " & Application.System.Version & vbCr
Selection.TypeText "OS Mac = " & Application.System.MacintoshName & vbCr
Selection.TypeText "Horz res = " & System.HorizontalResolution & vbCr
Selection.TypeText "Vert res = " & System.VerticalResolution & vbCr


##################################################

Options.SmartCutPaste = False
Options.SmartCutPaste = True

##################################################
```

If you do the arithmetic, your coding requires F&R to be repeated mypairs x 4 times. In tests on my large file the time for each F&R varies from 1.6 seconds if there are 5 instances to 5.1 seconds if there are 890 instances. Therefore if you have 1000 pairs and assuming an average search time of 2 seconds then the time taken will be 2.2 hours.

My previous macro CountHyphenWords gives a list of hyphenated words and a word count but it does not count variations such as cowbell and cow bell, using F&R for this would be time consuming. However, the macro produces an array of all words in the document and scanning an array is much faster than F&R. The macro has been expanded to CountHyphenWordsAndVariations and gives the word counts including three variations. Tested on a 880,000 word document the total time was 9 minutes for 4100 hyphenated words - 5 minutes if restricted to 1000 hyphenated words.

```
*A lot better than 2.2 hours.*
Sub CountHyphenWordsAndVariations()
Dim wrd As Range
Dim a() As String
Dim b() As String
Dim c() As String
Dim w() As Long
Dim j As Long
Dim k As Long
Dim kk As Long
Dim n As Long
Dim s As String
Dim schWd As String
Dim ss As String

Dim ta As Single
ta = Timer
```

```
' *** create array a() of all words ****
k = 0
ReDim a(0)
For Each wrd In ActiveDocument.Words
  k = k + 1
  ReDim Preserve a(k)
  a(k) = Trim(wrd.Text)
Next wrd

' **** Create array b() of hyphenated words ****
j = 0
ReDim b(0)
For k = 2 To UBound(a) - 1
  If a(k) = "-" Then
    s = Right(a(k - 1), 1)
    If s >= "A" And s <= "z" Then
      s = Left(a(k + 1), 1)
      If s >= "A" And s <= "z" Then
        j = j + 1
        ReDim Preserve b(j)
        b(j) = a(k - 1) & "-" & a(k + 1)
      End If
    End If
  End If
Next k

' **** Sort array b() case sensitive ****
For k = 1 To UBound(b) - 1
  For j = k + 1 To UBound(b)
    If b(k) > b(j) Then
      s = b(k)
      b(k) = b(j)
      b(j) = s
    End If
  Next j
Next k

' **** unique hyphenated words in c(),  count in w() ****
k = 0
s = ""
ReDim w(0)
ReDim c(0)
For j = 1 To UBound(b)
  If b(j) <> s Then
    s = b(j)
    k = k + 1
    ReDim Preserve c(k)
    ReDim Preserve w(k)
    c(k) = b(j)
    w(k) = 1
  Else
    w(k) = w(k) + 1
  End If
Next j

MsgBox "Phase 1" & Str(Timer - ta)
```

```
ta = Timer

 ' **** word count of hyphenated word variations ****
 ReDim x(UBound(c), 3)
 For k = 1 To UBound(c)
    ss = c(k)
    For j = 1 To 3
      Select Case j
        Case 1: schWd = Replace(ss, "-", "")
        Case 2: schWd = Replace(ss, "-", " ")
        Case 3: schWd = Replace(ss, "-", ChrW(8211))
      End Select
      n = 0
      For kk = 1 To UBound(a)
        If a(kk) = schWd Then n = n + 1
      Next kk
      x(k, j) = n
    Next j
 Next k

MsgBox "Phase 2" & Str(Timer - ta)

 ' **** display first 4 hyphernated words ****
 s = "Number of unique hyphenated words" & Str(UBound(c)) & vbCrLf
 For k = 1 To 4
   s = s & c(k) & Str(w(k)) & "  no hyphen" & Str(x(k, 1)) & _
      "  space" & Str(x(k, 2)) & "  long hyphen" & Str(x(k, 3)) & vbCrLf
 Next k
 MsgBox s
End Sub
```

Subject: Re: VBA help, please
To: WORD-PC@liverpool.ac.uk

I think that the following satisfies your requirements, it results in two arrays, a() contains a sorted list of unique hyphenated words and x() contains counts of how many times each word appears in the document. Separate entries are generated for lower case, title case and uppercase variations of the words.

For my test document of 688,000 words it took 4 minutes to give the result of 4166 unique words out of 14000 hyphenated words. I think that my method is about as fast as you can get, some alternatives that I tried take over an hour.

It would be straight forward to get counts of alternative words that have the hyphen replaced by a space or deleted ("cow bell" and "cow bell") by scanning down a() and using the function CountWords to determine the number of times they appear in the document.

```
Sub CountHyphenWords()
Dim wrd As Range
Dim a() As String
Dim b() As String
Dim x() As Long
Dim j As Long
Dim k As Long
Dim s As String
```

```
' *** create array a() of all words ****
k = 0
ReDim a(0)
For Each wrd In ActiveDocument.Words
  k = k + 1
  ReDim Preserve a(k)
  a(k) = Trim(wrd.Text)
Next wrd

' **** Create array b() of hyphenated words ****
j = 0
ReDim b(0)
For k = 2 To UBound(a) - 1
  If a(k) = "-" Then
    s = Right(a(k - 1), 1)
    If s >= "A" And s <= "z" Then
      s = Left(a(k + 1), 1)
      If s >= "A" And s <= "z" Then
        j = j + 1
        ReDim Preserve b(j)
        b(j) = a(k - 1) & "-" & a(k + 1)
      End If
    End If
  End If
Next k

' **** Sort array b() case sensitive ****
For k = 1 To UBound(b) - 1
  For j = k + 1 To UBound(b)
    If b(k) > b(j) Then
      s = b(k)
      b(k) = b(j)
      b(j) = s
    End If
  Next j
Next k

' **** Word count, Re-use array a() ****
k = 0
s = ""
ReDim a(0)
ReDim x(0)
For j = 1 To UBound(b)
  If b(j) <> s Then
    s = b(j)
    k = k + 1
    ReDim Preserve a(k)
    ReDim Preserve x(k)
    a(k) = b(j)
    x(k) = 1
  Else
    x(k) = x(k) + 1
  End If
Next j
MsgBox "Number of unique hyphenated words=" & UBound(a)
' **** The unique hyphenated words are in array a() and the wordcount is
```

```
in array x()
End Sub



Function CountWords(WordText As String) As Long
Dim aRange As Range
Dim n As Long
Dim atest As Boolean
  n = 0
  ActiveDocument.GoTo(What:=wdGoToBookmark, Name:="\StartOfDoc").Select
  Application.ScreenUpdating = False
  With Selection.Find
    .ClearFormatting
    .Text = WordText
    .MatchWholeWord = True
    .MatchCase = True
    Do Until Not .Execute
      n = n + 1
    Loop
  End With
  Application.ScreenUpdating = True
  CountWords = n
End Function

--
```

##

Hi Paul,

I've been using Jacques's utility regularly, as I have two clients that want different names in Tracked Changes. It works. Sometimes I have to change a file security attribute before it'll work.

It looks as though he's doing it using XML. That's the feeling I get from the messages displayed during processing. Both the Tracked Changes and Comments are handled.

From a Google search:

< https://docs.microsoft.com/en-us/office/open-xml/how-to-retrieve-comments-from-a-word-processing-document >

Santhosh

##

```
   Set myrange = Selection.Bookmarks("\page").Range
   myrange.Select

   Set myrange = Selection.Bookmarks("\line").Range
```

figures = "ddddddddddg"
clip.SetText figures
clip.PutInClipboard

##

And here is the very useful list of enumerations.

https://docs.microsoft.com/en-us/office/vba/api/word(enumerations)

It's part of the Word documentation. Here:
https://docs.microsoft.com/en-us/office/vba/api/overview/Word/object-model

##
Built-in Dialog Box Argument Lists (Word)

See file: Dialogs.doc in this directory.

##

If you have a computer that doesn't have a Break key, and you suspect that the macro you want to run might need stopping in the middle of its run, open the VBA window, and run the macro from there. Then you can use the pause and/or stop icons (with icons like those on a DVD player).
Well, you don't need to actually *run* it from VBA, as long as the VBA window is accessible.

##
The other thing to try is to go into Visual Basic and do

Tools -- references

It *should* say something like:

Available References:
Visual Basic for Applications
Microsoft Word 16.0 Object Library (or 14.0 in earlier versions of Word)
OLE Automation
Microsoft Word 16.0 Object Library (or 14.0 in earlier versions of Word)
Microsoft Scripting Runtime
Microsoft Forms 2.0 Object Library

You'll probably be missing two:

Look down through the list of unticked items and see if you've got
"Microsoft Scripting Runtime" and "Microsoft Forms 2.0 Object Library".

Tick them, click OK, close VBA and then close and re-open Word.

Does that do the trick? (If not, try a complete restart of the computer.)

##

I've just tried to make my macros *not* work, by using

File - Options - Trust Centre - Trust Centre Settings

and setting it to "Disable all macros".

And all my macros still work.

##########
Hello Paul

I've just been browsing through your latest version of 'The Book' and noticed the *CharToMacron* provided by a fellow Kiwi.

For a while I made use of MS Word's Maori keyboard in order to insert macrons but then I became aware of the 'Unicode Diacritics Macro for MS Word' available from https://collab.its.virginia.edu/wiki/toolbox/New%20Diacritics%20Word%20Macro.html that enables the insertion of almost every kind of diacritic, and I've found it very useful - not only for macrons but also common accents like grave, acute, circumflex, diaeresis, etc. that I come across fairly often. It might be 'overkill' for some people, depending on the range of material they work with, but it's worth checking out.

Regards

Thiers

##########
No worries. What has changed, I think, is not my macro but a setting on your computer: it has somehow set Option Explicit to On.

See: https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/option-explicit-statement

Basically 'proper' programmers declare ALL variables with Dim or ReDim, so for every single variable, I should have a line saying, for example, Dim doFinalBeep as Boolean. I didn't know that for years, and I now have 600-odd macros all of which have loads of variables, each of which should be Dim'med.

However, the compiler makes educated guesses about variable types and, where macros have actually needed Dim's to make them work, the compiler has told me, and I've added the relevant Dim's.

So I'm afraid that if you want to use my macros, you'll need to turn the Option Explicit command to Off, as explained in the web page quoted above.

In VBA, go to Tools -- Options -- Editor, and find "Require Variable Declaration". Turn this option off, and that, I think, should do the trick.

##########
Here is a macro that will recursively search through the specified directory and all of its sub-directories. I have put tabs between the items to make it easy to convert the resulting list to a table or drop into an Excel spreadsheet. The list comes out in inverse order of sub-directories but it can be sorted in Word or Excel.
Ah! the joys of recursive programming.

```
'  ***** These declarations must be at the top of the module *****
Dim flNames() As String
Dim flCount As Long
' ************************************************************************
Sub ScanSubFolders()
Dim FileSystem As Object
Dim mydir As String
Dim j As Long
  mydir = InputBox("Directory")

  flCount = 0
  ReDim flNames(0)

  Set FileSystem = CreateObject("Scripting.FileSystemObject")
  DoFolder FileSystem.GetFolder(mydir)

  Documents.Add
  Selection.TypeText "File name" & Chr(9) & "Folder" & Chr(9) & "Title" &
vbCrLf
  For j = 1 To flCount
    Selection.TypeText flNames(j)
  Next j

End Sub

Sub DoFolder(Folder)
Dim SubFolder
Dim aFile
Dim aRange As Range
Dim Flname As String
Dim nextdoc As Document

   For Each SubFolder In Folder.SubFolders
     DoFolder SubFolder
   Next SubFolder
   For Each aFile In Folder.Files
     Flname = aFile.Name
     If aFile.Name Like "*doc*" Then
       On Error GoTo nextFile ' Error can occur if file is already opened
by another application *****
       Set nextdoc = Documents.Open(FileName:=Folder & "\" & aFile.Name,
Visible:=False)
       Set aRange = Documents(nextdoc).Range
         With aRange.Find
           .ClearFormatting
           .Style = "Title"
           If .Execute Then
             flCount = flCount + 1
             ReDim Preserve flNames(flCount)
             flNames(flCount) = nextdoc & Chr(9) & Folder & Chr(9) &
aRange.Text
           End If
         End With
       nextdoc.Close SaveChanges:=wdDoNotSaveChanges
     End If
nextFile:
```

```
    On Error GoTo 0
  Next aFile

End Sub
```

--

##########
I haven't got Word 2019 so I can only look at your problem in 2010 and 2016.

There is a bug in Record Macro that produces in response to CTRL + ALT + PgUp the line

Selection.HomeKey Unit:=wdWindow

Only wdStory, wdColumn, wdLine and wdRow are valid units in this statement (you have to hunt around to find this, Word's Help doesn't help). This bug appears to be in all Word versions.

In Word 2010 and Word 2016 draft view, after PgUp and PgDn the display is scrolled by the number of lines on the screen and the cursor is on the same screen line in the new screen, which is what you are getting in 2019.

You will need to set the cursor to the top of the screen before PgUp or PgDn.

##########

wdFieldEmpty	-1	Empty field. Acts as a placeholder for field content that has not yet been added. A field added by pressing Ctrl+F9 in the user interface is an Empty field.
wdFieldRef	3	Ref field.
wdFieldIndexEntry	4	XE (Index Entry) field.
wdFieldFootnoteRef	5	FootnoteRef field. Not available through the Field dialog box. Inserted programmatically or interactively.
wdFieldSet	6	Set field.
wdFieldIf	7	If field.
wdFieldIndex	8	Index field.
wdFieldTOCEntry	9	TOC (Table of Contents Entry) field.
wdFieldStyleRef	10	StyleRef field.
wdFieldRefDoc	11	RD (Reference Document) field.
wdFieldSequence	12	Seq (Sequence) field.
wdFieldTOC	13	TOC (Table of Contents) field.
wdFieldInfo	14	Info field.
wdFieldTitle	15	Title field.
wdFieldSubject	16	Subject field.
wdFieldAuthor	17	Author field.
wdFieldKeyWord	18	Keywords field.
wdFieldComments	19	Comments field.
wdFieldLastSavedBy	20	LastSavedBy field.
wdFieldCreateDate	21	CreateDate field.
wdFieldSaveDate	22	SaveDate field.
wdFieldPrintDate	23	PrintDate field.
wdFieldRevisionNum	24	RevNum field.
wdFieldEditTime	25	EditTime field.
wdFieldNumPages	26	NumPages field.
wdFieldNumWords	27	NumWords field.

wdFieldNumChars 28 NumChars field.
wdFieldFileName 29 FileName field.
wdFieldTemplate 30 Template field.
wdFieldDate 31 Date field.
wdFieldTime 32 Time field.
wdFieldPage 33 Page field.
wdFieldExpression 34 = (Formula) field.
wdFieldQuote 35 Quote field.
wdFieldInclude 36 Include field. Cannot be added through the Field dialog box, but can be added interactively or through code.
wdFieldPageRef 37 PageRef field.
wdFieldAsk 38 Ask field.
wdFieldFillIn 39 Fill-In field.
wdFieldData 40 Data field.
wdFieldNext 41 Next field.
wdFieldNextIf 42 NextIf field.
wdFieldSkipIf 43 SkipIf field.
wdFieldMergeRec 44 MergeRec field.
wdFieldDDE 45 DDE field. No longer available through the Field dialog box, but supported for documents created in earlier versions of Word.
wdFieldDDEAuto 46 DDEAuto field. No longer available through the Field dialog box, but supported for documents created in earlier versions of Word.
wdFieldGlossary 47 Glossary field. No longer supported in Word.
wdFieldPrint 48 Print field.
wdFieldFormula 49 EQ (Equation) field.
wdFieldGoToButton 50 GoToButton field.
wdFieldMacroButton 51 MacroButton field.
wdFieldAutoNumOutline 52 AutoNumOut field.
wdFieldAutoNumLegal 53 AutoNumLgl field.
wdFieldAutoNum 54 AutoNum field.
wdFieldImport 55 Import field. Cannot be added through the Field dialog box, but can be added interactively or through code.
wdFieldLink 56 Link field.
wdFieldSymbol 57 Symbol field.
wdFieldEmbed 58 Embedded field.
wdFieldMergeField 59 MergeField field.
wdFieldUserName 60 UserName field.
wdFieldUserInitials 61 UserInitials field.
wdFieldUserAddress 62 UserAddress field.
wdFieldBarCode 63 BarCode field.
wdFieldDocVariable 64 DocVariable field.
wdFieldSection 65 Section field.
wdFieldSectionPages 66 SectionPages field.
wdFieldIncludePicture 67 IncludePicture field.
wdFieldIncludeText 68 IncludeText field.
wdFieldFileSize 69 FileSize field.
wdFieldFormTextInput 70 FormText field.
wdFieldFormCheckBox 71 FormCheckBox field.
wdFieldNoteRef 72 NoteRef field.
wdFieldTOA 73 TOA (Table of Authorities) field.
wdFieldTOAEntry 74 TOA (Table of Authorities Entry) field.
wdFieldMergeSeq 75 MergeSeq field.
wdFieldPrivate 77 Private field.
wdFieldDatabase 78 Database field.
wdFieldAutoText 79 AutoText field.
wdFieldCompare 80 Compare field.

wdFieldAddin 81 Add-in field. Not available through the Field dialog box. Used to store data that is hidden from the user interface.
wdFieldSubscriber 82 Macintosh only. For information about this constant, consult the language reference Help included with Microsoft Office Macintosh Edition.
wdFieldFormDropDown 83 FormDropDown field.
wdFieldAdvance 84 Advance field.
wdFieldDocProperty 85 DocProperty field.
wdFieldOCX 87 OCX field. Cannot be added through the Field dialog box, but can be added through code by using the AddOLEControl method of the Shapes collection or of the InlineShapes collection.
wdFieldHyperlink 88 Hyperlink field.
wdFieldAutoTextList 89 AutoTextList field.
wdFieldListNum 90 ListNum field.
wdFieldHTMLActiveX 91 HTMLActiveX field. Not currently supported.
wdFieldBidiOutline 92 BidiOutline field.
wdFieldAddressBlock 93 AddressBlock field.
wdFieldGreetingLine 94 GreetingLine field.
wdFieldShape 95 Shape field. Automatically created for any drawn picture.
wdFieldCitation96 Citation field.
wdFieldBibliography 97 Bibliography field.
wdFieldMergeBarcode 98 MergeBarcode field.
wdFieldDisplayBarcode99 DisplayBarcode field.
Name Value Description

##########
Name Value Description
wdFieldAddin 81 Add-in field. Not available through the Field dialog box. Used to store data that is hidden from the user interface.
wdFieldAddressBlock 93 AddressBlock field.
wdFieldAdvance 84 Advance field.
wdFieldAsk 38 Ask field.
wdFieldAuthor 17 Author field.
wdFieldAutoNum 54 AutoNum field.
wdFieldAutoNumLegal 53 AutoNumLgl field.
wdFieldAutoNumOutline 52 AutoNumOut field.
wdFieldAutoText 79 AutoText field.
wdFieldAutoTextList 89 AutoTextList field.
wdFieldBarCode 63 BarCode field.
wdFieldBidiOutline 92 BidiOutline field.
wdFieldComments 19 Comments field.
wdFieldCompare 80 Compare field.
wdFieldCreateDate 21 CreateDate field.
wdFieldData 40 Data field.
wdFieldDatabase 78 Database field.
wdFieldDate 31 Date field.
wdFieldDDE 45 DDE field. No longer available through the Field dialog box, but supported for documents created in earlier versions of Word.
wdFieldDDEAuto 46 DDEAuto field. No longer available through the Field dialog box, but supported for documents created in earlier versions of Word.
wdFieldDisplayBarcode99 DisplayBarcode field.
wdFieldDocProperty 85 DocProperty field.
wdFieldDocVariable 64 DocVariable field.
wdFieldEditTime 25 EditTime field.
wdFieldEmbed 58 Embedded field.
wdFieldEmpty -1 Empty field. Acts as a placeholder for field content that has not yet been added. A field added by pressing Ctrl+F9 in the user interface is an Empty field.
wdFieldExpression 34 = (Formula) field.
wdFieldFileName 29 FileName field.

wdFieldFileSize 69 FileSize field.
wdFieldFillIn 39 Fill-In field.
wdFieldFootnoteRef 5 FootnoteRef field. Not available through the Field dialog box. Inserted programmatically or interactively.
wdFieldFormCheckBox 71 FormCheckBox field.
wdFieldFormDropDown 83 FormDropDown field.
wdFieldFormTextInput 70 FormText field.
wdFieldFormula 49 EQ (Equation) field.
wdFieldGlossary 47 Glossary field. No longer supported in Word.
wdFieldGoToButton 50 GoToButton field.
wdFieldGreetingLine 94 GreetingLine field.
wdFieldHTMLActiveX 91 HTMLActiveX field. Not currently supported.
wdFieldHyperlink 88 Hyperlink field.
wdFieldIf 7 If field.
wdFieldImport 55 Import field. Cannot be added through the Field dialog box, but can be added interactively or through code.
wdFieldInclude 36 Include field. Cannot be added through the Field dialog box, but can be added interactively or through code.
wdFieldIncludePicture 67 IncludePicture field.
wdFieldIncludeText 68 IncludeText field.
wdFieldIndex 8 Index field.
wdFieldIndexEntry 4 XE (Index Entry) field.
wdFieldInfo 14 Info field.
wdFieldKeyWord 18 Keywords field.
wdFieldLastSavedBy 20 LastSavedBy field.
wdFieldLink 56 Link field.
wdFieldListNum 90 ListNum field.
wdFieldMacroButton 51 MacroButton field.
wdFieldMergeBarcode 98 MergeBarcode field.
wdFieldMergeField 59 MergeField field.
wdFieldMergeRec 44 MergeRec field.
wdFieldMergeSeq 75 MergeSeq field.
wdFieldNext 41 Next field.
wdFieldNextIf 42 NextIf field.
wdFieldNoteRef 72 NoteRef field.
wdFieldNumChars 28 NumChars field.
wdFieldNumPages 26 NumPages field.
wdFieldNumWords 27 NumWords field.
wdFieldOCX 87 OCX field. Cannot be added through the Field dialog box, but can be added through code by using the AddOLEControl method of the Shapes collection or of the InlineShapes collection.
wdFieldPage 33 Page field.
wdFieldPageRef 37 PageRef field.
wdFieldPrint 48 Print field.
wdFieldPrintDate 23 PrintDate field.
wdFieldPrivate 77 Private field.
wdFieldQuote 35 Quote field.
wdFieldRef 3 Ref field.
wdFieldRefDoc 11 RD (Reference Document) field.
wdFieldRevisionNum 24 RevNum field.
wdFieldSaveDate 22 SaveDate field.
wdFieldSection 65 Section field.
wdFieldSectionPages 66 SectionPages field.
wdFieldSequence 12 Seq (Sequence) field.
wdFieldSet 6 Set field.
wdFieldShape 95 Shape field. Automatically created for any drawn picture.
wdFieldSkipIf 43 SkipIf field.
wdFieldStyleRef 10 StyleRef field.

wdFieldSubject 16 Subject field.
wdFieldSubscriber 82 Macintosh only. For information about this constant, consult the language reference Help included with Microsoft Office Macintosh Edition.
wdFieldSymbol 57 Symbol field.
wdFieldTemplate 30 Template field.
wdFieldTime 32 Time field.
wdFieldTitle 15 Title field.
wdFieldTOA 73 TOA (Table of Authorities) field.
wdFieldTOAEntry 74 TOA (Table of Authorities Entry) field.
wdFieldTOC 13 TOC (Table of Contents) field.
wdFieldTOCEntry 9 TOC (Table of Contents Entry) field.
wdFieldUserAddress 62 UserAddress field.
wdFieldUserInitials 61 UserInitials field.
wdFieldUserName 60 UserName field.
wdFieldBibliography 97 Bibliography field.
wdFieldCitation 96 Citation field.

##########

=lorem()

or

=lorem(6,8) (or whatever)

Also

=rand()

or

=rand(3,7)

##########
Open files at specific places and sizes

Macro 1, open new file in middle of screen, with specific distance from the edge of the screen.

```
Sub OpenInMiddleScreen

Dialogs(wdDialogFileOpen).Show

scnHeight = Application.UsableHeight
scnWidth = Application.UsableWidth

mySideMargin = 100
myTopMargin = 50

' ActiveDocument.ActiveWindow.WindowState = wdWindowStateNormal
Application.Move Left:=mySideMargin, Top:=myTopMargin

wdth = scnWidth - 2 * mySideMargin
ht = scnHeight - 2 * myTopMargin
Application.Resize Width:=wdth, Height:=ht

End Sub
```

Macro 2, open each new file same size, but further down and right. For this, you need to read the parameters of the current window and open the new file

```
Sub OpenDownAndRight()
myJump = 50

nowWdth = Application.Width
nowHt = Application.Height
nowLeft = Application.Left
nowTop = Application.Top

Dialogs(wdDialogFileOpen).Show

newLeft = nowLeft + myJump
newTop = nowTop + myJump

Application.Move Left:=newLeft, Top:=newTop


scnHeight = Application.UsableHeight
scnWidth = Application.UsableWidth


wdth = nowWdth
rtMargin = scnWidth - newLeft - wdth

If rtMargin < 0 Then wdth = wdth + rtMargin

ht = nowHt
btmMargin = scnHeight - newTop - ht
If btmMargin < 0 Then ht = ht + btmMargin


Application.Resize Width:=wdth, Height:=ht
End Sub
```

```
#########################################################
scrollPosition = ActiveWindow.ActivePane.VerticalPercentScrolled (capture
the scroll position)
[things happen, the scroll moves]
ActiveWindow.ActivePane.VerticalPercentScrolled =  scrollPosition  (go back
to original position)

#########################################################
```

Track change viewing

```
Application.ScreenUpdating = False
Dim myView As Object
Set myView = ActiveWindow.View.MarkupMode r RevisionsFilter
If ActiveDocument.Revisions.Count > 0 Then
```

```
 If myView.Markup = wdRevisionsMarkupAll Then
    myView.Markup = wdRevisionsMarkupNone
 ElseIf myView.Markup = wdRevisionsMarkupNone Then
    myView.Markup = wdRevisionsMarkupAll
 End If
  ActiveWindow.ScrollIntoView Selection.range, True 'This keeps the cursor in view
E
```

##########

MsoShapeType Enumeration

06/14/2014

2 minutes to read

Office Developer Reference

Specifies the type of a shape or range of shapes.

Name	Value	Description
msoAutoShape	1	AutoShape.
msoCallout	2	Callout.
msoCanvas	20	Canvas.
msoChart	3	Chart.
msoComment	4	Comment.
msoDiagram	21	Diagram.
msoEmbeddedOLEObject	7	Embedded OLE object.
msoFormControl	8	Form control.
msoFreeform	5	Freeform.
msoGroup	6	Group.
msoIgxGraphic	24	SmartArt graphic
msoInk	22	Ink
msoInkComment	23	Ink comment
msoLine	9	Line
msoLinkedOLEObject	10	Linked OLE object
msoLinkedPicture	11	Linked picture
msoMedia	16	Media
msoOLEControlObject	12	OLE control object
msoPicture	13	Picture
msoPlaceholder	14	Placeholder
msoScriptAnchor	18	Script anchor
msoShapeTypeMixed	-2	Mixed shape type
msoTable	19	Table
msoTextBox	17	Text box
msoTextEffect	15	Text effect

##########

Please would you try following the instructions in the following link?

https://superuser.com/questions/988756/how-can-you-change-mac-excel-2011-to-separate-using-commas-not-semicolons

Thanks.

(I've checked my macros, and there are 174 occurrences of a comma that I'd
have to change to a semicolon.)

##########

```
Sub SymbolsInFunnyFonts()
```

```
' Doesn't work at all!


' Version 25.01.17
' Find something specific and do things to each one




myFont = "Calibri"
Set rng = ActiveDocument.Content
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = ""
  .Font.Name = myFont
  .Wrap = wdFindStop
  .Replacement.Text = ""
  .Forward = True
  .MatchWildcards = False
  .Execute
End With

myCount = 0
Do While rng.Find.Found = True
' If you want to count them...
  myCount = myCount + 1
' Note where the end of the found item is
  endNow = rng.End
  rng.Select
  Selection.Collapse wdCollapseStart
  Selection.TypeText Text:=rng.Text
  ' Be sure you're past the previous occurrence
  Set rng = ActiveDocument.range(endNow, endNow)

' Go and find the next occurrence (if there is one)
  rng.Find.Execute
Loop
MsgBox "Changed: " & myCount
End Sub

########################################################

Sub WINorMAC()
'Test the OperatingSystem

'MsgBox (Application.Version)

'MsgBox (Application.System.MacintoshName)


Selection.TypeText Text:=Application.ProductCode & vbCr
Selection.TypeText Text:=Application.Version & vbCr

Selection.TypeText Text:=Application.OperatingSystem & vbCr
```

```
Selection.TypeText Application.System.MacintoshName & vbCr
xvgd = 0
' blah = Application.System.MacintoshName
    ' If Not Application.OperatingSystem Like "*Mac*" Then
        'i am Windows

End Sub

#########################################################
Sub CommentPassiveVoice()
' Version 14.04.16
'A basic Word macro coded by Greg Maxey
Dim oGR As range
  For Each oGR In ActiveDocument.GrammaticalErrors
    oGR.GrammarChecked = 1
    ActiveDocument.Comments.Add oGR, "Passive Voice"
    oGR.Select
  Next
End Sub


#########################################################
Sub GrammarErrorNext()
  hereNow = Selection.End
  Selection.Expand wdParagraph
  Selection.Start = hereNow
  If Selection.range.GrammaticalErrors.Count > 0 Then
    Selection.range.GrammaticalErrors(1).Select
    MsgBox Selection.range.GrammaticalErrors.Type
  End If


End Sub

#########################################################


Sub NavigationPane()
  ActiveWindow.DocumentMap = True
  CommandBars("Navigation").Visible = True
End Sub


#########################################################
```

This won't get you from where you are to where you might like to be, but it's a nice simple tutorial I've found on writing macros and saving them straight into another template, with key binding. It has also taught me a bit more than I knew before. :-)

https://msdn.microsoft.com/en-us/library/office/ff604039(v=office.14).aspx

###

On error

https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/statements/on-error-statement
##

> I've tried this out now. I'm still in Word 2010, so this may be different if
> you've moved on.

> Briefly, when macros are saved in a different template to the Normal
> template, the key assignments they have are saved with them in that
> template. You need to be very well organised, however, which I'm not. In
> theory, all my macros are saved in a template called, imaginatively,
> mymacros.dotm. I make that available as a global template.

> Forgive egg-sucking lessons - I'm not sure whether you need this, but will
> say it now to save toing and froing. On the Developer tab (assuming you have
> that showing - if not go into File/Options/Customize Ribbon), click Document
> Template, then go to the lower window and click Add.. Browse to the folder
> with the templates in and choose, for instance, mymacros.dotm. This will
> then be listed in the window and you can check the box to load it. I think I
> remember that you can skip that step and have it available in all documents
> by putting it in the Word Startup folder, but I would have to research that
> again.

> The snag is that you can only edit the VBA file or save new macros in the
> separate template by opening the template file, and when you're in the
> middle of doing a job that feels like too much hassle, so I end up with some
> saved in Normal as well, and sometimes I've used the same key assignment for
> different things in the different templates. Word recognises them in a
> certain order - I think I may have sent you this way back.
> <http://wordribbon.tips.net/T011940_Finding_Default_Shortcut_Keys.html?awt_l
> =Ml_bR
> <http://wordribbon.tips.net/T011940_Finding_Default_Shortcut_Keys.html?awt_l
> =Ml_bR&awt_m=IpmhFje87wn74X> &awt_m=IpmhFje87wn74X>
>
> Fortunately I don't have a lot of these to sort out, so it won't take long,
> and it will be good discipline to sort out the key assignments. I hope that
> information helps you a bit, though.

##

Print as pdf

```
fjlskdflkdjs = ActivePrinter
  ActivePrinter = "Microsoft Print to PDF"
  Application.PrintOut FileName:="", Range:=wdPrintAllDocument, Item:= _
    wdPrintDocumentWithMarkup, Copies:=1, Pages:="", PageType:= _
    wdPrintAllPages, Collate:=True, Background:=True, PrintToFile:=False, _
    PrintZoomColumn:=0, PrintZoomRow:=0, PrintZoomPaperWidth:=0, _
    PrintZoomPaperHeight:=0
  ActivePrinter = "Canon MX490 series"
```

##

insertions and deletions.
insDels = ActiveWindow.View.ShowInsertionsAndDeletions
shwFormat = ActiveWindow.View.ShowFormatting
etc = ActiveWindow.View.ShowInkAnnotations
ActiveWindow.View.ShowInsertionsAndDeletions
ActiveWindow.View.ShowComments

##
Just to explain what @ does/doesn't do...

{1,} says "any quantity from 1 to infinity of the previous character"

whereas

@ says "any quantity from 1 to infinity of the previous character"

See the difference? No? OK then try this explanation.

{1,} is what I call "conscientious", while @ is "lazy".

If you've got, say, [0-9]@ in the *middle* of a Find, it's exactly the same, but if @ is used at the beginning or end of a Find, it works differently. So, some examples...

Text: "The box has area 146 x 120 cm"

Find1: area [0-9]@ x [0-9]@ cm will find "area 146 x 120 cm"

Find2: area [0-9]{1,} x [0-9]{1,} cm will find "area 146 x 120 cm" (exactly the same)

However...

Find3: [0-9]{1,} x [0-9]{1,} will find "146 x 120" (as many digits as possible)

Find4: [0-9]@ x [0-9]@ will find "6 x 1" (as few digits as possible)

Does that help? It's not an easy concept to explain.

If you watch this video, it'll help you "play with" wildcard F&Rs. That's the best way to learn - to do it!

Developing wildcard searches (2:09)
Speed up the process of fault-finding a wildcard search
https://youtu.be/8UCbmiiDq-c

##
Sub CreateNewFile()

```
Dim storyRange As Range
Dim oldFileName As String
  oldFileName = ActiveDocument.Name
  ActiveDocument.SaveAs "newfilename.docx"
  For Each storyRange In ActiveDocument.StoryRanges
    storyRange.Delete
  Next storyRange
  Documents.Open FileName:=oldFileName
End Sub
```

##

It is useful to also show the creation date:

```
  s = "Created in Word" & wrdVer(versionNo) & _
        "  " & ActiveDocument.BuiltInDocumentProperties("Creation
date").Value
```

##

Ken Endacott

Hello all

Here is a macro that I use to give some idea of the history of a document.
It gives the Word version in which the document was created.

```
Sub CreationVersion()

Dim versionNo As Long

Dim wrdNo As Long

Dim s As String

Dim wrdVer()

  wrdVer = Array("< 2002", "2002", "2003", "2007", "2008", "2010", "2013",
"2016", "????")

  versionNo = 8

  On Error Resume Next      ' versions below 2010 do not have
compatibilityMode

    versionNo = ActiveDocument.CompatibilityMode - 9

  On Error GoTo 0

  wrdNo = Int(Val(Application.Version)) - 9

  If wrdNo < 0 Or wrdNo > 8 Then wrdNo = 0
```

```
  s = "Created in Word" & wrdVer(versionNo)

  If wrdNo <> versionNo Then s = s & vbCrLf & "Will be edited in
compatibility mode"

  MsgBox s

End Sub



#######################################################
Sub temp1()
  With Application.Dialogs(wdDialogStyleManagement)
    SendKeys "%OCalibri%Z11%L1 cm{ENTER}"
    .Show
  End With
End Sub

#######################################################

Hidden text

If Selection.Start = Selection.End Then
  doAll = True
  Set rng = ActiveDocument.Content
Else
  doAll = False
  Set rng = Selection.range.Duplicate
End If

myTrack = ActiveDocument.TrackRevisions
ActiveDocument.TrackRevisions = False
Application.ScreenUpdating = False


If rng.Words(1).Font.Hidden = True Then
  hideText = False
Else
  hideText = True
End If
If doAll = True Then
  For Each par In rng.Paragraphs
    par.range.Font.Hidden = hideText
  Next par
Else
  For Each wd In rng.Words
    wd.Font.Hidden = hideText
  Next wd
End If
ActiveDocument.TrackRevisions = myTrack
Application.ScreenUpdating = True
Beep
Exit Sub

#######################################################
Sub gohari_comment()
```

```
'
' gohari_comment Macro
'
'
    Selection.Comments.Add range:=Selection.range

End Sub
Sub gohari_query()
'
' gohari_query Macro
'
'
Set rng = Selection.range.Duplicate
Application.Run MacroName:="gohari_comment"
Selection.TypeText Text:="??"
ActiveWindow.ActivePane.Close
rng.Select
Selection.Collapse wdCollapseEnd
End Sub
```

##

I hope this message finds you well. As you might recall, I had a question a few weeks ago about using macros with track changes and the big problem this entails--namely that VBA can't ignore deleted text, which often triggers an error. Well, I think I've found a solution, and I thought I'd share it, in case it's helpful to you or anyone you work with. The following macro solves the problem by producing a copy of the document, running whichever macros the editor wants on the copy, and then performing a "compare documents" between the copy and the original. The result is exactly as if track changes had been turned on during the editing macros. It's a pretty simple solution, and so far it works nicely for me.

Best,
Pablo

```
Sub TrackedEdits()

Application.ScreenUpdating = False

Dim Doc1 As Document, _
Doc2 As Document, Doc1Name As String, _
Doc2Name As String

Set Doc1 = ActiveDocument
Doc1Name = Doc1.Name
Set Doc2 = Documents.Add(YOUR TEMPLATE PATH HERE)
Doc2Name = Doc2.Name
Doc2.Range.FormattedText = Doc1.Range.FormattedText
```

```
'Call your editing macros here
'For some reason, Doc2 needs to be reactivated with each macro listed
'Otherwise, it sometimes reverts to working on Doc1

Doc2.Activate
'First macro
Doc2.Activate
'Second macro
Doc2.Activate
'etc.

Application.CompareDocuments _
OriginalDocument:=Documents(Doc1Name), _
RevisedDocument:=Documents(Doc2Name), _
Destination:=wdCompareDestinationRevised, _
Granularity:=wdGranularityWordLevel, _
CompareFormatting:=False, _
CompareCaseChanges:=True, _
CompareWhitespace:=True

Application.ScreenUpdating = True

End Sub
```

##########

KeysBoundTo Property

Returns a KeysBoundTo object that represents all the key combinations assigned to the specified item.

expression.KeysBoundTo(KeyCategory, Command, CommandParameter)

expression Optional. An expression that returns one of the objects in the Applies To list.

WdKeyCategory

WdKeyCategory can be one of these WdKeyCategory constants.

wdKeyCategoryAutoText
wdKeyCategoryCommand
wdKeyCategoryDisable
wdKeyCategoryFont
wdKeyCategoryMacro
wdKeyCategoryNil
wdKeyCategoryPrefix
wdKeyCategoryStyle
wdKeyCategorySymbol

Command Required String. The name of the command.

CommandParameter Optional Variant. Additional text, if any, required for the command specified by Command.

For more information, see the "Remarks" section in the Add method for the KeyBindings object.

Example

This example displays all the key combinations assigned to the FileOpen command in the template attached to the active document.

```
Dim kbLoop As KeyBinding
Dim strOutput As String

CustomizationContext = ActiveDocument.AttachedTemplate

For Each kbLoop In _
      KeysBoundTo(KeyCategory:=wdKeyCategoryCommand, _
      Command:="FileOpen")
   strOutput = strOutput & kbLoop.KeyString & vbCr
Next kbLoop
```

MsgBox strOutput

This example removes all key assignments from Macro1 in the Normal template.

```
Dim aKey As KeyBinding

CustomizationContext = NormalTemplate
For Each aKey In _
      KeysBoundTo(KeyCategory:=wdKeyCategoryMacro, _
      Command:="Macro1")
   aKey.Disable
Next aKey
```

##########

Microsoft Office Word sets and automatically updates a number of reserved bookmarks. You can use these predefined bookmarks just as you use bookmarks that you place in documents, except that you do not have to set them and they are not listed on the Go To tab in the Find and Replace dialog box.

You can use predefined bookmarks with the Bookmarks property. The following example sets the bookmark named "currpara" to the location marked by the predefined bookmark named "\Para".

```
ActiveDocument.Bookmarks("\Para").Copy "currpara"
```

The following table describes the predefined bookmarks available in Word.

Bookmark	Description
\Sel	Current selection or the insertion point.
\PrevSel1	Most recent selection where editing occurred; going to this bookmark is equivalent to running the GoBack method once.
\PrevSel2	Second most recent selection where editing occurred; going to this bookmark is equivalent to running the GoBack method twice.
\StartOfSel	Start of the current selection.
\EndOfSel	End of the current selection.
\Line	Current line or the first line of the current selection. If the insertion point is at the end of a line that is not the last line in the paragraph, the bookmark includes the entire next line.
\Char	Current character, which is the character following the insertion point if there is no selection, or the first character of the selection.
\Para	Current paragraph, which is the paragraph containing the insertion point or, if more than one paragraph is selected, the first paragraph of the selection. Note that if the insertion point or selection is in the last paragraph of the document, the "\Para" bookmark does not include the paragraph mark.
\Section	Current section, including the break at the end of the section, if any. The current section contains the insertion point or selection. If the selection contains more than one section, the "\Section" bookmark is the first section in the selection.
\Doc	Entire contents of the active document, with the exception of the final paragraph mark.
\Page	Current page, including the break at the end of the page, if any. The current page contains the insertion point. If the current selection contains more than one page, the "\Page" bookmark is the first page of the selection. Note that if the insertion point or selection is in the last page of the document, the "\Page" bookmark does not include the final paragraph mark.
\StartOfDoc	Beginning of the document.
\EndOfDoc	End of the document.

\Cell Current cell in a table, which is the cell containing the insertion point. If one or more cells of a table are included in the current selection, the "\Cell" bookmark is the first cell in the selection.
\Table Current table, which is the table containing the insertion point or selection. If the selection includes more than one table, the "\Table" bookmark is the entire first table of the selection, even if the entire table is not selected.
\HeadingLevel The heading that contains the insertion point or selection, plus any subordinate headings and text. If the current selection is body text, the "\HeadingLevel" bookmark includes the preceding heading, plus any headings and text subordinate to that heading.

##########

https://docs.microsoft.com/en-us/dotnet/visual-basic/language-reference/operators/like-operator

##########
Make Word wait...

Word suspends execution of this statement until the document is fully loaded hence the following statements operate on the whole document.

```
Set rng = ActiveDocument.GoTo(What:=wdGoToBookmark, Name:="\StartOfDoc")
```

##########

Delay loading a file
use a bookmark for end of document.

I have seen a similar situation when a VBA statement was executed before the document was fully loaded.
Set rng = ActiveDocument.Content
may give only give part of the document. To test, step through statement by statement using F8 pausing for a couple of seconds before the above statement. If this gives a different result to executing the macro with F5 then you have the loading problem.
The solution is to place the following statement after the document activation statement:
Set rng = ActiveDocument.GoTo(What:=wdGoToBookmark, Name:="\EndOfDoc")
Word suspends execution of this statement until the document is fully loaded hence the following statements operate on the whole document.

##########

Gary Smith Forum regular
Re: Automation of references - (macros?)
20 May 2017 at 10:42 Posts: 82
This should italicise book and journal titles (I'd do a test on a copy):

```
Sub ContControlFormat()

Dim AllControls As Long

AllControls = ActiveDocument.ContentControls.Count
```

```
x = 1

While x < AllControls
ActiveDocument.ContentControls.Item(x).Range.Select
If ActiveDocument.ContentControls.Item(x).Tag = "book-title" Then
Selection.Font.Italic = True
ElseIf ActiveDocument.ContentControls.Item(x).Tag = "journal-title" Then
Selection.Font.Italic = True
End If

x = x + 1

Wend

End Sub
```
I think moving the content around is probably possible where it's in blocks, but complex. For moving items within blocks (e.g. inverting names) I'd probably use a character style or highlight to isolate the text, then use a wildcard on just that character style to invert the text.

Edited 1 time(s). Last edit at 20/05/2017 at 10:44 by Gary Smith.

##########

Axxxx Bxxxx Cxxxx Dxxxx Exxxx Fxxxx Gxxxx Hxxxx Ixxxx Jxxxx Kxxxx Lxxxx
Mxxxx Nxxxx Oxxxx Pxxxx Qxxxx Rxxxx Sxxxx Txxxx Uxxxx Vxxxx Wxxxx Xxxxx
Yxxxx Zxxxx Äxxxx

##########
The VBA jargon term for the named values for constants is "enumeration".

This page: https://msdn.microsoft.com/en-us/library/office/dn353221.aspx

Lists enumerations for everything in Word 2013 and above.

This one for Word 2010: https://msdn.microsoft.com/en-us/library/office/ff846868(v=office.14).aspx

This one for Word 2007: https://msdn.microsoft.com/en-us/library/ee426857(v=office.12).aspx or this one: https://msdn.microsoft.com/en-us/library/ee426857(v=office.13).aspx

And this one for 2003: https://msdn.microsoft.com/en-us/library/office/aa211923(v=office.11).aspx

Couldn't dig up the one for Word 2011.

##########

Re: Changing default language in comment boxes (Word) New

I'd turn track changes off for this.

Put your cursor in a comment box (not sure if you have to do this but I do
just to be sure) and then click on the styles option, then manage styles.

Then find 'comment text' in the list and highlight that. Then click on modify.

Then in the bottom right, click format, then language, then choose your language.

Then click all the OKs required to get back out and that should do it.

If I've not explained it very well, I think you can do a Google search to find blogs on exactly this topic.

##########

To do an F&R from here to the end, use:

```
myFind = Selection
Selection.MoveStart , -1
Selection.End = ActiveDocument.Content.End

Options.DefaultHighlightColorIndex = wdGray25
Set rng = Selection.range.Duplicate
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = myFind
  .Wrap = False
  .Font.Name = "Times New Roman"
  .Replacement.Text = "<zczc>^&</zczc>"
  .Replacement.Highlight = True
  .Forward = True
  .MatchCase = True
  .MatchWildcards = False
  .Execute Replace:=wdReplaceAll
End With
```

##########

```
Sub AlphaCiteGroup()

Dim group() As String
Dim start As String
Dim arraySize As Long
Dim firstcount As Long
Dim secondcount As Long
Dim tempStr  As String
Dim arrayTrack As Long

start = Selection.Text

firstcount = Len(start)
tempStr = Replace(start, ";", "")
secondcount = Len(tempStr)

arraySize = (firstcount - secondcount) + 1
```

```
ReDim group(arraySize)
group = Split(start, "; ", -1) <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<

WordBasic.SortArray group()  <<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<<
arrayTrack = 0
tempStr = ""

While arrayTrack < arraySize
tempStr = tempStr & group(arrayTrack) & "; "
arrayTrack = arrayTrack + 1
Wend

tempStr = Trim(tempStr)
firstcount = Len(tempStr)
tempStr = Left(tempStr, firstcount - 1)

Selection.TypeText tempStr

End Sub
#####################################################
> To transfer the keystrokes you'll need to use the ShortcutOrganizer -
> available at http://www.addbalance.com/word/download.htm#ChrisWoodman -
> Version 13.7. Dec2015
> <http://www.addbalance.com/word/download/KeyboardShortcutOrganizer2010.zip>

#####################################################

Sub TaskPaneCloser()
Dim CB As CommandBar
sdfasd = CommandBars.Count
For Each CB In CommandBars
  ' On Error Resume Next 'we dont care about CB's that wont respond
   ' If CB.Visible Then CB.Visible = False
   dskfhdkas = CB.Context
   dskfdkas = CB.Position
   Debug.Print CB.Name
Next
End Sub

#####################################################

Sub Test_1()
myText = CurDir()
MsgBox myText
End Sub

Sub Test_2()
Dim myText As String
myText = CurDir()
MsgBox myText
End Sub

Sub Test_3()
nowFile = ActiveDocument.Name
MsgBox nowFile
End Sub
```

```
Sub Test_4()
Dialogs(wdDialogFileOpen).Show
nowFile = ActiveDocument.Name
MsgBox nowFile
End Sub

Sub Test_5()
Dialogs(wdDialogFileOpen).Show
dirPath = ActiveDocument.Path
MsgBox dirPath
End Sub

Sub Test_7()
Dialogs(wdDialogFileOpen).Show
ActiveDocument.Close
dirPath = ActiveDocument.Path
myFile = Dir(dirPath & Application.PathSeparator)
MsgBox myFile
myFile = Dir(dirPath & Application.PathSeparator)
MsgBox myFile
myFile = Dir()
MsgBox myFile

End Sub

Sub sdhfk()

myText = CurDir()
nowFile = ActiveDocument.Name
MsgBox myText
Dialogs(wdDialogFileOpen).Show
dirPath = ActiveDocument.Path
r = r & "dirPath = " & ActiveDocument.Path & CR2
Debug.Print Right(r, 75)
If ActiveDocument.Name <> nowFile Then ActiveDocument.Close
r = r & "CurDir(dirPath) " & CurDir(dirPath) & CR2
ChDir dirPath
r = r & "After ChDir dirPath, CurDir(dirPath) =" & CurDir(dirPath) & CR2
' Read the names of all the files in this directory
myFile = Dir(dirPath & Application.PathSeparator)
r = r & "Dir(dirPath & Application.PathSeparator) gives: " & myFile & CR2
myFile = Dir()
r = r & "The first Dir() gives: " & myFile & CR2
myFile = Dir()
r = r & "The first Dir() gives: " & myFile & CR2
Documents.Add
Selection.TypeText Text:=r
End Sub
```

##########

To find the enumeration of commandbars by name and internal number, google "Word commandbar enumeration".

For more general information google "Word 2013 object model".

##
In case this is useful to anyone else, this macro (found on internet, not invented by me) changes the shading colour of table cells from the colour of the cell the cursor is in to grey. Change the RGB value for a different colour.

```
Sub ChangeCellFills()
Dim myRange As Range
myColor = Selection.Range.Cells(1).Shading.BackgroundPatternColor
For Each myTable In ActiveDocument.Tables
Set myRange = myTable.Range
For Each myCell In myRange.Cells
If myCell.Shading.BackgroundPatternColor = myColor Then
myCell.Shading.BackgroundPatternColor = RGB(201, 201, 201)
End If
Next myCell
Next myTable
End Sub
```

And this version lets you pick a replacement colour:

```
Sub ChangeCellFills()

If Not Selection.Information(wdWithInTable) Then
MsgBox "Place the cursor in a table cell."
Exit Sub
end if

Dim myRange As Range
CurrentColor = Selection.Range.Cells(1).Shading.BackgroundPatternColor

Set myDialog = Dialogs(wdDialogFormatBordersAndShading)
myDialog.DefaultTab = wdDialogFormatBordersAndShadingTabShading
myDialog.Display
NewColor = myDialog.BackgroundRGB

For Each myTable In ActiveDocument.Tables
Set myRange = myTable.Range
For Each myCell In myRange.Cells
If myCell.Shading.BackgroundPatternColor = CurrentColor Then
myCell.Shading.BackgroundPatternColor = NewColor
End If
Next myCell
Next myTable
End Sub
```

##

Weird fonts in Asian documents. Try changing the language/font in the Themes section, to the left on the Page Layout tab.

##

> Error I am getting:Run-time error '4198':Command failed

> Debug highlights this:Selection.PasteSpecial DataType:=wdPasteText

Yes, sorry about this, but it is a known problem, though not one for which I have a consistent solution.

If you look at those of my programs that cause this error, you will probably see two lines, something like this:

```
' Selection.PasteAndFormat (wdFormatPlainText)
Selection.PasteSpecial DataType:=wdPasteText
```

These are two different commands, each of which is supposed to do exactly the same thing - paste a text-only version of what is currently held in the clipboard (i.e. the text that you - or the macro - selected, before doing a Ctrl-C).

Over the years, I've found that on some computers both work, on others the one works and the other doesn't, and on yet others vice versa. What a pain!

So what I suggest you try is to switch over which one is enabled, and which one is "commented out" by having an apostrophe in front of it, i.e. change it to:

```
Selection.PasteAndFormat (wdFormatPlainText)
' Selection.PasteSpecial DataType:=wdPasteText
```

Let me know if the problem persists. If it does, then the only thing I can think is that it's being caused by that program you mentioned.

##

```
Sub ScratchMacro()
'A basic Word macro coded by Greg Maxey
Dim oGR As Range
  For Each oGR In ActiveDocument.GrammaticalErrors
    oGR.GrammarChecked = 1
    ActiveDocument.Comments.Add oGR, "Passive Voice"
  Next
lbl_Exit:
  Exit Sub
End Sub
```

##

> I just downloaded your Free Macros FR Edit and Proofing Pack. I get the
> following compile error :
>
> "Variable not defined"

I'm afraid that I'm a very lazy programmer. :-(

I don't declare *any* of my variables.

Presumably, on your computer someone must have set VBA to detect undeclared variables. So you will have to turn this option off:

In VBA, go to Tools -- Options -- Editor, and find "Require Variable Declaration". Turn this option off, and that, I think, should do the trick.

Extra info:
~~~~~~~~~~~

I checked the options menu, the "Require Variable Declaration" was turned OFF, but on further exploration, another macro that I had pasted from the internet contained "Option Explicit". Deleted it, and the macros work fine.

##

One thing I have wondered about is a macro to check for quotation marks and especially when the Bible has several levels at times. I have succeeded in developing a routine that is useful for us. This macro depends upon firstly changing all normal quotes to open curly quotes if they are present. The rest will work itself out.
The only thing it cannot do is determine what happens when there is a non coupling open and closing quote. In this case the paragraph is highlighted.

```
Sub CheckQuotes()
   Dim sRaw As String
   Dim jSmart As Integer
   Dim J As Long

   Selection.HomeKey Unit:=wdStory
   Application.ScreenUpdating = False
   Selection.Find.ClearFormatting
   With Selection.Find
      .Text = """
      .Replacement.Text = """
      .Forward = True
      .Wrap = wdFindStop
      .Format = True
      .MatchCase = False
      .MatchWholeWord = False
      .MatchWildcards = False
      .MatchSoundsLike = False
      .MatchAllWordForms = False
   End With
   Selection.Find.Execute

   While Selection.Find.Found
      Selection.MoveDown Unit:=wdParagraph, Count:=1, Extend:=wdExtend
      Selection.MoveLeft Unit:=wdCharacter, Count:=1, Extend:=wdExtend
      sRaw = Selection.Text
      jSmart = 0
      For J = 1 To Len(sRaw)
'Opening quotes smart single/double and normal single/double quote
         If Mid(sRaw, J, 1) = Chr(145) Or Mid(sRaw, J, 1) = Chr(147) Or Mid(sRaw, J, 1) = Chr(34) Or
(Mid(sRaw, J, 1) = Chr(39)) Then
            If J > 1 Then
               If Mid(sRaw, J - 1, 1) = " " Then
                  jSmart = jSmart + 1
                  If jSmart = 2 Or jSmart = 4 Or jSmart = 6 Then
'Make every other opening quote a double quote
```
~~~~~~~~~~~

```
                    Mid(sRaw, J, 1) = Chr(147)
                  Else
                    Mid(sRaw, J, 1) = Chr(145)
                  End If
                End If
              Else
'Opening quote at beginning of paragraph
                jSmart = jSmart + 1
                If jSmart = 2 Or jSmart = 4 Or jSmart = 6 Then
                  Mid(sRaw, J, 1) = Chr(147)
                End If
              End If
            End If
'Closing quotes smart single/double and normal single/double
            If Mid(sRaw, J, 1) = Chr(146) Or Mid(sRaw, J, 1) = Chr(148) Or Mid(sRaw, J, 1) = Chr(34) Or
(Mid(sRaw, J, 1) = Chr(39)) Then
              If Mid(sRaw, J + 1, 1) = " " Or Mid(sRaw, J + 1, 1) = "." Or Mid(sRaw, J + 1, 1) = ")" Or Mid(sRaw, J +
1, 1) = "," Then
                jSmart = jSmart - 1
                If jSmart = 1 Or jSmart = 3 Or jSmart = 5 Then
                  Mid(sRaw, J, 1) = Chr(148)
                Else
                  Mid(sRaw, J, 1) = Chr(146)
                End If
              End If
            End If

        Next
        If jSmart <> 0 Then
          jSmart = 0
          Selection.Range.HighlightColorIndex = wdYellow
        End If
        Selection.TypeText sRaw
        Selection.MoveDown Unit:=wdParagraph, Count:=1, Extend:=wdExtend
        If jSmart <> 0 Then
          Selection.Range.HighlightColorIndex = wdYellow
        End If
        Selection.Collapse Direction:=wdCollapseEnd
        Selection.Find.Execute
    Wend
' Move cursor to start of document
    Selection.HomeKey Unit:=wdStory
    Application.ScreenUpdating = True
End Sub
```

##
Date: Sat, 14 Nov 2015 09:06:38 +1100
Subject: Progress bars (WAS 'Microsoft Word - Not Responding)

After looking at the example mentioned by Steve (
http://www.excel-easy.com/vba/examples/progress-indicator.html) I decided
to have a go at creating a Word-oriented version, based on the same code.
Some of you may be interested to see the result.

The only Word-specific part is in the procedure the author calls 'code'. I
replaced this with a macro that performs a simple and pointless task on a
long Word document - making the font colour red, one word at a time. Though

it's definitely pointless, it does provide a demonstration of the working, because you can see it happening at the same time as you see the progress bar.

It should be fairly easy to adapt for use with other macros that use a For loop at the top level - which is the reason for posting it. In most cases you'll probably want to introduce an Application.ScreenUpdating = False line to speed it up and not force the user to watch each step.

i followed the construction pretty much as described for the original example, apart from the 'code' procedure. You can see the final result at http://1drv.ms/1Y9LYsI (you'll need to download it to run the macro). Below is the new version of the 'code' procedure and the accompanying 'progress' procedure:

```
Sub code()
   Dim i As Integer, n As Integer
   Dim d As Integer
   Dim pctCompl As Integer, pct_incr As Integer
   Dim StartTime As Variant

   With ActiveDocument
      pct_incr = 10          ' size of increment in progress bar
      n = .Words.Count
      d = Int(n * pct_incr / 100)
      For i = 1 To n
         .Words(i).Font.ColorIndex = wdRed

         If i Mod d = 0 Then
            pctCompl = pct_incr * Int(i / d)
            progress pctCompl
         End If
      Next i
   End With
   UserForm1.Hide
End Sub

Sub progress(pctCompl As Integer)
   With UserForm1
      .Text.Caption = pctCompl & "% complete"
      .Bar.Width = pctCompl * 2
   End With

   DoEvents
End Sub
```

I have kept the button to launch the macro, but i can't see why you would want to launch such a macro from a button in the document in real life.

Regards

Howard

http://listserv.liv.ac.uk/archives/word-pc.html

##

Multiple views
Set thisDoc = ActiveDocument.ActiveWindow

thisWindow = Windows(1).Caption

Windows(thisWindow).Activate

Seems to me there's a bit of a conceptual problem here. When I think about it, I realise that it's really a bit strange that you can activate a document when really it's only windows for which activation makes sense. I don't often work with multiple windows for the same doc, so haven't thought about this before. 'ActiveDocument' is
?, as far as I can see,?
really a shorthand for ActiveWindow.Document.
? So the assignment Set thisDoc = ActiveDocument.ActiveWindow is really Set thisDoc = ActiveWindow.Document.ActiveWindow, which is exactly the same as Set thisDoc = ActiveWindow. (I may be pedantic, but this notation really grates for me as I would only use the variable thisDoc for a document, which would be declared using Dim thisDoc as Document. And so that assignment would cause a type mismatch. ?I'd use thisWin in place of thisDoc and declare Dim thisWin as Window, then if you need the document use thisWin.Document.)

I don't how what logic VBA uses to decide which window to activate when you have a statement Doc.Activate and the document has multiple windows. Seems pretty much random to me when I try it out.

##

Set rng = ActiveDocument.range(Selection.Start, Selection.End)

is equivalent to

Set rng = Selection.range

##

As an alternative to:

Selection.PasteAndFormat (wdFormatPlainText)
' Selection.PasteSpecial DataType:=wdPasteText

you could try using

Selection.Paste
ActiveDocument.Content.Style = ActiveDocument.Styles(wdStyleNormal)

```
ActiveDocument.Content.Font.Reset
ActiveDocument.Content.HighlightColorIndex = wdNoHighlight
```

##

Howard Silcock

I had to see if I could implement the ideas I mentioned in my last post.

It turned out that with the approach I described, the table problem doesn't arise, because you can't cross a cell boundary without encountering a "non-allowable" character.

Here's the macro I came up with. I followed Jacques's suggestion: if the selection is a rectangular one (not just an insertion point), then the link's attached to that selection, but if it's just an insertion point, the macro finds the URL or email address and attaches the link to that.

I think it works when the text is in a table or at the start or end of the document. I haven't found a case where it doesn't work yet!

```
Sub URLlink3()
   Dim i As Long, InitPos As Long
   Dim LinkRng As Range
   Dim LinkRngStartPos As Long, LinkRngEndPos As Long
   Dim AddressText As String, DisplayText As String
   Dim URLchars As String

   URLchars = "[A-Za-z0-9._/(?)%=&(#)~@:-]"
   With ActiveDocument
      If Selection.Type = wdSelectionIP Then
         InitPos = .Range(0, Selection.Start).Characters.Count

         i = InitPos
         Do While i > 0
            If .Characters(i).Text Like URLchars Then
               i = i - 1
            Else
               Exit Do
            End If
         Loop
         LinkRngStartPos = .Characters(i + 1).Start

         i = InitPos + 1
         Do While i < .Characters.Count + 1
            If .Characters(i).Text Like URLchars Then
               i = i + 1
            Else
               Exit Do
            End If
         Loop
         LinkRngEndPos = .Characters(i).Start

         Set LinkRng = .Range(LinkRngStartPos, LinkRngEndPos)

      ElseIf Selection.Type = wdSelectionNormal Then
```

```
        Set LinkRng = Selection.Range

      Else
        MsgBox "Please make another selection"
        Exit Sub
      End If

      DisplayText = LinkRng.Text
      AddressText = IIf(InStr(DisplayText, "@") > 0, "mailto:", "") & _
        DisplayText

      ActiveDocument.Hyperlinks.Add Anchor:=LinkRng,
Address:=AddressText, _
        TextToDisplay:=DisplayText
   End With
End Sub
```

(As a matter of style, I prefer not to move the cursor around in a macro
unless that's necessary.)

Howard

```
##########################################################
  ChangeFileOpenDirectory _
    "E:\Users\User\Desktop\HardDisc4CATS\MyFiles2\WIP\aaa\"

  Selection.InsertFile FileName:="Abstract_2_+Engl_150115_PB.docx", Range:= _
    "", ConfirmConversions:=False, Link:=False, Attachment:=False


##########################################################
On Error GoTo ReportIt
'WordBasic.PreviousChangeOrComment
Restart:
thisRev = ActiveDocument.Range(0, _
    Selection.Range.Revisions(1).Range.End).Revisions.Count
Do
  i = thisRev
  Set myRev = ActiveDocument.Revisions(i)
  myRev.Range.Select
  myType = myRev.FormatDescription
  thisRevString = InputBox(myType, "Accept this change?", Trim(Str(i)))
  If thisRevString = "" Then Exit Sub
  thisRev = Val(thisRevString)
Loop Until i = thisRev

theEnd = ActiveDocument.Content.End
i = 0
For Each rev In ActiveDocument.Range.Revisions
  Set rng = rev.Range
  thisType = rev.FormatDescription
  If thisType = myType Then
  i = i + 1
  rng.Revisions.AcceptAll
  End If
  StatusBar = "Accepting track changes...  " _
```

```
      & Str(theEnd - rng.End)
Next rev
StatusBar = "Accepted track changes:  " & Str(i)
Exit Sub

ReportIt:
' There are no files open at all
If Err.Number = 5941 Then
  WordBasic.NextChangeOrComment
  Beep
  Resume Restart
Else
' Display Word's error message
  MsgBox Err.Description, vbExclamation, "Error from Word"
End If
```

##

Editing comment balloons characteristics

"Comment Text" style isn't the only place you'll have to make adjustments
to get the results you want. You'll also need to change the Bubble Text style.

A blog post by my colleague Liz Dexter will
guide you through the necessary steps:

http://libroediting.com/2012/10/19/customising-comment-boxes-in-word/

The trick turned out to be that the document was created in Japan.
Continued Google searching gave me the answer at
http://www.internationalskeptics.com/forums/showthread.php?t=235700.
I added Japanese as an editing language (but didn't install the
corresponding proofing tools which would cost $ and isn't necessary). Now,
I just have to right-click on one of the comments that shows up as
double-spaced and uncheck the box next to "snap to grid when document grid
is defined" (this box will not be displayed without Japanese being
available as an editing language). Once that one comment balloon is showing
up single-spaced (the default and my preference), I can highlight it,
right-click on the Comment Text style, and choose "update comment text to
match selection." That fixes all of the comment balloons in the document!

##

Ariela Marks wants to know how to do a mass table delete in Word.

Hi, Ariela.
        One S&R command won't do the job, but you can quickly add macro
to do it:

```
   Do While ActiveDocument.Tables.Count >= 1
      Selection.GoTo What:=wdGoToTable, Which:=wdGoToNext, Count:=1,
Name:=""
```

```
    Selection.Tables(1).Select
    Selection.Tables(1).Delete
Loop
```

##
> < Making Word 2010 Work for You

My little book would be helpful to the extent of establishing a bedrock understanding of styles and templates - but I think Susan needs (also needs?) something more detailed. I don't, for example, get into the interplay of styles and themes that Jessica mentions.

The advice for Mac users boils down to _Run an emulator and get a PC version of Word._ Mac Word just isn't the same product, and while a lot of the observations in the book can be applied on a Mac, I abandoned the effort made in the first book to keep track of the differences and address them wherever they're relevant.

--

##

The macro book on the web!

http://oreilly.com/catalog/writewordmacro/chapter/ch17.html

##
Page Layout/Hyphenation and click 'None' (in Word 2013), and check that 'automatically hyphenate document' is not ticked.

##

(Version 1)

Ah yes, this is a standard query from continental Europe and South Africa. It's to do with a thing called the "list separator" that is set in your operating system, but then used by Word.

You just need to change the list separator from a semicolon to a comma. Here are my standard instructions:

The "list separator" used within Word needs to be comma, not semicolon.

However, this is not a *Word* option, rather it's an operating system option.

So, on Windows 7, 8.1 and 10, it is in the Control Panel under "Clock Language and Region" and then "Region" and then "Additional Settings" (which is a button near the bottom of the Region window). In Aditional Settings, the fourth from the bottom is "List separator". Change it to a comma and click OK.

And then you need to restart the computer to force Word to take account of the fact that you've changed the list separator.

Hope that gets it sorted.

Out of interest, can I ask how you found out about my macros?

##

(Version 2)

This is a problem with computers set up for Continental Europe and South Africa, where they use the decimal comma instead of the UK's decimal point. If you're interested, I'll give you the full explanation (or just jump to the solution, at the end, my "standard instructions"...

The "pattern match" error is caused by the use of semicolons instead of commas in wildcard searches:

For example, if I wanted to find "one or more consecutive vowels", I could use a wildcard search for:

[aeiou]{1,}

and it would find, "ea", "oo", "ie", "ooooo" etc.

However, if you try that in the normal Find window, I think it will produce that "pattern match" error.

But if you change the Find to:

[aeiou]{1;}

with a semicolon, it should now work on your computer.

But my macros use comma, not semicolons, so you either have to change all my macros, or change the option in your computer.

Here are my standard instructions:

The "list separator" used by Word needs to be comma, not semicolon.

However, this is not a *Word* option, rather it's a Windows operating system option.

So, on Windows 7, 8.1 and 10, it is in the Control Panel under "Clock Language and Region" and then "Region" and then "Additional settings" (which is a button near the bottom of the Region window). In Aditional settings, the fourth from the bottom is "List separator". Change it to a comma and click OK.

Then you will need to do a Restart.

##
iz/is spelling

http://blog.oxforddictionaries.com/2011/03/ize-or-ise/

```
#########################################################
On Error GoTo ReportIt
'WordBasic.PreviousChangeOrComment
Restart:
thisRev = ActiveDocument.Range(0, _
    Selection.Range.Revisions(1).Range.End).Revisions.Count
Do
  i = thisRev
  Set myRev = ActiveDocument.Revisions(i)
  myRev.Range.Select
  myType = myRev.FormatDescription
  thisRevString = InputBox(myType, "Accept this change?", Trim(Str(i)))
  If thisRevString = "" Then Exit Sub
  thisRev = Val(thisRevString)
Loop Until i = thisRev

theEnd = ActiveDocument.Content.End
i = 0
For Each rev In ActiveDocument.Range.Revisions
  Set rng = rev.Range
  thisType = rev.FormatDescription
  If thisType = myType Then
  i = i + 1
  rng.Revisions.AcceptAll
  End If
  StatusBar = "Accepting track changes...  " _
      & Str(theEnd - rng.End)
Next rev
StatusBar = "Accepted track changes:  " & Str(i)
Exit Sub

ReportIt:
' There are no files open at all
If Err.Number = 5941 Then
  WordBasic.NextChangeOrComment
  Beep
  Resume Restart
Else
' Display Word's error message
  MsgBox Err.Description, vbExclamation, "Error from Word"
End If


#########################################################

comment stuff


thisComm = 0
If Selection.Information(wdInCommentPane) = True Then
  thisComm = Selection.Comments(1).Index
End If


totComments = ActiveDocument.Comments.Count
If totComments = 0 Then Beep: Exit Sub
```

```
thisComm = 0
If Selection.Information(wdInCommentPane) = True Then
  thisComm = Selection.Comments(1).Index
End If

If thisComm = 0 Then
  WordBasic.GoToNextComment
'  WordBasic.NextComment
  thisComm = 0
  If Selection.Information(wdInCommentPane) = True Then
    thisComm = Selection.Comments(1).Index
  End If

  If Selection.Start = hereNow Then
    Beep
  End If
End If

If thisComm = totComments Then
  Beep
  Exit Sub
Else
  WordBasic.GoToNextComment
'  WordBasic.NextComment
End If

hereNow = Selection.Start

totComments = ActiveDocument.Comments.Count
If totComments = 0 Then Beep: Exit Sub

thisComm = 0
If Selection.Information(wdInCommentPane) = True Then
  thisComm = Selection.Comments(1).Index
End If

If thisComm = 0 Then
'  WordBasic.GoToPreviousComment
  WordBasic.PreviousComment
  If Selection.Start = hereNow Then
    Beep
    Exit Sub
  End If
End If

If thisComm = 1 Then
  Beep
  Exit Sub
Else
  WordBasic.GoToPreviousComment
'  WordBasic.GoToPreviousComment
End If


#####################################################
> Have you tried Reveal Formatting (Shift F1)?
```

```
######################################################
If Selection.Bookmarks.Exists("\EndOfDoc") = True Then
  MsgBox "END OF DOCUMENT REACHED!"
End If
```

##

wildcard F&R.

The other resource is in the public domain, and is very useful:

http://word.mvps.org/faqs/general/usingwildcards.htm
##

From: http://word.mvps.org/faqs/macrosvba/WordBasicCommands.htm

FileNameInfo$()

This is another very useful function for which there is no direct VBA equivalent. FileNameInfo allows you to get just the filename or a fully qualified pathname from a filename given to it. The nearest equivalent in VBA are the Name, FullName and Path properties of the Document object.

FileNameInfo is different in that you don't need to have the document open.

The syntax is

x = WordBasic.FilenameInfo$(Filename$, FileType)

where Filename is the name of the file, and FileType is a number which defines the part of the filename you want to return:

1 - the full pathname, e.g. C:\My Documents\My File.doc"
2 - the filename only, if the file is in the current folder, otherwise the full pathname
3 - the filename only
4 - the filename without the extension
5 - the path without the filename
6 - the UNC pathname

One case where FileNameInfo$ is very useful is to get the pathname of a file which has just been selected by the user in the FileOpen dialog. The following code returns the full pathname of a file selected by the user.

```
With Dialogs(wdDialogFileOpen)
   If .Display Then
      MsgBox WordBasic.FilenameInfo$(.Name, 1)
   Else
      MsgBox "No file selected"
   End If
End With
```

##

pattern match not valid - use ';' instead of ',' in

I am aware that quite a lot of my macros won't work on continental machines because of a single option that is set up in the operating system (not in Word itself), and it relates to the continental use of the decimal comma. Sounds odd, I know, but we've had this many times, and it's easy to fix.

The trouble is that on the continent, you would normally use a decimal comma rather than a decimal full stop (i.e. four and a half is written as "4,5", and not "4.5").

So, within the operating system, Microsoft have chosen to use a semicolon, rather than a comma, as the so-called "list separator".

So, in wildcard find and replace, which a number of my macros use, instead of looking for, say "[A-Z]{2,4}" (between two and four capital letters) you would have to use "[A-Z]{2;4}".

Now, you could either change all of the lines of all of the macros from comma to semicolon (DocAlyse alone has 30 of them!) or you could tell the computer to use comma, not semicolon, as the list separator.

This can be set up via the Windows control panel:

In Windows 7 you can find the list separator via Control Panel -- Region & Language -- Format -- Additional Settings.

In Windows 8 and 10, it's Control Panel -- Clock, Language and Region -- Region -- Additional settings... -- List separator.

So once you've found it, if you change the list separator from ";" to "," - then do a computer Restart - and all should be well.

##

This is how to switch between headers and footers and the main text:

```
Sub Sub_FTR_0()
'
ActiveDocument.ActiveWindow.ActivePane.View.SeekView = wdSeekCurrentPageFooter

For i = 1 To ActiveDocument.Sections.Count
 'REM: INSERT Code from RECORD MACRO recorded when editing one Footer correctly
    Selection. [[xxx]], etc.

If i = ActiveDocument.Sections.Count Then GoTo Line1

    ActiveDocument.ActiveWindow.ActivePane.View.NextHeaderFooter

Line1:
Next

    ActiveWindow.ActivePane.View.SeekView = wdSeekMainDocument

End Sub
```

WdSeekView can be one of these WdSeekView constants:

wdSeekCurrentPageFooter

wdSeekCurrentPageHeader

wdSeekEndnotes

wdSeekEvenPagesFooter

wdSeekEvenPagesHeader

wdSeekFirstPageFooter

wdSeekFirstPageHeader

wdSeekFootnotes

wdSeekMainDocument

wdSeekPrimaryFooter

wdSeekPrimaryHeader

##########
This macro highlights all additions and deletions in the main body of the
document:

```
Sub HighLightThosePeskyRevisions()
Dim aRevision As Revision
For Each aRevision In ActiveDocument.Revisions
   If (aRevision.Type = wdRevisionDelete) Or (aRevision.Type =
wdRevisionInsert) Then
       aRevision.Range.HighlightColorIndex = wdYellow
   End If
  Next aRevision
End Sub
```

##########

> Yes, this procedure works OK. I can't see how to turn it into a
> macro, so I'll just have to go through the tedious keystrokes, but
> it's much better than trying to delete all headers and footers
> manually, as it were. Thanks both.

OK, I've had a ferret around, found the general principle
and then honed it down to the following macro:

```
Sub GetRidOfHeadersAndFooters()
' Version 11.10.13
' Delete all headers, footers and watermarks
```

```
ActiveDocument.DocumentInspectors(2).Fix 0, ""

End Sub
```

##########
Neat little macro I just found on LinkedIn (with thanks to Kathy Evans, who posted it):

```
Sub PrintCurrentPage()
Application.PrintOut Range:=wdPrintCurrentPage
End Sub
```

It does exactly what it says in the sub line, and I often want to do that.

##########

In 2010 you can specify how changes are marked, but text highlight is not one of the choices. You can make them all a specific color rather than different colors by author, and you can make them bold.

This macro highlights all additions and deletions in the main body of the document:

```
Sub HighLightThosePeskyRevisions()
Dim aRevision As Revision
For Each aRevision In ActiveDocument.Revisions
   If (aRevision.Type = wdRevisionDelete) Or (aRevision.Type = wdRevisionInsert) Then
      aRevision.Range.HighlightColorIndex = wdYellow
   End If
 Next aRevision
End Sub
```

##########
Paul:

I've hit the "list separator" problem in two places. First in statements like your Find, and second in TOC fields.

I have code that determines the separator in use and executes the correctly formatted statement.

```
Select Case Application.International(wdListSeparator)
   Case ","
      With objRange.Find
         .MatchWildcards = True
         .Text = " {2,}"
         .Replacement.Text = " "
         .Execute Replace:=wdReplaceAll
      End With
   Case ";"
      With objRange.Find
         .MatchWildcards = True
         .Text = " {2;}"
         .Replacement.Text = " "
         .Execute Replace:=wdReplaceAll
      End With
End Select
```

I don't know how to compensate for the TOC (and perhaps other fields?) stumbling on the separator. You can change the separator on the fly using Control Panel > Region and Language.

I've also run into problems referring to the built-in styles by their English names. There's a language-independent enumeration of the built-in styles you can use in code, Word.WdBuiltinStyle. So referring to wdStyleHeading1 in your code will get the right style no matter which language and spelling is used in the GUI.

Most times, however, I expect and use a specific set of aliases.

Bear

Why not simplify that code with this:

```
strLS=Application.International(wdListSeparator)
With objRange.Find
  .MatchWildcards = True
  .Text = " {2" & strLS & "}"
  .Replacement.Text = " "
  .Execute Replace:=wdReplaceAll
End With
```

Jacques Raubenheimer
Lecturer 5/8: Biostatistics
Lektor 5/8: Biostatistiek
PO Box / Posbus 339, Bloemfontein 9300, Republic of South Africa / Republiek van Suid-Afrika
051 4013115

```
#########################################################
 ActiveDocument.undo


#########################################################
Sub SquareNestCurly()
' Version 17.08.12
' Find any [] nested inside a [] and make them into {}

Selection.HomeKey Unit:=wdStory
insideSquares = False
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "[\[\]]"
    .Replacement.Text = ""
    .Forward = True
    .Wrap = False
    .MatchWildcards = True
End With
Selection.Find.Execute
Do While Selection.Find.Found
  If Selection = "[" Then
    If insideSquares = True Then
      Selection.TypeText Text:="{"
      Selection.Collapse wdCollapseEnd
      Selection.Find.Execute
      Selection.TypeText Text:="}"
```

```
    Else
      insideSquares = True
    End If
  Else
    insideSquares = False
  End If
  Selection.Collapse wdCollapseEnd
  Selection.Find.Execute
Loop
End Sub



##########################################################

Sub JumpScroll2()
Dim OrigSel As Range

Set OrigSel = Selection.Range
Selection.EndKey Unit:=wdStory
OrigSel.Select
ActiveDocument.ActiveWindow.SmallScroll down:=1
End Sub

##########################################################

Sub EmailFormatter()
' Version 08.07.13
' Format paragraphs in an email

Set rng = ActiveDocument.Content
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "^p^p"
  .Wrap = wdFindContinue
  .Replacement.Text = "zczc"
  .MatchWildcards = False
  .MatchWholeWord = False
  .MatchSoundsLike = False
  .Execute Replace:=wdReplaceAll
End With
With rng.Find
  .Text = "^p"
  .Replacement.Text = " "
  .Execute Replace:=wdReplaceAll
End With
With rng.Find
  .Text = "zczc"
  .Replacement.Text = "^p^p"
  .Execute Replace:=wdReplaceAll
End With

End Sub

##########################################################
```

Word has a peculiar dual nature when it comes to language settings. I suspect that you need to go into Find and Replace and change everything to US English there, too:

Find What: no text, no formatting
Replace With: no text, Language: English (U.S.)

Probably a good idea to go to the Track Changes options and turn off format tracking, first....

##########

I guess the macro fails when you first start Word? It does work when creating a new document or opening an existing document? In that case, you may work around it by modifying the code so that it adds a delay. Something like this:

```
Sub AutoOpen()
Call SetDesiredView
End Sub

Sub AutoNew()
Call SetDesiredView
End Sub

Sub SetDesiredView()

Application.OnTime Now + TimeValue("00:00:05"), "View2"

Exit Sub
End Sub

Sub View2()
On Error GoTo errhandler
ActiveWindow.View.Type = 3
With ActiveWindow.View.Zoom
   .PageColumns = 1
   .Percentage = 100
End With
Exit Sub
errhandler:
Exit Sub
End Sub
```

In the above macros, the delay is 5 seconds (as specified in the TimeValue function), but it may work with less time than that.
Stefan Blom, Microsoft Word MVP
##########
Try this:

```
Sub RemovePersonalInfo()
   ActiveDocument.RemovePersonalInformation = True
End Sub
```

##########

Building on Jessica's solution, I had an idea of how you could set things up so that you can double-click on the initial bookmarked text and go to the first occurrence of a corresponding REF field, then you could double-click on that occurrence and go to the next occurrence, and so on.

The main trick is to use MACROBUTTON fields, which launch a macro when you double-click on them. You also need to assign a bookmark to each of the REF fields so you can tell the macro where to jump to. (I can't see any way of avoiding using macros, which is unfortunate as it means you have to save the document as a .docm. Or maybe you could store them in an add-in template?)

If you assign bookmark names systematically, you can use the same macro in every case.

Suppose your first bookmarked text is Text1, which is assigned a bookmark - say John. Then your text might look like this

Text1 blah blah Text1 blah blah Text1 blah blah Text1

but if you view the field codes it would look like this

Text1 blah blah { John } blah blah { John } blah blah { John }

(Oh, by the way, I didn't write { REF John } as it's optional to include the REF in almost all cases - you can just put the bookmark inside the field markers.)

My idea involves the following steps:

1. Replace the initial Text1 with the field
{ MACROBUTTON GoNext Text1 }, where GoNext is a macro to be defined later. When the field codes are turned off, this still reads as Text1, but now when you double-click the text it runs the macro GoNext. Make sure the bookmark John is now assigned to the range of this field.

2. Replace each of the fields { John } except the last with the field
{ MACROBUTTON GoNext { John }}
where you now have the field { John } nested inside the MACROBUTTON field. Again when you turn off the field codes it will just display Text1.

3. How you want to handle the last { John } depends what you want to happen when you reach the last reference field. I defined another macro NoMore that displays a message that you've reached the last one. So then you'd replace the final { John } with
{ MACROBUTTON NoMore { John }}. Alternatively you could again use the GoNext but incorporate an error trap that would display a message when it's tried to go to a non-existent bookmark.

4. Now select each of the MACROBUTTON fields in turn and insert a bookmark, using the names John_01, John_02, etc. (Using two digits allows you to have more than 10 reference fields if you need to.)

Now it remains to define the macro GoNext (and NoMore if you want it).

Sub GoNext()

```
   Dim MyName As String, NextName As String
   Dim CurNum As Integer

   If Selection.Range.Bookmarks.Count > 0 Then
      ' Retrieve the bookmark name you assigned
      MyName = Selection.Range.Bookmarks(1).Name
      ' If it ends in _nn increment the terminating number
      If Right(MyName, 3) Like "_[0-9][0-9]" Then
         CurNum = CInt(Right(MyName, 2))
         NextName = Left(MyName, Len(MyName) - 2) & Format(CurNum + 1,
"00")
      Else
         NextName = MyName & "_01"
      End If
      ActiveDocument.Bookmarks(NextName).Range.Select
   Else
      Msgbox "No bookmark defined for this field"
   End If
End Sub

Sub NoMore()
   MsgBox "This is the last one"
End Sub
```

I have tried this out, but can't claim to have really tested it.

Howard

> Here is a macro that will jump to each cross-ref to the right bookmark
> and select it. Not well tested.
>
> A refinement would be to let the user enter the bookmark name.
>
> A hot refinement would be to find the first field in the selection, and if
> it is a cross-reference select the cross-reference and then move on click
> to the next ref in the document to that bookmark. If it isn't a
> cross-reference check the next field in the selection and so on.
>
> - Jessica
>
>
> Sub JumpByBkMkRefs()
> Dim afield As Field
> Dim bkmkName As String
> ' replace "ll" in the next line with the name of your bookmark
> bkmkName = "ll"
> For Each afield In ActiveDocument.Fields
> If afield.Type = wdFieldRef Then
> If InStr(1, afield.code, bkmkName, vbTextCompare) > 0 Then
> afield.Select
> ' The line below shows the page number of the reference. Comment it
> back in if you like.
> ' MsgBox Selection.Range.Information(wdActiveEndAdjustedPageNumber)

```
>    End If
>  End If
> Next afield
> End Sub


#########################################################

myColour1 = wdTurquoise
myColour2 = wdYellow
thisColour = myColour1

Set rng = ActiveDocument.Content
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = " p."
  .Font.StrikeThrough = False
  .Wrap = wdFindContinue
  .Replacement.Text = " pzFPz"
  .Forward = True
  .MatchWildcards = False
  .MatchWholeWord = False
  .MatchSoundsLike = False
  .Execute Replace:=wdReplaceAll
End With
With rng.Find
  .Text = " pp."
  .Replacement.Text = " ppzFPz"
  .Execute Replace:=wdReplaceAll
End With
With rng.Find
  .Text = " trans."
  .Replacement.Text = " transzFPz"
  .Execute Replace:=wdReplaceAll
End With
With rng.Find
  .Text = " vol."
  .Replacement.Text = " volzFPz"
  .Execute Replace:=wdReplaceAll
End With
With rng.Find
  .Text = " Co."
  .Replacement.Text = " CozFPz"
  .Execute Replace:=wdReplaceAll
End With
With rng.Find
  .Text = " No."
  .Replacement.Text = " NozFPz"
  .Execute Replace:=wdReplaceAll
End With
With rng.Find
  .Text = " ed."
  .Replacement.Text = " edzFPz"
  .Execute Replace:=wdReplaceAll
End With
With rng.Find
```

```
  .Text = " no."
  .Replacement.Text = " nozFPz"
  .Execute Replace:=wdReplaceAll
End With



With Selection.Find
  .ClearFormatting
  .Highlight = False
  .Font.StrikeThrough = False
  .Replacement.ClearFormatting
  .Text = "[12][0-9]{3}[!0-9]"
  .Replacement.Text = ""
  .Forward = True
  .Wrap = False
  .MatchWildcards = True
  .Execute
End With
Do While Selection.Find.Found = True
  Selection.Sentences(1).Select
  Selection.MoveEnd wdCharacter, -2
  If thisColour = myColour1 Then
    thisColour = myColour2
  Else
    thisColour = myColour1
  End If
  Selection.Range.HighlightColorIndex = thisColour
  Selection.Collapse wdCollapseEnd

' Go and find the next occurence (if there is one)
  Selection.Find.Execute
Loop

With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "zFPz"
  .Replacement.Text = "."
  .Execute Replace:=wdReplaceAll
End With
Exit Sub


With rng.Find
  .Text = " ."
  .Replacement.Text = " zFPz"
  .Execute Replace:=wdReplaceAll
End With

With rng.Find
  .Text = " ."
  .Replacement.Text = " zFPz"
  .Execute Replace:=wdReplaceAll
End With

With rng.Find
```

```
  .Text = " ."
  .Replacement.Text = " zFPz"
  .Execute Replace:=wdReplaceAll
End With
```

##
Fro: Samantha Hartburn

> Are you aware of code such as the following? It will display a file
> selection dialog, and return the path and name of the selected file.
>
```
>    Dim sInitialFileName As String
>    Dim sPath As String
>
>    sInitialFileName = "C:\FREdit\" 'The dialog will automatically navigate to this location when it is displayed
>
>    'Ask the user to select a .doc file
>    With Application.FileDialog(msoFileDialogFilePicker)
>       .InitialFileName = sInitialFileName
>       .Filters.Clear
>       .Filters.Add "Word Documents", "*.doc", 1 'You can change this to whatever type of file you want to show
>       .AllowMultiSelect = False
>       .Show
>       If .SelectedItems.Count > 0 Then
>          sPath = .SelectedItems(1) 'sPath will now be populated with the path and name of the selected file
>       Else
>          MsgBox "File selection cancelled", vbInformation, Title:="Process Cancelled"
>       End If
>    End With
```
>
> It would mean that you don't need to hardcode a load of document
> names, and you won't need to change the macro if you add a new FRedit
> list.Might be of help in the future, but if you already know about it
> then just ignore me!

##

```
Sub Test()
' Version 28.11.12
' How many the's on average?

  Set rng = ActiveDocument.Content
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = "\<*\>"
    .Replacement.Text = ""
    .MatchWildcards = True
    .MatchWholeWord = False
    .MatchSoundsLike = False
    .Execute Replace:=wdReplaceAll
  End With
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = "..."
```

```
    .Replacement.Text = "."
    .MatchWildcards = False
    .Execute Replace:=wdReplaceAll
  End With
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = ")"
    .Replacement.Text = ""
    .MatchWildcards = False
    .Execute Replace:=wdReplaceAll
  End With
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = "("
    .Replacement.Text = ""
    .MatchWildcards = False
    .Execute Replace:=wdReplaceAll
  End With
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = """"
    .Replacement.Text = " "
    .MatchWildcards = False
    .Execute Replace:=wdReplaceAll
  End With
totWords = 0

For Each sn In ActiveDocument.Sentences
  mySnt = sn
'           sn.Select
  sentWords = sn.Words.Count
  If sentWords > 40 Then
    For Each wd In sn.Words
      If LCase(Trim(wd)) = "the" Then theCount = theCount + 1
    Next wd
    totWords = totWords + sentWords
    StatusBar = (theCount * 100 / totWords)
  End If
Next sn
MsgBox (theCount * 100 / totWords)
End Sub
```

##

Splitting a comma-separated list?

You could try the Mark 2 version. This prompts you for the IDs whose records you want deleted - you type them in the input box, separated by commas.

```
Sub RemoveSpecifiedRowsMk2()
    Dim tbl As Table
    Dim r As Row
    Dim i As Integer
    Dim InputStr As String
```

```
    Dim IDs As Variant

    InputStr = InputBox(prompt:="Type the IDs for the records you wish to
delete, " & _
              "separated by commas", Title:="Remove specified
rows")
    If InputStr = "" Then Exit Sub

    IDs = Split(InputStr, ",")

    For Each tbl In ActiveDocument.Tables
        For Each r In tbl.Rows
            For i = 0 To UBound(IDs)
                If InStr(r.Cells(1).Range.Text, IDs(i)) > 0 Then
                    r.Delete
                    Exit For
                End If
            Next
        Next
    Next
End Sub



#########################################################


Sub SpellAlyseOld()
' Version 30.04.12
' Spellcheck system

checkForeign1 = False
foreignLanguage1 = wdEnglishUS

checkForeign2 = False
foreignLanguage2 = wdFrench

compareProperNouns = True
doCountProperNouns = True

makeFReditList = False


' General options
myFile = "zzSwitchList"

minLengthProper = 4
minLengthSpell = 3

properNounColour = wdNoHighlight
properNounColour = wdGray25

mainColour = wdBrightGreen

foreign1Colour = wdTurquoise
foreign2Colour = wdPink
```

```
' Check file for alternative variable settings
Set mainDoc = ActiveDocument
For Each myWnd In Windows
  thisFileName = myWnd.Document.Name
  thisFileName = Replace(thisFileName, ".docx", "")
  thisFileName = Replace(thisFileName, ".doc", "")
  If thisFileName = myFile Then
    myWnd.Activate
    allText = ActiveDocument.Content

    myVariable = "checkForeign1 = "
    myPos = InStr(allText, myVariable)
    If myPos > 0 Then
      If Mid(allText, myPos + Len(myVariable), 4) = "True" Then
        checkForeign1 = True
      Else
        checkForeign1 = False
      End If
    End If

    myVariable = "foreignLanguage1 = "
    myPos = InStr(allText, myVariable)
    If myPos > 0 Then
      myData = Mid(allText, myPos + Len(myVariable))
      endPos = InStr(myData, Chr(13)) - 1
      If endPos >= 0 Then myData = Left(myData, endPos)

      Select Case myData
        Case "wdEnglishUS": foreignLanguage1 = wdEnglishUS
        Case "wdEnglishUK": foreignLanguage1 = wdEnglishUK
        Case "wdFrench": foreignLanguage1 = wdFrench
        Case "wdGerman": foreignLanguage1 = wdGerman
        Case Else: MsgBox "Unknown language. Please contact Paul Bev."
      End Select

    End If

    myVariable = "checkForeign2 = "
    myPos = InStr(allText, myVariable)
    If myPos > 0 Then
      If Mid(allText, myPos + Len(myVariable), 4) = "True" Then
        checkForeign2 = True
      Else
        checkForeign2 = False
      End If
    End If

    myVariable = "foreignLanguage2 = "
    myPos = InStr(allText, myVariable)
    If myPos > 0 Then
      myData = Mid(allText, myPos + Len(myVariable))
      endPos = InStr(myData, Chr(13)) - 1
      If endPos >= 0 Then myData = Left(myData, endPos)
      Select Case myData
        Case "wdEnglishUS": foreignLanguage2 = wdEnglishUS
        Case "wdEnglishUK": foreignLanguage2 = wdEnglishUK
        Case "wdFrench": foreignLanguage2 = wdFrench
```

```
        Case "wdGerman": foreignLanguage2 = wdGerman
        Case Else: MsgBox "Unknown language. Please contact Paul Bev."
                Exit Sub
      End Select
    End If

    myVariable = "compareProperNouns = "
    myPos = InStr(allText, myVariable)
    If myPos > 0 Then
      If Mid(allText, myPos + Len(myVariable), 4) = "True" Then
        compareProperNouns = True
      Else
        compareProperNouns = False
      End If
    End If

    myVariable = "doCountProperNouns = "
    myPos = InStr(allText, myVariable)
    If myPos > 0 Then
      If Mid(allText, myPos + Len(myVariable), 4) = "True" Then
        doCountProperNouns = True
      Else
        doCountProperNouns = False
      End If
    End If

    myVariable = "makeFReditList = "
    myPos = InStr(allText, myVariable)
    If myPos > 0 Then
      If Mid(allText, myPos + Len(myVariable), 4) = "True" Then
        makeFReditList = True
      Else
        makeFReditList = False
      End If
    End If
  End If
Next myWnd

' Start of main program
mainDoc.Activate
If Selection.LanguageID = wdEnglishUK Then
  mainLanguage = wdEnglishUK: myLang = "UK"
Else
  mainLanguage = wdEnglishUS: myLang = "US"
End If

If compareProperNouns = True Then myPro = "(Proper noun list: YES)" _
    Else: myPro = "(Proper noun list: NO)"
CR2 = vbCrLf & vbCrLf
myResponse = MsgBox("Main language = " & myLang & CR2 & myPro, _
      vbQuestion + vbYesNoCancel, "SpellAlyse")
If myResponse <> vbYes Then Exit Sub

' To measure the time taken
timeStart = Timer

' Check that tracking is off!
```

```
ActiveDocument.TrackRevisions = False
allPropNouns = vbCrLf & vbCrLf

' Blank off all apostrophe-s
Set rng = ActiveDocument.Range
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = ChrW(8217) & "s"
  .MatchWildcards = False
  .MatchCase = True
  .Replacement.Text = " zczc"
  .Execute Replace:=wdReplaceAll
End With

' Spellcheck the endnotes
myjump = 100
If ActiveDocument.Endnotes.Count > 0 Then
  Set rng = ActiveDocument.StoryRanges(wdEndnotesStory)
  countWds = 0
  For Each wd In rng.Words
    If countWds Mod myjump = 1 Then StatusBar = "Checking words in endnotes: " & Str(Int(countWds / myjump)
* myjump)
    countWds = countWds + 1
    If Len(Trim(wd)) >= minLengthSpell Then
      If Application.CheckSpelling(wd, MainDictionary:=Languages(mainLanguage).NameLocal) = False _
          And Trim(wd) <> "zczc" And LCase(wd) <> UCase(wd) Then
        wd.HighlightColorIndex = mainColour
        If checkForeign2 = True Then
          If Application.CheckSpelling(wd, MainDictionary:=Languages(foreignLanguage2).NameLocal) _
              = True Then wd.HighlightColorIndex = foreign2Colour
        End If
        If checkForeign1 = True Then
          If Application.CheckSpelling(wd, MainDictionary:=Languages(foreignLanguage1).NameLocal) _
              = True Then wd.HighlightColorIndex = foreign1Colour
        End If
      ' But might it be a proper noun?
        maybeProper = False
      ' If the first letter is a cap, it may be a PN
        If Asc(wd) > 64 And Asc(wd) < 91 Then
          maybeProper = True
        ' But if the second is uppercase, it's probably not
          If Asc(Mid(wd, 2, 1)) < 96 Then maybeProper = False
        ' Check if it's a sentence-start word
          thisWord = Trim(wd)
          wdStart = wd.Start
          wdEnd = wd.End
        ' Check the previous char
          wd.Start = wd.Start - 1
        ' If it's the start of a new line or after a tab,
        ' it may not be a PN
          If Asc(wd) = 13 Or Asc(wd) = 9 Then maybeProper = False
        ' check the three chars before that
          wd.Start = wd.Start - 1
          wd.End = wd.Start + 1
          minusTwo = wd
          wd.Start = wd.Start - 1
```

```
          wd.End = wd.Start + 1
          minusThree = wd
          wd.Start = wd.Start - 1
          wd.End = wd.Start + 1
          minusFour = wd
        ' If minusTwo is not a letter
          If LCase(minusTwo) = UCase(minusTwo) Then
            maybeProper = False
            If minusTwo = "," Or minusTwo = ";" Then maybeProper = True
          ' Check for, e.g. P. Funnyname
            If minusTwo = "." And UCase(minusThree) = minusThree _
                And minusFour = " " Then maybeProper = True
          ' Check for, e.g. P Funnyname
            If UCase(minusTwo) = minusTwo And minusThree = " " Then _
                maybeProper = True
          End If
          If maybeProper = True And Len(thisWord) >= minLengthProper Then
            Set rng2 = ActiveDocument.StoryRanges(wdEndnotesStory)
            rng2.Start = wdStart
            rng2.End = wdEnd
            rng2.HighlightColorIndex = properNounColour
            allPropNouns = allPropNouns & thisWord & vbCrLf
          End If
        End If
      End If
    End If
  Next wd
End If

' Spellcheck the footnotes
myjump = 100
If ActiveDocument.Footnotes.Count > 0 Then
  Set rng = ActiveDocument.StoryRanges(wdFootnotesStory)
  countWds = 0
  For Each wd In rng.Words
    If countWds Mod myjump = 1 Then StatusBar = "Checking words in footnotes: " & Str(Int(countWds / myjump)
* myjump)
    countWds = countWds + 1
    If Len(Trim(wd)) >= minLengthSpell Then
      If Application.CheckSpelling(wd, MainDictionary:=Languages(mainLanguage).NameLocal) = False _
          And Trim(wd) <> "zczc" And LCase(wd) <> UCase(wd) Then
        wd.HighlightColorIndex = mainColour
        If checkForeign2 = True Then
          If Application.CheckSpelling(wd, MainDictionary:=Languages(foreignLanguage2).NameLocal) _
              = True Then wd.HighlightColorIndex = foreign2Colour
        End If
        If checkForeign1 = True Then
          If Application.CheckSpelling(wd, MainDictionary:=Languages(foreignLanguage1).NameLocal) _
              = True Then wd.HighlightColorIndex = foreign1Colour
        End If
      ' But might it be a proper noun?
        maybeProper = False
      ' If the first letter is a cap, it may be a PN
        If Asc(wd) > 64 And Asc(wd) < 91 Then
          maybeProper = True
        ' But if the second is uppercase, it's probably not
          If Asc(Mid(wd, 2, 1)) < 96 Then maybeProper = False
```

```
        ' Check if it's a sentence-start word
          thisWord = Trim(wd)
          wdStart = wd.Start
          wdEnd = wd.End
        ' Check the previous char
          wd.Start = wd.Start - 1
        ' If it's the start of a new line or after a tab,
        ' it may not be a PN
          If Asc(wd) = 13 Or Asc(wd) = 9 Then maybeProper = False
        ' check the three chars before that
          wd.Start = wd.Start - 1
          wd.End = wd.Start + 1
          minusTwo = wd
          wd.Start = wd.Start - 1
          wd.End = wd.Start + 1
          minusThree = wd
          wd.Start = wd.Start - 1
          wd.End = wd.Start + 1
          minusFour = wd
        ' If minusTwo is not a letter
          If LCase(minusTwo) = UCase(minusTwo) Then
            maybeProper = False
            If minusTwo = "," Or minusTwo = ";" Then maybeProper = True
          ' Check for, e.g. P. Funnyname
            If minusTwo = "." And UCase(minusThree) = minusThree _
                And minusFour = " " Then maybeProper = True
          ' Check for, e.g. P Funnyname
            If UCase(minusTwo) = minusTwo And minusThree = " " Then _
                maybeProper = True
          End If
          If maybeProper = True And Len(thisWord) >= minLengthProper Then
            Set rng2 = ActiveDocument.StoryRanges(wdFootnotesStory)
            rng2.Start = wdStart
            rng2.End = wdEnd
            rng2.HighlightColorIndex = properNounColour
            allPropNouns = allPropNouns & thisWord & vbCrLf
          End If
        End If
      End If
    End If
  Next wd
End If

' Spellcheck the main text
i = ActiveDocument.Words.Count
For Each wd In ActiveDocument.Words
  If Len(Trim(wd)) >= minLengthSpell Then
    If Application.CheckSpelling(wd, MainDictionary:=Languages(mainLanguage).NameLocal) = False _
        And Trim(wd) <> "zczc" And LCase(wd) <> UCase(wd) Then
      wd.HighlightColorIndex = mainColour
      If checkForeign2 = True Then
        If Application.CheckSpelling(wd, MainDictionary:=Languages(foreignLanguage2).NameLocal) _
            = True Then wd.HighlightColorIndex = foreign2Colour
      End If
      If checkForeign1 = True Then
        If Application.CheckSpelling(wd, MainDictionary:=Languages(foreignLanguage1).NameLocal) _
            = True Then wd.HighlightColorIndex = foreign1Colour
```

```
      End If
    ' But might it be a proper noun?
      maybeProper = False
    ' If the first letter is a cap, it may be a PN
      If Asc(wd) > 64 And Asc(wd) < 91 Then
        maybeProper = True
      ' But if the second is uppercase, it's probably not
        If Asc(Mid(wd, 2, 1)) < 91 Then maybeProper = False
      ' Check if it's a sentence-start word
        thisWord = Trim(wd)
        wdStart = wd.Start
        wdEnd = wd.End
      ' Check the previous char
        wd.Start = wd.Start - 1
      ' If it's the start of a new line or after a tab,
      ' it may not be a PN
        If Asc(wd) = 13 Or Asc(wd) = 9 Then maybeProper = False
      ' check the three chars before that
        wd.Start = wd.Start - 1
        wd.End = wd.Start + 1
        minusTwo = wd
        wd.Start = wd.Start - 1
        wd.End = wd.Start + 1
        minusThree = wd
        wd.Start = wd.Start - 1
        wd.End = wd.Start + 1
        minusFour = wd
      ' If minusTwo is not a letter
        If LCase(minusTwo) = UCase(minusTwo) Then
          maybeProper = False
          If minusTwo = "," Or minusTwo = ";" Then maybeProper = True
        ' Check for, e.g. P. Funnyname
          If minusTwo = "." And UCase(minusThree) = minusThree _
              And minusFour = " " Then maybeProper = True
        ' Check for, e.g. P Funnyname
          If UCase(minusTwo) = minusTwo And minusThree = " " Then _
              maybeProper = True
        End If
        If maybeProper = True And Len(thisWord) >= minLengthProper Then
          Selection.Start = wdStart
          Selection.End = wdEnd
          Selection.Range.HighlightColorIndex = properNounColour
          allPropNouns = allPropNouns & thisWord & vbCrLf
        End If
      End If
    End If
  End If
  i = i - 1
  If i Mod 100 = 0 Then StatusBar = "Spellchecking. To go: " & Str(i)
Next wd

Selection.HomeKey Unit:=wdStory
If compareProperNouns = True Then
  ' Now compare the proper nouns
  Documents.Add
  Selection.TypeText Text:=allPropNouns
```

```
' Remove superfluous apostrophes
Set rng = ActiveDocument.Range
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = ChrW(8217) & "^p"
  .MatchWildcards = False
  .Replacement.Text = "^p"
  .Execute Replace:=wdReplaceAll
End With

' Sort the proper nouns and remove duplicates
Selection.Sort ExcludeHeader:=False, CaseSensitive:=True, FieldNumber:="Paragraphs"
For i = ActiveDocument.Paragraphs.Count To 2 Step -1
  Set rng1 = ActiveDocument.Paragraphs(i).Range
  Set rng2 = ActiveDocument.Paragraphs(i - 1).Range
  If rng1 = rng2 Then rng1.Delete
Next i

' Collect all proper nouns into an array
totWords = ActiveDocument.Words.Count
ReDim myWords(totWords)
j = 0
For i = 1 To totWords
  wrd = Trim(ActiveDocument.Words(i))
  gotFunnies = False
  If Len(wrd) < minLengthProper Then
    gotFunnies = True
  Else
    For k = 1 To Len(wrd)
      myChar = Mid(wrd, k, 1)
      If Asc(myChar) > 127 Then gotFunnies = True: Exit For
      If AscW(myChar) > 127 Then gotFunnies = True: Exit For
    Next k
  End If
  If gotFunnies = False Then
    j = j + 1
    myWords(j) = wrd
  End If
Next
totWords = j
ActiveDocument.Close SaveChanges:=False
mainDoc.Activate
Selection.HomeKey Unit:=wdStory

' Now collect the similar words
duplicateList = ""
For i = 1 To totWords
  gotOne = False
  StatusBar = "Comparing proper nouns: " & Str(totWords - i)
  Set rng = ActiveDocument.Content
  ' Check if this word has a sounds-like word anywhere
  mwd = myWords(i)
  Do
    With rng.Find
      .ClearFormatting
      .Replacement.ClearFormatting
```

```
        .Text = mwd
        .Wrap = False
        .Replacement.Text = ""
        .MatchSoundsLike = True
        .Forward = True
        .MatchWildcards = False
        .Execute
      End With
    ' But make sure that the sounds-like word isn't all
    ' lower case, and isn't a very short word.
      If rng.Find.Found = True Then
        fnd = rng
        rng.Start = rng.End
        rng.End = rng.End + 1
        abit = rng
        If Len(abit) > 0 Then aa = Asc(abit)

      ' But we don't want it if it's the same word
        If fnd = mwd Then GoTo no
      ' or if it's already on the list
        If InStr(duplicateList, fnd & "!") > 0 Then GoTo no
      ' or is it's all lcase
        If LCase(fnd) = fnd Then GoTo no
      ' or if it's a very short word
        If Len(fnd) < minLengthProper Then GoTo no
      ' or if it's a very different length
        If Abs(Len(fnd) - Len(mwd)) > 3 Then GoTo no
      ' or if it's part of a longer word
        If Asc(aa) < 123 And Asc(aa) > 96 Then GoTo no
        gotOne = True
        duplicateList = duplicateList & fnd & "!"
no:
        stopNow = False
      Else
        stopNow = True
      End If
      rng.Start = rng.End
    Loop Until stopNow = True
    If gotOne = True Then duplicateList = duplicateList & mwd & "!" & vbCrLf
  Next i
  StatusBar = ""
  Selection.Find.MatchSoundsLike = False

  ' Put the list in the text file
  Selection.EndKey Unit:=wdStory
  Selection.TypeText Text:=vbCrLf & vbCrLf & "Possible proper noun pairs:" _
      & vbCrLf & vbCrLf
  backHere = Selection.Start
  newList = ""
  thisMany = Len(duplicateList)
  For myCount = 1 To thisMany
    myChar = Mid(duplicateList, myCount, 1)
    If myChar = "!" Then myChar = vbCrLf
    newList = newList & myChar
  Next myCount
  Selection.InsertAfter Text:=newList
```

```
  If doCountProperNouns = True Then
    Do
      ' select word
      Selection.Words(1).Select
      thisWord = Selection
      thisMany = 0
      If Len(thisWord) > 2 Then
        Set rng = ActiveDocument.Range
        With rng.Find
          .ClearFormatting
          .MatchCase = True
          .MatchWholeWord = True
          .Text = thisWord
          .Execute
        End With
        Do While rng.Find.Found = True And rng.Start <= backHere
          thisMany = thisMany + 1
          rng.Find.Execute
          rng.Collapse wdCollapseEnd
        Loop
        num = Trim(Str(thisMany))
        Selection.Start = Selection.End
        Selection.TypeText Text:=vbTab & Trim(Str(num))
        Selection.MoveRight Unit:=wdCharacter, Count:=1
        ' Move down a line
        If Asc(Selection) = 13 Then Selection.MoveRight Unit:=wdCharacter, Count:=1
      End If
    Loop Until Asc(Selection) = 13
  End If
  thisMany = -1
End If

If makeFReditList = True Then
  allList = ""
  ' List words in endnotes
  If ActiveDocument.Endnotes.Count > 0 Then
    Set rng = ActiveDocument.StoryRanges(wdEndnotesStory)
    i = ActiveDocument.StoryRanges(wdEndnotesStory).Words.Count
    For Each wd In ActiveDocument.StoryRanges(wdEndnotesStory).Words
      rng.Start = wd.Start
      rng.End = wd.Start + 1
      If rng.HighlightColorIndex = mainColour Then
        theWord = Trim(wd)
        If Right(theWord, 1) = ChrW(8217) Then theWord = Left(theWord, Len(theWord) - 1)
        allList = allList & "m" & theWord & vbCrLf
      End If
      If rng.HighlightColorIndex = foreign1Colour Then
        theWord = Trim(wd)
        If Right(theWord, 1) = ChrW(8217) Then theWord = Left(theWord, Len(theWord) - 1)
        allList = allList & "y" & theWord & vbCrLf
      End If
      If rng.HighlightColorIndex = foreign2Colour Then
        theWord = Trim(wd)
        If Right(theWord, 1) = ChrW(8217) Then theWord = Left(theWord, Len(theWord) - 1)
        allList = allList & "z" & theWord & vbCrLf
      End If
      i = i - 1
```

```
    If i Mod 10 = 0 Then StatusBar = "Endnote word list. To go: " & Str(i)
  Next wd
End If

' List words in footnotes
If ActiveDocument.Footnotes.Count > 0 Then
  Set rng = ActiveDocument.StoryRanges(wdFootnotesStory)
  i = ActiveDocument.StoryRanges(wdFootnotesStory).Words.Count
  For Each wd In ActiveDocument.StoryRanges(wdFootnotesStory).Words
    rng.Start = wd.Start
    rng.End = wd.Start + 1
    If rng.HighlightColorIndex = mainColour Then
      theWord = Trim(wd)
      If Right(theWord, 1) = ChrW(8217) Then theWord = Left(theWord, Len(theWord) - 1)
      allList = allList & "m" & theWord & vbCrLf
    End If
    If rng.HighlightColorIndex = foreign1Colour Then
      theWord = Trim(wd)
      If Right(theWord, 1) = ChrW(8217) Then theWord = Left(theWord, Len(theWord) - 1)
      allList = allList & "y" & theWord & vbCrLf
    End If
    If rng.HighlightColorIndex = foreign2Colour Then
      theWord = Trim(wd)
      If Right(theWord, 1) = ChrW(8217) Then theWord = Left(theWord, Len(theWord) - 1)
      allList = allList & "z" & theWord & vbCrLf
    End If
    i = i - 1
    If i Mod 10 = 0 Then StatusBar = "Footnote word list. To go: " & Str(i)
  Next wd
End If

' List words in main text
i = ActiveDocument.Words.Count
Set rng = ActiveDocument.Range
For Each wd In ActiveDocument.Words
  rng.Start = wd.Start
  rng.End = wd.Start + 1
  If rng.HighlightColorIndex = mainColour Then
    theWord = Trim(wd)
    If Right(theWord, 1) = ChrW(8217) Then theWord = Left(theWord, Len(theWord) - 1)
    allList = allList & "m" & theWord & vbCrLf
  End If
  If rng.HighlightColorIndex = foreign1Colour Then
    theWord = Trim(wd)
    If Right(theWord, 1) = ChrW(8217) Then theWord = Left(theWord, Len(theWord) - 1)
    allList = allList & "y" & theWord & vbCrLf
  End If
  If rng.HighlightColorIndex = foreign2Colour Then
    theWord = Trim(wd)
    If Right(theWord, 1) = ChrW(8217) Then theWord = Left(theWord, Len(theWord) - 1)
    allList = allList & "z" & theWord & vbCrLf
  End If
  i = i - 1
  If i Mod 100 = 0 Then StatusBar = "Constructing FRedit list. To go: " & Str(i)
Next wd

Documents.Add
```

```
  Selection.TypeText Text:=allList

  Selection.WholeStory
  ' Sort the list and remove duplicates
  Selection.Sort ExcludeHeader:=False, CaseSensitive:=True, FieldNumber:="Paragraphs"
  For i = ActiveDocument.Paragraphs.Count To 2 Step -1
    Set rng1 = ActiveDocument.Paragraphs(i).Range
    Set rng2 = ActiveDocument.Paragraphs(i - 1).Range
    If rng1 = rng2 Then rng1.Delete
  Next i

  Selection.HomeKey Unit:=wdStory
  For Each myPara In ActiveDocument.Paragraphs
    If Asc(myPara) = Asc("m") Then myPara.Range.HighlightColorIndex = mainColour
    If Asc(myPara) = Asc("y") Then myPara.Range.HighlightColorIndex = foreign1Colour
    If Asc(myPara) = Asc("z") Then myPara.Range.HighlightColorIndex = foreign2Colour
  Next myPara
  Set rng = ActiveDocument.Range
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = "^p^$"
    .Replacement.Text = "^p"
    .Replacement.Highlight = False
    .Execute Replace:=wdReplaceAll
  End With
End If


' restore all apostrophe-s
Set rng = ActiveDocument.Range
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = " zczc"
  .Replacement.Text = ChrW(8217) & "s"
  .Replacement.Highlight = False
  .MatchCase = True
  .MatchWildcards = False
  .Execute Replace:=wdReplaceAll
End With

Selection.End = backHere
Selection.Start = backHere

MsgBox ((Int(10 * (Timer - timeStart) / 60) / 10) & "  minutes")
StatusBar = ""
End Sub


######################################################
Yes, Word can check for passive sentences.

See http://www.ehow.com/how_2273684_change-microsoft-word-settings-check.html


######################################################
```

To: Paul Beverley <paul@archivepub.co.uk>
Date: Fri, 25 May 2012 08:09:24 -0400
Subject: macros to change a selected European-format date into a US-format
 date IN TEXT

Hi, Paul.

Someone asked me for such a macro yesterday and it turned out to be easy. Put 'em in the book if you want to.

And yes, if the dates were in fields or something it would be different. But this author had gazillions of them in running text.

cDate turns a string into a date, and Format changes date formats. Using range.text puts it back into string/text form.

Play with the date format spec in quotes and you get other formats including dashes or slashes or a leading 0 or the day of the week.

Do you know of a faster or slicker way to do this?

- Jessica

```
Sub MakeEuropean()
  ' turn a date that is selected into a European date.  Beeps if there is an error.
  On Error GoTo oops
    Selection.Range.Text = Format(CDate(Selection.Range.Text), "d mmmm yyyy")
    End
oops:
  Beep
End Sub

Sub MakeAmerican()
 ' turn a date that is selected into a US date.  Beeps if there is an error.
 On Error GoTo oops
   Selection.Range.Text = Format(CDate(Selection.Range.Text), "mmmm d, yyyy")
End
oops:
  Beep
End Sub
```


##########
Take a look at this: http://msdn.microsoft.com/en-us/library/dd723636(v=office.12)

if you have time. There are other articles in the same section of MSDN that have interesting examples, too.

##########

```
Sub LongSentenceAndParagraphHighlighter()
' Version 19.05.12
' Highlight all sentences more than a certain length

paraMaxLength = 120
sentMaxLength = 80

For Each myPara In ActiveDocument.Paragraphs
  If myPara.Range.Words.Count > paraMaxLength Then
    myPara.Range.HighlightColorIndex = wdYellow
    myPara.Range.Select
  End If
Next

For Each mySent In ActiveDocument.Sentences
  If mySent.Words.Count > sentMaxLength Then
    mySent.HighlightColorIndex = wdRed
    mySent.Select
  End If
Next

End Sub


#########################################################
' Version 18.05.12
' Collect text of one style

thisStyle = Selection.Style
Selection.EndKey Unit:=wdStory
numParas = ActiveDocument.Paragraphs.Count
For i = 1 To numParas
  If ActiveDocument.Paragraphs(i).Style = thisStyle Then
    ActiveDocument.Paragraphs(i).Range.Copy
    Selection.Paste
  End If
  iLeft = numParas - i
  If iLeft Mod 50 = 0 Then StatusBar = "Paras left =" & Str(iLeft)
Next i
Beep

Exit Sub

thisStyle = Selection.Style
Dim myText(1000)
For Each myPara In ActiveDocument.Paragraphs
  If myPara.Range.Style = thisStyle Then
  i = i + 1
  myText(i) = myPara.Range
  End If
Next myPara
iMax = i

Selection.EndKey Unit:=wdStory

Selection.TypeText Text:=vbCrLf & vbCrLf
```

```
For i = 1 To iMax
  Selection.TypeText myText(i)
Next



##########################################################


Sub IStoIZ()
' Version 14.03.12
' Correct file to give -iz, -yz spellings

doExtraWords = False
szExceptions = "analys,reanalys,overanalys,catalys,dialys,"
szExceptions = szExceptions & "electrolys,paralys,hydrolys"

changeColour = wdGray25
nonoStyles = "DisplayQuote,ReferenceList"
' Either open the file automatically...
loadFileAutomatically = True

' from this address
myFile = "C:\Program Files\VirtualAcorn\VirtualRPC-SA" _
    & "\HardDisc4\MyFiles2\WIP\zzzTheBook\aFRedit\IS_words.doc"
' myFile = "C:\Documents and Settings\Paul\My Documents\IS_words.doc"

' or open this file first...
exceptionFile = "IS_words"

myResponse = MsgBox("IS to IZ: Edit the text?", vbQuestion + vbYesNoCancel)
If myResponse = vbCancel Then Exit Sub



Set mainDoc = ActiveDocument
If loadFileAutomatically = True Then
  Documents.Open myFile
Else
  gottadoc = False
  For Each myDoc In Documents
    If InStr(myDoc.Name, exceptionFile) > 0 Then
      myDoc.Activate
      gottadoc = True
      Exit For
    End If
  Next myDoc

  If gottadoc = False Then
    MsgBox ("Please load the IS file.")
    Exit Sub
  End If
End If


timeStart = Timer
```

```
allWords = "!"
For Each wd In ActiveDocument.Words
  thisWord = Trim(wd)
  If Asc(thisWord) > 32 Then allWords = allWords & thisWord & "!"
Next wd
allAlphas = ""
For i = 192 To 255
  If i <> 215 And i <> 247 Then allAlphas = allAlphas & ChrW(i)
Next i
For i = 65 To 90
  allAlphas = allAlphas & ChrW(i)
Next i
For i = 97 To 122
  allAlphas = allAlphas & ChrW(i)
Next i

mainDoc.Activate
myTrack = ActiveDocument.TrackRevisions
If myResponse = vbNo Then ActiveDocument.TrackRevisions = False
totChanges = 0
For hit = 1 To 3
  If hit = 1 Then
    thisMany = ActiveDocument.Endnotes.Count
    If thisMany > 0 Then
      Set rng = ActiveDocument.StoryRanges(wdEndnotesStory)
      Set rng1 = ActiveDocument.StoryRanges(wdEndnotesStory)
    End If
  End If
  If hit = 2 Then
    thisMany = ActiveDocument.Footnotes.Count
    If thisMany > 0 Then
      Set rng = ActiveDocument.StoryRanges(wdFootnotesStory)
      Set rng1 = ActiveDocument.StoryRanges(wdFootnotesStory)
    End If
  End If
  If hit = 3 Then
    Set rng = ActiveDocument.Content
    Set rng1 = ActiveDocument.Content
    thisMany = 1
  End If

  If thisMany > 0 Then
    theEnd = rng.End
    Do
      With rng.Find
        .ClearFormatting
        .Replacement.ClearFormatting
        .Text = "[iy]s[iea]"
        .Wrap = False
        .Replacement.Text = ""
        .Forward = True
        .MatchWildcards = True
        .Execute
      End With

      If rng.Find.Found = True Then
      ' Find whole word
```

```
      rng1.Start = rng.Start
      rng1.End = rng.End
      rng1.MoveEndWhile cset:=allAlphas, Count:=wdForward
      rng1.MoveStartWhile cset:=allAlphas, Count:=wdBackward
      fullWord = rng1

      changeIt = True
      ' But don't make the change if...
      thisStyle = rng.Style
      If InStr(nonoStyles, thisStyle) > 0 Then changeIt = False
      If rng.Font.StrikeThrough = True Then changeIt = False
      ' If -is- is near the beginning of the word...
      If rng.Start - rng1.Start < 4 Then
        ' look for an -is- later in the word
        rng.Start = rng1.Start + 4
        rng.End = rng.Start
        With rng.Find
          .ClearFormatting
          .Replacement.ClearFormatting
          .Text = "is[iea]"
          .Wrap = False
          .Replacement.Text = ""
          .Forward = True
          .MatchWildcards = True
          .Execute
        End With
        If rng.Find.Found = False Or rng.Start > rng1.End Then changeIt = False
      End If
    ' Check that it's not in the list of s's
      If InStr(allWords, "!" & LCase(fullWord) & "!") > 0 Then changeIt = False
      If InStr(szExceptions, Left(LCase(fullWord), 6)) > 0 And rng1.LanguageID = wdEnglishUK _
          Then changeIt = False
      If changeIt = True Then
      ' then change it to a z
        If myResponse = vbYes Then
          opposite = Replace(rng, "s", "z")
          rng.Delete
          rng.InsertAfter Text:=opposite
          If ActiveDocument.TrackRevisions = True Then
            rng1.End = rng.Start + 3
          Else
            rng1.End = rng.Start
          End If
        End If
        rng1.HighlightColorIndex = changeColour
        totChanges = totChanges + 1
      End If
      stopNow = False
    Else
      stopNow = True
    End If
    rng.Start = rng1.End
    rng.End = rng1.End
    i = theEnd - rng.End
    If (i Mod 100) = 0 And hit = 3 Then StatusBar = "To go: " & Str(i)
  Loop Until stopNow = True
End If
```

```
Next hit

StatusBar = "Finished!"

' Now see if there are any replace/mark word pairs
lastWords = Replace(allWords, "!|!", "|")
i = InStr(lastWords, "|")

' If there are some and we want to use them, do so
If doExtraWords = True And i > 0 Then
  oldColour = Options.DefaultHighlightColorIndex
  Options.DefaultHighlightColorIndex = changeColour

  oldFind = Selection.Find.Text
  oldReplace = Selection.Find.Replacement.Text

' Look back to find the beginning of the word pairs
  Do
    i = i - 1
  Loop Until Mid(lastWords, i, 1) = "!"
  lastWords = Right(lastWords, Len(lastWords) - i)
  Do
    lenWd = InStr(lastWords, "|") - 1
    If lenWd > 0 Then oldWord = Left(lastWords, lenWd)
    If lenWd > 0 And Asc(oldWord) <> Asc("#") Then
      lastWords = Right(lastWords, Len(lastWords) - lenWd - 1)
      lenWd = InStr(lastWords, "!") - 1
      newWord = Left(lastWords, lenWd)
      lastWords = Right(lastWords, Len(lastWords) - lenWd - 1)
      Set rng = ActiveDocument.Content
      With rng.Find
        .ClearFormatting
        .Replacement.ClearFormatting
        .Text = oldWord
        If myResponse = vbYes Then
          .Replacement.Text = newWord
        Else
          .Replacement.Text = "^&"
        End If
        .Replacement.Highlight = True
        .MatchWildcards = False
        .MatchCase = False
        .Execute Replace:=wdReplaceAll
      End With
    End If
  Loop Until lenWd = -1 Or Asc(oldWord) = Asc("#")

  With Selection.Find
    .Text = oldFind
    .Replacement.Text = oldReplace
    .MatchCase = True
  End With
  Options.DefaultHighlightColorIndex = oldColour
End If
ActiveDocument.TrackRevisions = myTrack
MsgBox (Timer - timeStart)
```

```
If myResponse = vbYes Then
   MsgBox ("IS words changed:  " & Str(totChanges) & "  ")
Else
   MsgBox ("IS words to be changed:  " & Str(totChanges) & "  ")
End If

End Sub
```

##

hi there,
I have loaded the spell check macro from the internet but when the document is big (say 20 pages), the Word 2003 stops responding after some pages. Sometimes, I cannot even use other Office software as it will affect the Word. Is there a limit on the resource? The macro was attached here. I want to use this macro so I can replace the mispelled word automatically and at this time, pick the top choice Word provides.

```
Sub Auto_Spell()

Dim myDoc As Document
Dim SpellSuggs As SpellingSuggestions
Set myDoc = ActiveDocument
'********* START THE SPELLING CHECK LOOP **********
Do While myDoc.SpellingErrors.Count >= 1
'Check to make sure there is at least one spelling error.
Set SpellSuggs = GetSpellingSuggestions(myDoc.SpellingErrors(1).Text)
'Get the array of spelling suggestions that Word is offering
'for the first error in the document.
If SpellSuggs.Count >= 1 Then
'If there are any suggestions, then accept the first suggestion.
myDoc.SpellingErrors(1).Text = SpellSuggs(1)
Else
'If there are NO suggestions, then IGNORE the misspelled word.
'Note: without this step, we will be stuck in an endless loop
'that will constantly attempt to get suggestions for this word.
myDoc.SpellingErrors(1).NoProofing = True
End If
Loop
'********* END THE SPELLING CHECK LOOP ********** End Sub

Sub Auto_Spell()
Dim SBar As Boolean          ' Status Bar flag
Dim TrkStatus As Boolean     ' Track Changes flag
' Store current Status Bar status, then switch on
SBar = Application.DisplayStatusBar
Application.DisplayStatusBar = True
' Store current Track Changes status, then switch off
With ActiveDocument
  TrkStatus = .TrackRevisions
  .TrackRevisions = False
End With
' Turn Off Screen Updating
Application.ScreenUpdating = False
Dim myDoc As Document, SpellSuggs As SpellingSuggestions, i As Long
Set myDoc = ActiveDocument
With myDoc
  If .SpellingErrors.Count = 0 Then
    MsgBox "No spelling errors found", vbExclamation
    Exit Sub
```

```
  End If
  'Check for spelling errors
  For i = .SpellingErrors.Count To 1 Step -1
    StatusBar = i & " spelling errors remaining"
    'Check to make sure there is at least one spelling error.
    Set SpellSuggs = GetSpellingSuggestions(myDoc.SpellingErrors(i).Text)
    'Get the array of spelling suggestions that Word offers
    'for the current error.
    If SpellSuggs.Count > 0 Then
      'If there are any suggestions, then accept the first suggestion.
      With myDoc.Content.Find
        .ClearFormatting
        .Replacement.ClearFormatting
        .Format = False
        .MatchAllWordForms = False
        .MatchWholeWord = True
        .Text = myDoc.SpellingErrors(i).Text
        .Replacement.Text = SpellSuggs(1)
        .Execute Replace:=wdReplaceAll
      End With
    Else
      'If there are NO suggestions, then IGNORE the misspelled word.
      'Note: without this step, we will be stuck in an endless loop.
      myDoc.SpellingErrors(i).NoProofing = True
    End If
    i = .SpellingErrors.Count + 1
  Next
End With
' Clear the Status Bar
Application.StatusBar = False
' Restore original Status Bar status
Application.DisplayStatusBar = SBar
' Restore original Track Changes status
ActiveDocument.TrackRevisions = TrkStatus
' Restore Screen Updating
Application.ScreenUpdating = True
MsgBox "Spellcheck Finished", vbExclamation
End Sub
```

_____

Cheers

thanks.

##

```
>     With Options
>         .ReplaceSelection = True
>         .AllowDragAndDrop = True
>         .AutoWordSelection = True
>         .INSKeyForPaste = False
>         .PasteSmartCutPaste = True
>         .AllowAccentedUppercase = False
>         .PictureEditor = "Microsoft Office Word"
>         .TabIndentKey = True
>         .Overtype = False
>         .AllowClickAndTypeMouse = True
>         .CtrlClickHyperlinkToOpen = True
```

```
>         .AutoKeyboardSwitching = False
>         .PictureWrapType = wdWrapMergeInline
>         .DisplayPasteOptions = True
>         .PromptUpdateStyle = False
>         .FormatScanning = True
>         .ShowFormatError = False
>         .SmartParaSelection = True
>         .SmartCursoring = True
>     End With
>     ActiveDocument.ClickAndTypeParagraphStyle = "Normal"
> End Sub
```

##########

I didn't think the NumPages issue was still present in 2007 / 2010. Shows you what I know.

You could try this macro unless you already have an AutoOpen macro in your template or document. It updates fields all over the doc - doc body, footers, headers, text boxes, footnote and endnote numbers and so on. Calling it AutoOpen makes it execute when the file is opened.

- Jessica

```
Sub AutoOpen
' update all fields everywhere
Dim story As Word.Range
For Each story In ActiveDocument.StoryRanges
   story.Fields.Update
Next story
End Sub
```

##########

That said, there is one resource I always recommend when strange things start happening in Word. This is Microsoft's utility for fixing issues:
http://support.microsoft.com/kb/822005

If you click the 'fix it' button, it resets registry settings. It's a bit like a 'factory reset' for MS Word. It often solves a variety of problems that can't be explained any other way.

##########
> Hi - Look here for the dialog enumerations:
http://msdn.microsoft.com/en-us/library/ff836540.aspx
>
> and in the lefthand panel click on Enumerations to see more constants and sets.
>
> This for 2010, but nearly all of it works for earlier versions.

##########

```
Sub ReplaceTwoDisPty()
ActiveDocument.TrackRevisions = False
Set rng = ActiveDocument.Content
rng.Start = Selection.Start
With rng.Find
```

```
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "DIS"
  .Wrap = wdFindContinue
  .Replacement.Text = "PTY"
  .Forward = True
  .MatchCase = True
  .MatchWildcards = False
  .Execute Replace:=wdReplaceOne
End With
rng.Start = rng.End
rng.Find.Execute Replace:=wdReplaceOne
rng.Start = rng.End
rng.Select
ActiveDocument.TrackRevisions = True
End Sub
```

###

...

Doesn't work!!

This is for Excel, but it might work.

Test if it is a Mac or a Windows Machine

Below there are two example macros that call a macro named My_Windows_Macro in Windows and a macro named My_Mac_Macro in Excel 2011 on the Mac

You can test the operating system like this with VBA code :

```
Sub WINorMAC()
'Test the OperatingSystem
  If Not Application.OperatingSystem Like "*Mac*" Then
      MsgBox "I'm a PC"
  Else
      'I am a Mac and will test if it is Excel 2011 or higher
      If Val(Application.Version) > 14 Then
        MsgBox "I'm a modern Mac"
      Else
        MsgBox "I'm an old Mac"
      End If
  End If
End Sub
```

Or use use conditional compiler constants like this with VBA code :

```
Sub WINorMAC_2()
'Test the conditional compiler constants
    #If Win32 Or Win64 Then
        'I am Windows
        Call My_Windows_Macro
```

```
   #Else
      'I am a Mac and will test if it is Excel 2011 or higher
      If Val(Application.Version) > 14 Then
         Call My_Mac_Macro
      End If
   #End If
End Sub
```

For Windows you can also test the VBA version so you know it is Excel 2010 or higher

```
   #If VBA7 Then
```

##

```
Sub InsertNamesOfFilesInAFolder()

Dim MyPath As String
Dim MyName As String

'let user select a path
With Dialogs(wdDialogCopyFile)
   If .Display() <> -1 Then Exit Sub
   MyPath = .Directory
End With

'strip quotation marks from path

If Len(MyPath) = 0 Then Exit Sub

If Asc(MyPath) = 34 Then
   MyPath = Mid$(MyPath, 2, Len(MyPath) - 2)
End If

'get files from the selected path
'and insert them into the doc
MyName = Dir$(MyPath & "*.*")
Do While MyName <> ""
   Selection.InsertAfter MyName & vbCr
   MyName = Dir
Loop

'collapse the selection
Selection.Collapse wdCollapseEnd

End Sub
```

##

Type curly quotes (before and after alpha character),
but highlighted (don't ask why!)

```
Selection.Collapse wdCollapseStart
If LCase(Selection) <> UCase(Selection) Then
  Selection.TypeText Text:=ChrW(8216)
' Or for double quotes
```

```
'  Selection.TypeText Text:=ChrW(8220)
Else
  Selection.TypeText Text:=ChrW(8217)
' Or for double quotes
'  Selection.TypeText Text:=ChrW(8221)
End If
Selection.MoveStart wdCharacter, -1
Selection.Range.HighlightColorIndex = wdBrightGreen

' Or for dark green:
' Selection.Range.HighlightColorIndex = wdGreen

Selection.Start = Selection.End
```

##

Number of current para

```
paraNum = ActiveDocument.Range(0, Selection.Paragraphs(1).Range.End).Paragraphs.Count
```

##

Information Property

Office 2003

Returns information about the specified selection or range. Read-only Variant.

expression.Information(Type)

expression Required. An expression that returns one of the objects in the Applies To list.

WdInformation

_____

WdInformation can be one of these WdInformation constants.

wdActiveEndAdjustedPageNumber Returns the number of the page that contains the active end of the specified selection or range. If you set a starting page number or make other manual adjustments, returns the adjusted page number (unlike wdActiveEndPageNumber).

wdActiveEndPageNumber Returns the number of the page that contains the active end of the specified selection or range, counting from the beginning of the document. Any manual adjustments to page numbering are disregarded (unlike wdActiveEndAdjustedPageNumber).

wdActiveEndSectionNumber Returns the number of the section that contains the active end of the specified selection or range.

wdAtEndOfRowMarker Returns True if the specified selection or range is at the end-of-row mark in a table.

wdCapsLock Returns True if Caps Lock is in effect.

wdEndOfRangeColumnNumber Returns the table column number that contains the end of the specified selection or range.

wdEndOfRangeRowNumber Returns the table row number that contains the end of the specified selection or range.

wdFirstCharacterColumnNumber Returns the character position of the first character in the specified selection or range. If the selection or range is collapsed, the character number immediately to the right of the range or selection is returned (this is the same as the character column number displayed in the status bar after "Col").

wdFirstCharacterLineNumber Returns the character position of the first character in the specified selection or range. If the selection or range is collapsed, the character number immediately to the right of the range or selection is returned (this is the same as the character line number displayed in the status bar after "Ln").

wdFrameIsSelected Returns True if the selection or range is an entire frame or text box.

wdHeaderFooterType Returns a value that indicates the type of header or footer that contains the specified selection or range, as shown in the following table.

wdHorizontalPositionRelativeToPage Returns the horizontal position of the specified selection or range; this is the distance from the left edge of the selection or range to the left edge of the page measured in points (1 point = 20 twips, 72 points = 1 inch). If the selection or range isn't within the screen area, returns - 1.

wdHorizontalPositionRelativeToTextBoundary Returns the horizontal position of the specified selection or range relative to the left edge of the nearest text boundary enclosing it, in points (1 point = 20 twips, 72 points = 1 inch). If the selection or range isn't within the screen area, returns - 1.

wdInClipboard For information about this constant, consult the language reference Help included with Microsoft Office Macintosh Edition.

wdInCommentPane Returns True if the specified selection or range is in a comment pane.

wdInEndnote Returns True if the specified selection or range is in an endnote area in print layout view or in the endnote pane in normal view.

wdInFootnote Returns True if the specified selection or range is in a footnote area in print layout view or in the footnote pane in normal view.

wdInFootnoteEndnotePane Returns True if the specified selection or range is in the footnote or endnote pane in normal view or in a footnote or endnote area in print layout view. For more nformation, see the descriptions of wdInFootnote and wdInEndnote in the preceding paragraphs.

wdInHeaderFooter Returns True if the selection or range is in the header or footer pane or in a header or footer in print layout view.

Value	Type of header or footer
- 1	None (the selection or range isn't in a header or footer)
0 (zero)	Even page header
1	Odd page header (or the only header, if there aren't odd and even headers)
2	Even page footer
3	Odd page footer (or the only footer, if there aren't odd and even footers)
4	First page header
5	First page footer

wdInMasterDocument Returns True if the selection or range is in a master document (that is, a document that contains at least one subdocument).

wdInWordMail Returns True if the selection or range is in the header or footer pane or in a header or footer in print layout view.

Value	Location
0(zero)	The selection or range isn't in an e-mail message.
1	The selection or range is in an e-mail message you are sending.
2	The selection or range is in an e-mail you are reading.

wdMaximumNumberOfColumns Returns the greatest number of table columns within any row in the selection or range.

wdMaximumNumberOfRows Returns the greatest number of table rows within the table in the specified selection or range.

wdNumberOfPagesInDocument Returns the number of pages in the document associated with the selection or range.

wdNumLock Returns True if Num Lock is in effect.

wdOverType Returns True if Overtype mode is in effect. The Overtype property can be used to change the state of the Overtype mode.

wdReferenceOfType Returns a value that indicates where the selection is in relation to a footnote, endnote, or comment reference, as shown in the following table.

Value	Description
- 1	The selection or range includes but isn't limited to a footnote, endnote, or comment reference.
0 (zero)	The selection or range isn't before a footnote, endnote, or comment reference.
1	The selection or range is before a footnote reference.
2	The selection or range is before an endnote reference.
3	The selection or range is before a comment reference.

wdRevisionMarking Returns True if change tracking is in effect.

wdSelectionMode Returns a value that indicates the current selection mode, as shown in the following table.

Value	Selection mode
0 (zero)	Normal selection

1 Extended selection ("EXT" appears on the status bar)

2 Column selection. ("COL" appears on the status bar)

wdStartOfRangeColumnNumber Returns the table column number that contains the beginning of the selection or range.

wdStartOfRangeRowNumber Returns the table row number that contains the beginning of the selection or range.

wdVerticalPositionRelativeToPage Returns the vertical position of the selection or range; this is the distance from the top edge of the selection to the top edge of the page measured in points (1 point = 20 twips, 72 points = 1 inch). If the selection isn't visible in the document window, returns - 1.

wdVerticalPositionRelativeToTextBoundary Returns the vertical position of the selection or range relative to the top edge of the nearest text boundary enclosing it, in points (1 point = 20 twips, 72 points = 1 inch). This is useful for determining the position of the insertion point within a frame or table cell. If the selection isn't visible, returns - 1.

wdWithInTable Returns True if the selection is in a table.

wdZoomPercentage Returns the current percentage of magnification as set by the Percentage property.

Example

This example displays the current page number and the total number of pages in the active document.

```
MsgBox "The selection is on page " & _
    Selection.Information(wdActiveEndPageNumber) & " of page " _
    & Selection.Information(wdNumberOfPagesInDocument)
```

If the selection is in a table, this example selects the table.

```
If Selection.Information(wdWithInTable) Then _
    Selection.Tables(1).Select
```

This example displays a message that indicates the current section number.

```
Selection.Collapse Direction:=wdCollapseStart
MsgBox "The insertion point is in section " & _
    Selection.Information(wdActiveEndSectionNumber)
```

```
#####################################################


Set rng = ActiveDocument.Range
jhkshkg = rng.Start
jhkhdf = rng.End
' rng.GoTo wdGoToLine, wdGoToAbsolute, 3
Set rng = ActiveDocument.Comments(3)
kjhkshkg = rng.Start
kjhkhdf = rng.End
cvxszcv = 0
Exit Sub
Selection.HomeKey Unit:=wdStory
Set rng = ActiveDocument.Range
' Add initials + index number to each comment
  For i = 1 To ActiveDocument.Comments.Count
    myInits = "[" & ActiveDocument.Comments(i).Initial & Trim(Str(i)) & "]"
    Selection.GoTo wdGoToComment, wdGoToAbsolute, 1
    Selection.End = Selection.End + 6
    Selection.InsertAfter myInits
  Next i
Exit Sub

  ' This works!!!
Set rng = ActiveDocument.Paragraphs(3).Range
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "e"
  .Wrap = wdFindContinue
  .Replacement.Text = ""
  .Forward = True
  .MatchWildcards = False
  .Execute
End With

rng.Select
Exit Sub

rng.Font.Italic = True
rng.SetRange Start:=0, End:=10



  ' This works!!!
Selection.SetRange Start:=0, End:=10
Exit Sub

' This doesn't work
' activedocument.Content.SetRange(0,0)
Exit Sub
```

```
 ' This works!!!
Set rng = ActiveDocument.Content
rng.End = 10
rng.Font.Bold = True
```

##

```
Give me synonyms if there are any...
Selection.Words(1).Select
Set rng = Selection.Range
Selection.Start = Selection.End
Set synInfo = rng.SynonymInfo
If synInfo.MeaningCount >= 2 Then
   synList = synInfo.SynonymList(2)
   allSyns = "["
   For i = 1 To UBound(synList)
      allSyns = allSyns & synList(i) & " "
   Next i
 rng.InsertAfter Replace(allSyns & "] ", ", ]", "]")
Else
  Beep
End If
```

##

Howard's suggestion

```
Sub DeleteAllLinksHS()
   Dim i As Integer

   With ActiveDocument
      For i = .Hyperlinks.count To 1 Step -1
         .Hyperlinks(i).Delete
      Next
   End With
End Sub
```

##
Howard's suggestion

```
Sub DeleteAllLinks2HS()
   Dim i As Integer
   Dim StoryRng As Range

   For Each StoryRng In ActiveDocument.StoryRanges
      With StoryRng
         For i = .Hyperlinks.count To 1 Step -1
            .Hyperlinks(i).Range.HighlightColorIndex = wdGray25
            .Hyperlinks(i).Delete
         Next
      End With
   Next StoryRng
End Sub
```

```
#####################################################

Sub ToggleNextCharacterCaseWithTrackingHS()
        Dim ch As Range

        Set ch = ActiveDocument.Range(Selection.Range.End, Selection.Range.End + 1)
        If UCase(ch.Text) <> LCase(ch.Text) Then ch.Text = ch.Text ' Does nothing, but is
                                                                   ' recorded as a change
        ch.Case = wdToggleCase
        .MoveRight Unit:=wdCharacter, Count:=1
End Sub


#####################################################

Create a list of macros in Normal Template.

Selection.HomeKey Unit:=wdStory
Selection.EndKey Unit:=wdStory, Extend:=wdExtend
Selection.Range.HighlightColorIndex = wdYellow

With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "Sub [a-zA-Z_]{3,}\(\)"
  .Replacement.Text = ""
  .Wrap = wdFindContinue
  .Forward = True
  .Replacement.Highlight = False
  .MatchWildcards = True
  .Execute Replace:=wdReplaceAll
End With

With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = ""
  .Highlight = True
  .Replacement.Text = "^p"
  .Forward = True
  .MatchWildcards = False
  .Execute Replace:=wdReplaceAll
End With


#####################################################
Sub StripOutMacros()
' Version 21.12.11

Set myBook = ActiveDocument
Documents.Add DocumentType:=wdNewBlankDocument
Set myMacros = ActiveDocument
myBook.Activate

Do
  With Selection.Find
```

```
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = "Sub [a-zA-Z]{3,}\(\)"
    .Wrap = False
    .Replacement.Text = ""
    .Forward = True
    .MatchWildcards = True
    .Execute
  End With
  If Selection.Find.Found = True Then
    Selection.Copy
    Selection.End = Selection.Start
    Selection.Paste
    Selection.TypeText Text:=vbCrLf
    Selection.Start = Selection.End
    With Selection.Find
      .ClearFormatting
      .Replacement.ClearFormatting
      .Text = "^pEnd S" & "ub"
      .Wrap = False
      .Forward = True
      .Replacement.Text = ""
      .MatchWildcards = False
      .Execute
    End With
    endMacro = Selection.End
    Selection.Start = Selection.End
    With Selection.Find
      .ClearFormatting
      .Replacement.ClearFormatting
      .Text = "Sub *\(\)^13"
      .Forward = False
      .Replacement.Text = ""
      .MatchWildcards = True
      .Execute
    End With
    Selection.End = Selection.Start
    Selection.End = endMacro
    Selection.Cut
    myMacros.Activate
    Selection.Paste
    Selection.Start = Selection.End
    Selection.TypeText Text:=vbCrLf & vbCrLf & vbCrLf
    myBook.Activate

  stopNow = False
  Else
    stopNow = True
  End If
  Selection.Start = Selection.End
Loop Until stopNow = True

Exit Sub



##########################################################
```

Howard Silcock's ideas

Comments on parts of Macros for Editors

I'll start with some general comments.

I notice that your style of programming involves moving the cursor around a lot. I don't know if this derives from basing your programming on recorded macros, or if you used to program with Word Basic, but I think it means that some of your macros are much more complex than they need to be. In Word Basic you couldn't do anything without moving the cursor to the place you wanted to make a change, but when they introduced VBA that all changed and you can now refer to ranges, paragraphs, and so on, without actually selecting them. That speeds things up, since moving the cursor around takes processing time. I try to avoid changing the selection or moving the cursor in a macro unless that is specifically one of the aims of the macro. My philosophy, I suppose, is to leave everything as it was except for the specific changes you want to achieve — I guess it's an extension of your practice of cleaning everything out of the Find and Replace text boxes and deselecting Match Wildcards before leaving a macro.

For example, if you stop worrying about where the cursor is, most of the work in your macro AddComment can be put into the single line

ActiveDocument.Comments.Add Range:=Selection.Range, Text:=Selection.Range.Text

(This line copies the selected text into a new comment, but doesn't add the embellishments or highlights, etc. See below for a more complete version.)

Similarly, I created my own version of your macro ScareQuoteAdd and it looked like this:

```
Sub ScareQuoteAddHS()
   With Selection.Words(1)
      .MoveEndWhile cset:=" ", Count:=wdBackward
      .InsertBefore Text:=Chr(145)
      .InsertAfter Text:=Chr(146)
   End With
End Sub
```

Notice that the selection itself isn't changed at all. The first task is to identify the word containing the selection. You can use Selection.Words(1) except that, as you know, that contains also the space or spaces after the word (I'm assuming here that the selection is just a point within the word). So you shrink that range from the end (using the range method called MoveEndWhile) until you move past the spaces. But you don't need to select the range before you do so; you just apply the change to the range you've already identified. The next two steps are to add the quote marks, but again you apply InsertBefore and InsertAfter to the now slightly shrunken range and leave the selection unchanged. When you run the macro, the selection remains exactly as it was. (Of course, sometimes it's much more difficult to pinpoint the range or whatever it is you want to change.)

The other thing that would probably make your life much easier is to • modularise' your code. Every time you find yourself carrying out the same steps, or a very similar set of steps, more than once, think whether you could separate them out into another macro or maybe a function. For example, your macros often seem to read instructions from one document and then apply them to another and it would make sense to use two separate macros, which you can then develop and test independently.

Here's an example I've been thinking about where a user-defined function could be really useful. I've been wondering how to define a function OffsetWord that would let you identify a word displaced a certain number of positions after the one containing the selection. For instance OffsetWord(Selection.Range, 1) would be the word that comes after the one containing the selection, and OffsetWord(Selection.Range, 2) would be the word after that, and so on. And hopefully you can get the word before the word containing the selection as OffsetWord(Selection.Range, —1). (And OffsetWord(Selection.Range, 0) would just be the word containing the selection— that is, it'd be the same as Selection.Range.Words(1).)

Now I'll follow with a few comments about individual bits of your document. (Note that I've skipped over your really big macros, mainly because I was too daunted to go through all the code!)

Delete All Hyperlinks

DeleteAllLinks will mess up all other fields as well as the hyperlinks — e.g. TOCs and cross-refs will be frozen. Try this instead:

```
Sub DeleteAllLinksHS()
   Dim i As Integer
```

```
   With ActiveDocument
      For i = .Hyperlinks.count To 1 Step -1
         .Hyperlinks(i).Delete
      Next
   End With
End Sub
```

(You loop through the links in reverse — from the last to the first — because each time you delete a link the numbering of the subsequent links is changed.)
That one didn‘t deal with footnotes etc. To do that, you need to loop through story ranges:

```
Sub DeleteAllLinks2HS()
   Dim i As Integer
Dim StoryRng As Range

For Each StoryRng In ActiveDocument.StoryRanges
   With StoryRng
      For i = .Hyperlinks.count To 1 Step -1
         .Hyperlinks(i).Range.HighlightColorIndex = wdGray25
         .Hyperlinks(i).Delete
      Next
   End With
Next StoryRng
End Sub
```

In that second one I also included the option (highlighted in grey) to highlight the ranges from which the links were removed.
Change Case of Next Letter
A minor quibble about this one. ToggleNextCharCase doesn‘t always preserve text formatting: if the character is originally italic or bold or red, or whatever, then its replacement only has those properties if the preceding character also had them. This is because the original character is deleted, then its replacement is typed; and a newly typed character by default just shares the formatting of the preceding one. E.g. if the text was • The white one‘ and you put the cursor before the • w‘, it would be changed to • The White one‘, unless the space was also italicised.
This one avoids that problem because nothing is typed into the document:

```
Sub ToggleNextCharCaseHS()
        With Selection
                ActiveDocument.Range(.Range.End,.Range.End + 1).Case = wdtoggleCase
                .MoveRight Unit:=wdCharacter, Count:=1
        End With
End Sub
```

If you have tracking on, that version won‘t show the change (if there is one). If you want to record a change you could force a • dummy change‘ by a trick like this:

```
Sub ToggleNextCharacterCaseWithTrackingHS()
        Dim ch As Range

        Set ch = ActiveDocument.Range(Selection.Range.End, Selection.Range.End + 1)
        If UCase(ch.Text) <> LCase(ch.Text) Then ch.Text = ch.Text ' Does nothing, but is
                                                                   ' recorded as a change
        ch.Case = wdToggleCase
        .MoveRight Unit:=wdCharacter, Count:=1
End Sub
```

Word and Phrase Frequency
There is a problem with searching for multi-word phrases. If you set MatchWholeWord = True, and try to search for a multi-word phrase, VBA simply ignores the MatchWholeWord option. If you try to do the same search manually (using the dialog box), then, once you‘ve entered a multi-word search phrase in the dialog box‘s search box, you‘ll find that the Find Whole Words Only check box is greyed out. But when you do the same search with VBA, nothing prevents you; Word goes to work and you think it‘s doing what you want, but in fact it ignores the

Find Whole Words Only command and returns strings containing parts of words as well (e.g. it‘ll find • valid entry‘ even when it appears as part of • invalid entry‘).
I think the easiest way around this may be to define a Boolean function that tests whether a range consists of whole words, then apply it to any range found in the search and ignore that range if the result is False. The following is the function I came up with.

```
Function IsWholeWordPhraseHS(Rng As Range) As Boolean
   Dim FirstWord As String, LastWord As String, RngText As String

   RngText = Trim(Rng.Text)
   FirstWord = Trim(Rng.Words(1).Text)
   LastWord = Trim(Rng.Words(Rng.Words.Count).Text)
   ' If Rng doesn‘t consist of whole words, then part of either FirstWord or
   ' Lastword must lie outside Rng

   ' Next line handles the case where the last word is followed by a single
   ' quote (either straight or curly). In this case the quote is treated as
   ' part of the word, so we need to remove it before proceeding.
   If InStr("'‘", Right(LastWord, 1)) > 0 And Len(LastWord) > 1 Then
      LastWord = Left(LastWord, Len(LastWord) - 1)
   End If

   IsWholeWordPhrase = (InStr(RngText, FirstWord) = 1 And _
            InStrRev(RngText, LastWord) = Len(RngText) - Len(LastWord) + 1)
End Function
```

As it stands, this returns True when Rng consists of a number of whole words followed (optionally) by any number of spaces, but not when there are spaces in front of the first word. That conforms, however, to Word‘s habit of including the following space or spaces when you select a word.
Using this we can define a function CountPhrOccurrencesHS that counts the number of occurrences of a phrase phr in the active document, with further parameters CaseSens and MatchWholeWords that you can set. Each time the phrase is found, it‘s tested to see whether it‘s a ”whole word phrase•; if it‘s not and you‘re doing a whole word search, then that occurrence is ignored and doesn‘t contribute to the count. This works just as well with hyphenated phrases, so you can avoid all that replacing hyphens with weird text strings. (However, the need to include the test for each string found does slow the macro down.)

```
Function CountPhrOccurrencesHS(phr As String, MatchWholeWords As Boolean, _
        Optional CaseSens As Boolean = False) As Integer
' Finds and counts occurrences of the phrase phr in a document or in the selected range.
' Set MatchWholeWords = True to ignore occurrences where one or more words in phr occur
' only as part of a word in the document or range - e.g. where 'valid response' is found
' inside 'invalid response'.
' If a range is selected, the count is confined to that range. Otherwise it applies to
' the entire document. Set CaseSens = True for a case sensitive search; omit or set to
' False for a case insensitive search.
   Dim Rng As Range
   Dim Count As Integer, RngEndPt As Long

   If phr = "" Then    ' Put this check in to avoid possible unintended searches
      CountPhrOccurrencesHS = 0
      Exit Function
   End If

   Count = 0

   If Selection.Type = wdSelectionNormal Then  ' Selection is a block of text
      Set Rng = Selection.Range
   Else    ' Selection is just a point (or maybe a graphic or whatever)
```

```
    Set Rng = ActiveDocument.Range
  End If

  RngEndPt = Rng.End
  With Rng.Find
    .ClearFormatting
    .Text = phr
    .MatchCase = CaseSens
    .Format = False
    .MatchWholeWord = False 'Turn off built-in whole-word matching to avoid problems
    .Execute

    While .Found And Len(Rng.Text) > 0
      If IsWholeWordPhrase(Rng) Or Not MatchWholeWords Then
        Count = Count + 1
      End If
      Set Rng = ActiveDocument.Range(Rng.End, RngEndPt)
      .Execute
    Wend
  End With
  CountPhrOccurrencesHS = Count
End Function
```

Add Comment

This is a more complete version of the one-liner mentioned in the introduction.

```
Sub CommentAddHS(attrib As String,
          Optional textHighlightColour As WdColorIndex = wdNoHighlight, _
          Optional textColour As WdColorIndex = wdAuto)
  Dim CommentText As String

  With Selection
    CommentText = attrib & "p." & .Range.Information(wdActiveEndPageNumber) & _
      " " & Chr(150) & " " _
      & Chr(145) & Trim(.Range.Text) & Chr(146)
    ActiveDocument.Comments.Add Range:=.Range, Text:=CommentText
    .Range.HighlightColorIndex = textHighlightColour
    .Range.Font.Color = textColour
  End With
End Sub
```

You‘ll note that I‘ve put the user-specified values attrib, textHighlightColour and textColour as parameters of the macro and set default values. As I notice that you don‘t use this approach, I wonder if you are familiar with this use of parameters. When a macro has parameters, you can‘t launch it with a keystroke, but you define another macro that sets the values for these (maybe you could prompt the user to supply them) and then call the macro with parameters from within that macro — which you can launch by a keystroke. The • optional‘ parameters can be omitted when you call the macro and are then set to the specified values (in this case wdNoHighlight and wdAuto, which effectively amount to • doing nothing‘ to the highlighting and font attributes). One more thing: I guess you may not be aware that you can get the page number of a range using the built-in Information(wdActiveEndPageNumber). One of those things that are hard to find out about except by chance or from someone else‘s code. Now you know where to look, you can find more about it from the online help.
I didn‘t cover the issue of change-tracking, but you could add another parameter to tell the macro if you want it to turn tracking off before and then back on again after.

Well, I guess that should be enough to be going on withŒ
Howard

##
> I've not discovered the Immediate window - that's new to me.

I think it was a while before I started using it, It's a great way to try lines of code out. Just press Ctrl-G to open it. Then you can type a line of code, hit return and it will carry it out on the active document. Or type ? then some expression (as in the example I suggested) and it will evaluate it and type out the result. If a macro aborts during execution, you can use this to evaluate different things to work out what's going on.

##

> The one place where you specifically ask for input is on the topic of
> fields - particularly your FieldsULink macro. For the example task you
> mention, it seems likely that for automatic figure numbers and equation
> numbers the fields used would most likely be SEQUENCE fields and it would
> be much safer, if that's the case, to unlink only that type of field. You
> can test this hypothesis by selecting one of the fields and then typing in
> the Immediate window of the VBA editor

I've not discovered the Immediate window - that's new to me.

> ?selection.Range.Fields(1).Type
>
> If I'm right, this would return a value of 12 (which is equal to the VBA
> constant wdFieldSequence). In that case, you could do what you want by
> looping through the fields and testing whether fld.type = 12 instead of
> testing for fld.type <> 58. That would mean you'd avoid unlocking a whole
> lot of other fields that could potentially be in the document, including
> things likes filenames, page numbers and TOCs. In fact you may be able to
> narrow down the fields to be unlocked even further because each SEQUENCE
> field has a particular sequence name that you could test for.

##

```
Selection.MoveRight Unit:=wdCharacter, Count:=1, Extend:=wdExtend
NormalTemplate.AutoTextEntries.AppendToSpike Range:=Selection.Range
Selection.Delete
With NormalTemplate.AutoTextEntries("Spike")
  .Insert Where:=Selection.Range, RichText:=True
  .Delete
End With
```

##

```
Sub TrackChangeExamine()
' Version 02.03.11
' Examine the track changes
maxRev = ActiveDocument.Revisions.Count
WordBasic.NextChangeOrComment
nowRev = ActiveDocument.Range(0, Selection.Range.Revisions(1).Range.End).Revisions.Count
Selection.Start = Selection.End
```

```
For i = nowRev To maxRev
  Set myRev = ActiveDocument.Revisions(i)
  If ActiveDocument.Revisions(i).Type > 2 Then
    myRev.Range.Select
    MsgBox ("Type: " & myRev.Type & "  - " & myRev.FormatDescription)
  End If
  StatusBar = "Revision number: " & Str(maxRev - i)
Next i
Beep
End Sub


######################################################
Sub FindAndReplaceFirstStoryOfEachType()
  Dim rngStory As Range
  For Each rngStory In ActiveDocument.StoryRanges
    With rngStory.Find
      .Text = "find text"
      .Replacement.Text = "I'm found"
      .Wrap = wdFindContinue
      .Execute Replace:=wdReplaceAll
    End With
  Next rngStory
End Sub


######################################################
```
(From GoldMine)

Manipulating the clipboard using VBA

Article contributed by Jonathan West

Although VB6 has a Clipboard object which you can manipulate, Word VBA doesn't. This is how to clear the clipboard in VBA:

```
Dim myData As DataObject

Set MyData = New DataObject
MyData.SetText ""
MyData.PutInClipboard
```

This is how to get the text on the clipboard into a string variable:

```
Dim MyData As DataObject
Dim strClip As String

Set MyData = New DataObject
MyData.GetFromClipboard
strClip = MyData.GetText
Set MyData = New DataObject
MyData.GetFromClipboard
strClip = MyData.GetText
```

This is how to get the text from a string variable into the clipboard:

```
Dim MyData As DataObject
```

```
Dim strClip As String
strClip = "Hi there"

Set MyData = New DataObject

MyData.SetText strClip
MyData.PutInClipboard
```

The DataObject object is a part of the Forms library in VBA. In order to make this code work, you must do one of two things.

Have at least one UserForm in your project, or
In the VBA editor, go to Tools, References, and set a reference to the "Microsoft Forms 2.0 Object Library"

##########

(From GoldMine)
SortArray

This is perhaps the most useful of the commands left behind. It allows you to sort the elements of an array using a single line of code. At its simplest, you can use it on a one-dimensional array as follows.

```
Sub SortTest()
   Dim ss(2) As String
   Dim i As Long

   ss(0) = "orange"
   ss(1) = "apple"
   ss(2) = "banana"
   WordBasic.SortArray ss()

   For i = 0 To 2
      Debug.Print ss(i)
   Next i

End Sub
```

This sorts the array in ascending alphabetical order

However, you can also sort in descending order, and sort either dimension of a two-dimension array. The full list of the SortArray arguments is as follows

SortArray ArrayName[$]() [, Order] [, From] [, To] [, SortType] [, SortKey]

ArrayName is the name of the array

Order is 0 for ascending (by default), 1 for descending

From is the first element to sort (0 by default)

To is the last element to sort (by default the last element of the array)

SortType determines whether you are sorting rows or columns. 0 (default) for rows, 1 for columns

SortKey is applicable only to two-dimensional arrays, and indicates the row or column used as the sort key. It is 0 by default

Note that, unlike most VBA methods, you don't use named arguments with this command; thus you can have

```
WordBasic.SortArray MailingList$(), 1, 1, 20, 0, 1
```

but not

```
WordBasic.SortArray ArrayName:=MailingList$(), Order:=1, From:=1, To:=20, _
      SortType:=0, SortKey:=1
```

Also, you cannot miss out arguments if you want to use later ones, thus you can have

```
WordBasic.SortArray Test(), 0, 0, 2, 0, 1
```

but not

```
WordBasic.SortArray Test(), 0, , , , 1
```

There is one other limitation of the SortArray command. It will sort an array declared as such, but it will not sort an array that is contained in a Variant. If you create an array like this:

```
Dim vArray as Variant
vArray = Array("orange", "apple", "banana")
```

SortArray will not sort it.

(Also if you do not declare your array at all, it will be treated as a variant and will not be sorted).

FileNameInfo$()

This is another very useful function for which there is no direct VBA equivalent. FileNameInfo allows you to get just the filename or a fully qualified pathname from a filename given to it. The nearest equivalent in VBA are the Name, FullName and Path properties of the Document object.

FileNameInfo is different in that you don't need to have the document open.

The syntax is

x = WordBasic.FilenameInfo$(Filename$, FileType)

where Filename is the name of the file, and FileType is a number
which defines the part of the filename you want to return:

1 - the full pathname, e.g. C:\My Documents\My File.doc"
2 - the filename only, if the file is in the current folder, otherwise the full pathname
3 - the filename only
4 - the filename without the extension
5 - the path without the filename
6 - the UNC pathname

One case where FileNameInfo$ is very useful is to get the pathname of
a file which has just been selected by the user in the FileOpen
dialog. The following code returns the full pathname of a file
selected by the user.

```
With Dialogs(wdDialogFileOpen)
   If .Display Then
      MsgBox WordBasic.FilenameInfo$(.Name, 1)
   Else
      MsgBox "No file selected"
   End If
End With
```

ToolsBulletsNumbers

WordBasic allows you to remove all manually typed numbering from a
selection using the old Word 2 command:

```
WordBasic.ToolsBulletsNumbers Replace:=0, Type:=1, Remove:=1
```

This is particularly useful for removing manually typed numbering
from Headings in a document you have been emailed, prior to applying
List Numbering. If you go into Outline View, set the Heading Level to
the number of levels you need to remove the typed numbering from, and
run the above line, it will just remove numbering from those Headings
and will leave the body text alone.

##
Displaying errors?

```
Err.Raise Err.Number, Err.Source, Err.Description
```

##
Problems with CiteCheck not recognising filenames:

I figured out the problem. It was the stupid compatibility mode all along.
You had taken care of this problem in your code except that I think Word
2010 has a slightly different naming convention for the actual window name
where the window name is now "FileName.docx [compatibility mode] " instead
of "FileName [compatibility mode] .docx"

So, the code that works for me looks like this (change highlighted in
yellow):

```
' Go back to text and grey all identical citations

lookingFor = textDoc

      Windows(textDoc).Activate

      If Err.Number = 5941 Then

        Err.Clear

        textDoc = Replace(textDoc, ".docx", ".docx [Compatibility Mode]")

        'textDoc = Replace(textDoc, ".", " [Compatibility Mode].")

        Windows(textDoc).Activate

        If Err.Number = 5941 Then GoTo ReportIt
```

I tried adding a few lines so that it would work for both naming conventions, but I kept getting an error, so maybe you can figure out how to do that?

Cheers,
Clark

##
The following sample Visual Basic for Applications macros demonstrate how to change the value of the *Title* field in the *Properties* dialog box. The following sample also includes code to trap the error, in case there are no documents open, and to display a message:

```
Sub ChangeDocProperties()

  On Error GoTo ErrHandler
    ActiveDocument.BuiltInDocumentProperties("Title") = "My Title"
  Exit Sub

ErrHandler:
  If Err <> 0 Then

    ' Display an error message.
    MsgBox Err.Description

    'Clear the error.
    Err.Clear
    Resume Next

  End If

End Sub
```

##
Word 2004 doesn't have the Replace() function!

If you find the rogue line:

```
  Selection.InsertAfter Text:=Replace(duplicateList, "!", vbCrLf)
```

delete that single line and then replace it with:

```
  newList = ""
  For i = 1 To Len(duplicateList)
    myChar = Mid(duplicateList, i, 1)
    If myChar = "!" Then myChar = vbCrLf
    newList = newList & myChar
  Next i
  Selection.InsertAfter Text:=newList
```

It takes those first six lines to do exactly the same thing as the
single command Replace(duplicateList, "!", vbCrLf)!

##

> I have a little time on my hands today. Here is a pair of macros
that substitute for Word's command to toggle change tracking. Tested
in 2003, which is what I have available at the moment.
>
> If change tracking is on, it's turned off with a reminder appearing
one minute later. You can change that time interval.
>
> If change tracking is off, it gives you the choice of leaving it
off or turning it back on. No reminder message in either case.
>
> Limitations: you have to respond to the initial message box. And
there is only one reminder message. It doesn't continue to nag you.
>
> Potential danger: you're replacing one of Word's commands. The
TRK toggle and the toggle in the menus and toolbars (and ribbon, I
assume) run the macro, not Word's internal code. I've done some
testing, but not enough to ensure that there are no hidden dangers.
You can change the macro name to something else, but then you'll have
to use some other method of invoking it other than the Word UI.
>
> All that said, here are the macros. Use at your own risk. Let me know if they don't work for you.
>

```
> Sub ToolsRevisionMarksToggle()
> Dim Msg, Style, Title, Response
> Style = vbYesNo + vbCritical + vbDefaultButton2
> Msg = "Track Changes is already off.  Turn it back on?"
> Title = "Track Changes with Reminder"
> If ActiveDocument.TrackRevisions = False Then
>     Response = MsgBox(Msg, Style, Title)
>     If Response = vbYes Then    ' User chose Yes.
>         ActiveDocument.TrackRevisions = True
>     Else    ' User chose No.
>         MsgBox "ok, it's still off."
>     End If
> Else
>    MsgBox "Track Changes going off.  Reminder in 1 minute"  'change text of message if you change the
reminder time
>     ActiveDocument.TrackRevisions = False
```

```
>     alertTime = Now + TimeValue("00:01:00")
>       ' change the value in parentheses to whatever you want. Format is hours:minutes:seconds.
>     Application.OnTime alertTime, "TCReminderMacro"
> End If
> End Sub
>
> Sub TCReminderMacro()
> If ActiveDocument.TrackRevisions = False Then
>   MsgBox "Don't forget to turn Track Changes back on"
> End If
> End Sub
> * Posting rules & information: http://www.copyediting-l.info
> * Job opportunity messages must be tagged "JOB-OP:"
>
```

##

Background colours:

Selection.Shading.BackgroundPatternColorIndex = wdTurquoise

##

```
Sub Test()
' Version 15.09.11
' Display a highlighted (or selected) quote


Dim cmnt As Word.Comment
totCmnts = ActiveDocument.Comments.Count
ReDim Auth(totCmnts) As String
ReDim cmText(totCmnts) As String
CR2 = vbCrLf & vbCrLf
Authlist = ""
numAuths = 0
For i = 1 To totCmnts
  Set cmnt = ActiveDocument.Comments(i)

'  sdfsd = cmnt.Creator
'  sdsd = cmnt.Initial
  Auth(i) = cmnt.Initial
  cmText(i) = cmnt.Range
  If Left(cmText(i), Len(Auth(i))) <> Auth(i) Then
    cmText(i) = Auth(i) & ": " & cmText(i)
  End If
'  If InStr(Authlist, Auth(i)) = 0 Then
'    numAuths = numAuths + 1
'    Authlist = Authlist & Auth(i) & "|"
'  End If
Next i

Documents.Add
Selection.TypeParagraph
For i = 1 To totCmnts
  startHere = Selection.Start
  Selection.TypeText Text:="[" & cmText(i) & CR2
  endHere = Selection.Start
  nowLen = Selection.Start - startHere
```

```
  Selection.End = startHere + InStr(cmText(i), ":")
  extraBit = Trim(Str(i)) & "]"
  Selection.TypeText Text:=extraBit
  Selection.Start = endHere + Len(extraBit)
  Selection.End = Selection.Start
Next i

Exit Sub
For au = 1 To numAuths
  nameEnd = InStr(Authlist, "|")
  thisName = Left(Authlist, nameEnd - 1)
  Authlist = Mid(Authlist, nameEnd + 1)
  Set rng = ActiveDocument.Content
  i = 1
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = "^p" & thisName & ":"
    .Wrap = wdFindContinue
    .Replacement.Text = ""
    .Forward = True
    .MatchWildcards = False
    .Execute
  End With

  While rng.Find.Found
    theEnd = rng.End
    rng.Start = rng.Start + 1
    rng.End = rng.Start
    rng.InsertAfter "["
' rng.Select
    rng.Start = theEnd
    rng.InsertAfter Str(i) & "]"
    rng.Start = rng.End
    i = i + 1
    rng.Find.Execute
rng.Select
  Wend
Next au
Exit Sub


achar = "a"

MsgBox (Val(achar))
Exit Sub

##########################################################

Bridge characters: clubs, diamonds, hearts, spades

Sub SuitToText()
For Each fld In ActiveDocument.Fields
  myText = fld.Code.Text
  codePos = InStr(myText, "SYMBOL ") + 7
  myCode = Mid(myText, codePos, 3)
  Select Case Val(myCode)
```

```
    Case 167: mySuit = "cx"
    Case 168: mySuit = "dx"
    Case 169: mySuit = "hx"
    Case 170: mySuit = "sx"
    Case Else: mySuit = "??????"
  End Select
  fld.Select
  Selection.TypeText Text:=mySuit
Next
End Sub

#########################################################
Sub NewMembers()
' Version 23.07.11
' Create alpha list in a paragraph
Selection.WholeStory
Selection.Fields.Unlink
Set rng = ActiveDocument.Content
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "[0-9][!^13]@ ^= "
  .Wrap = wdFindContinue
  .Replacement.Text = ""
  .MatchWildcards = True
  .Execute Replace:=wdReplaceAll
End With

With rng.Find
  .Text = " ^= *^13"
  .Replacement.Text = "^p"
  .MatchWildcards = True
  .Execute Replace:=wdReplaceAll
End With

With rng.Find
  .Text = "^p^p"
  .Replacement.Text = "^p"
  .MatchWildcards = False
  .Execute Replace:=wdReplaceAll
End With

With rng.Find
  .Text = " "
  .Replacement.Text = "^t"
  .MatchWildcards = False
  .Execute Replace:=wdReplaceAll
End With

Selection.WholeStory
Selection.ConvertToTable Separator:=wdSeparateByTabs, NumColumns:=2, AutoFitBehavior:=wdAutoFitFixed
With Selection.Tables(1)
  .Style = "Table Grid"
  .ApplyStyleHeadingRows = True
  .ApplyStyleLastRow = False
  .ApplyStyleFirstColumn = True
  .ApplyStyleLastColumn = False
```

```
End With
Selection.Sort ExcludeHeader:=False, FieldNumber:="Column 2", _
    SortFieldType:=wdSortFieldAlphanumeric, SortOrder:=wdSortOrderAscending, _
    FieldNumber2:="", SortFieldType2:=wdSortFieldAlphanumeric, SortOrder2:= _
    wdSortOrderAscending, FieldNumber3:="", SortFieldType3:= _
    wdSortFieldAlphanumeric, SortOrder3:=wdSortOrderAscending, Separator:= _
    wdSortSeparateByCommas, SortColumn:=False, CaseSensitive:=False, _
    LanguageID:=wdEnglishUK, SubFieldNumber:="Paragraphs", SubFieldNumber2:= _
    "Paragraphs", SubFieldNumber3:="Paragraphs"
Selection.WholeStory
Selection.Cut
Selection.PasteAndFormat (wdFormatPlainText)
Set rng = ActiveDocument.Content
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "^t"
  .Wrap = wdFindContinue
  .Replacement.Text = " "
  .MatchWildcards = True
  .Execute Replace:=wdReplaceAll
End With
With rng.Find
  .Text = "^p"
  .Replacement.Text = ", "
  .MatchWildcards = False
  .Execute Replace:=wdReplaceAll
End With

End Sub

########################################################
Sub RefGrabber()
' Version 19.10.10
' Select the current reference, then say "Not in list"
Selection.End = Selection.Start
oldFind = Selection.Find.Text
oldReplace = Selection.Find.Replacement.Text

' Find a number
startHere = Selection.Start
With Selection.Find
  .ClearFormatting
  .Text = ")"
  .Replacement.Text = ""
  .MatchWildcards = False
  .Execute
End With

Selection.Start = startHere

Selection.End = Selection.End - 1
' Restore Find to original
With Selection.Find
  .Text = oldFind
  .Replacement.Text = oldReplace
End With
```

```
  Application.Run MacroName:="Normal.NewMacros.CommentAdd2"

End Sub


#########################################################
Sub PDFpagerSimple()
' Version 11.05.11
' Highlight all the page numbers (left aligned)

numDashes = 20
FontSize = 24

' Find the first number
Set rng = ActiveDocument.Range
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "\>\>[0-9]@\<\<"
  .MatchWildcards = True
  .Replacement.Text = ""
  .Execute
End With

If rng.Find.Found = False Then
  MsgBox ("Mark first and last page numbers, e.g. >>1<<")
  Exit Sub
End If

startHere = rng.Start
rng.Start = rng.Start + 2
firstNum = Val(rng)
rng.Start = rng.End + 2

' Find the final number
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "\>\>[0-9]@\<\<"
  .MatchWildcards = True
  .Replacement.Text = ""
  .Execute
End With
If rng.Find.Found = False Then
  MsgBox ("Mark first and last page numbers like this: >>123<<")
  Exit Sub
End If

rng.Start = rng.Start + 2
lastNum = Val(rng)

Set rng = ActiveDocument.Range
rng.Start = startHere
For i = lastNum - 1 To firstNum + 1 Step -1
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
```

```
    .Text = "^p" & Trim(Str(i))
    .MatchWildcards = False
    .Forward = False
    .Replacement.Text = ""
    .Execute
  End With
  rng.Select
  If rng.Find.Found = True Then
    rng.MoveStart wdCharacter, 1
    rng.InsertBefore ">>"
    rng.InsertAfter "<<"
  Else
    rng.InsertBefore vbCrLf & ">>" & Trim(Str(i)) & "<<" & vbCrLf
  End If
  rng.End = rng.Start
  StatusBar = "Page: " & Str(i)
Next i

dottedLine = ""
For i = 1 To numDashes
  dottedLine = dottedLine & Chr(150) & " "
Next i

Set rng = ActiveDocument.Range
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "^13([ixv]@)^13"
  .Replacement.Text = "^p>>\1<<^p"
  .Replacement.Font.Size = FontSize
  .Forward = True
  .Replacement.Font.Bold = True
  .MatchWildcards = True
  .Execute Replace:=wdReplaceAll
End With

Set rng = ActiveDocument.Range
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "\>\>[0-9ixv]@\<\<"
  .Replacement.Text = dottedLine & "^p^&"
  .Replacement.Font.Size = FontSize
  .Replacement.Font.Bold = True
  .MatchWildcards = True
  .Execute Replace:=wdReplaceAll
End With

Selection.HomeKey Unit:=wdStory
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = ">>" & Trim(Str(firstNum)) & "<<"
  .MatchWildcards = False
  .Replacement.Text = ""
  .Execute
End With
```

```
Selection.End = Selection.Start
End Sub
Sub PDFpagerOddEven()
' Version 11.05.11
' Highlight all the page numbers alternately left & right

numDashes = 20
FontSize = 24

' Find the first number
Set rng = ActiveDocument.Range
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "\>\>[0-9]@\<\<"
  .MatchWildcards = True
  .Replacement.Text = ""
  .Execute
End With

If rng.Find.Found = False Then
  MsgBox ("Mark first and last page numbers, e.g. >>1<<")
  Exit Sub
End If

startHere = rng.Start
rng.Start = rng.Start + 2
firstNum = Val(rng)
rng.Start = rng.End + 2

' Find the final number
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "\>\>[0-9]@\<\<"
  .MatchWildcards = True
  .Replacement.Text = ""
  .Execute
End With
endHere = rng.Start
If rng.Find.Found = False Then
  MsgBox ("Mark first and last page numbers like this: >>123<<")
  Exit Sub
End If

rng.Start = rng.Start + 2
lastNum = Val(rng)

Set rng = ActiveDocument.Range
'  rng.Start = endHere
For i = lastNum - 1 To firstNum + 1 Step -1
  If i Mod 2 = 0 Then
    findText = "^p" & Trim(Str(i))
  Else
    findText = Trim(Str(i)) & "^p"
  End If
  With rng.Find
```

```
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = findText
    .Forward = False
    .MatchWildcards = False
    .Replacement.Text = ""
    .Execute
  End With
  If rng.Find.Found = True Then
    If i Mod 2 = 0 Then
      rng.MoveStart wdCharacter, 1
    Else
      rng.MoveEnd wdCharacter, -1
    End If
    rng.InsertBefore ">>"
    rng.InsertAfter "<<"
  Else
    rng.InsertBefore vbCrLf & ">>" & Trim(Str(i)) & "<<" & vbCrLf
  End If
  rng.End = rng.Start
  StatusBar = "Page: " & Str(i)
Next i

dottedLine = ""
For i = 1 To numDashes
  dottedLine = dottedLine & Chr(150) & " "
Next i

Set rng = ActiveDocument.Range
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "^13([ixv]@)^13"
  .Replacement.Text = "^p>>\1<<^p"
  .Replacement.Font.Size = FontSize
  .Replacement.Font.Bold = True
  .Forward = True
  .MatchWildcards = True
  .Execute Replace:=wdReplaceAll
End With

Set rng = ActiveDocument.Range
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "\>\>[0-9ixv]@\<\<"
  .Replacement.Text = "^p" & dottedLine & "^p^&"
  .Replacement.Font.Size = FontSize
  .Replacement.Font.Bold = True
  .MatchWildcards = True
  .Execute Replace:=wdReplaceAll
End With

Set rng = ActiveDocument.Range
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
```

```
  .Text = "^p^p"
  .Replacement.Text = "^p"
  .MatchWildcards = False
  .Execute Replace:=wdReplaceAll
End With

Selection.HomeKey Unit:=wdStory
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = ">>" & Trim(Str(firstNum)) & "<<"
  .MatchWildcards = False
  .Replacement.Text = ""
  .Execute
End With
Selection.End = Selection.Start

End Sub

Sub PDFpagerRightOnly()
' Version 11.05.11
' Highlight all the page numbers (right aligned)

numDashes = 20
FontSize = 24

' Find the first number
Set rng = ActiveDocument.Range
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "\>\>[0-9]@\<\<"
  .MatchWildcards = True
  .Replacement.Text = ""
  .Execute
End With

If rng.Find.Found = False Then
  MsgBox ("Mark first and last page numbers, e.g. >>1<<")
  Exit Sub
End If

startHere = rng.Start
rng.Start = rng.Start + 2
firstNum = Val(rng)
rng.Start = rng.End + 2

' Find the final number
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "\>\>[0-9]@\<\<"
  .MatchWildcards = True
  .Replacement.Text = ""
  .Execute
End With
If rng.Find.Found = False Then
```

```
  MsgBox ("Mark first and last page numbers like this: >>123<<")
  Exit Sub
End If

rng.Start = rng.Start + 2
lastNum = Val(rng)

Set rng = ActiveDocument.Range
rng.Start = startHere
For i = lastNum - 1 To firstNum + 1 Step -1
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting

' I've added a letter or a space before the number

    .Text = "[a-zA-Z ]" & Trim(Str(i)) & "^13"
    .MatchWildcards = True
    .Forward = False
    .Replacement.Text = ""
    .Execute
  End With
  If rng.Find.Found = True Then

' so I have to move the start of the find forward one char

    rng.MoveStart wdCharacter, 1
    rng.MoveEnd wdCharacter, -1
    rng.InsertBefore ">>"
    rng.InsertAfter "<<"
  Else
    rng.InsertBefore vbCrLf & ">>" & Trim(Str(i)) & "<<" & vbCrLf
  End If
  rng.End = rng.Start
  StatusBar = "Page: " & Str(i)
Next i

dottedLine = ""
For i = 1 To numDashes
  dottedLine = dottedLine & Chr(150) & " "
Next i

Set rng = ActiveDocument.Range
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "\>\>[0-9ixv]@\<\<"
  .Replacement.Text = dottedLine & "^p^&"
  .Replacement.Font.Size = FontSize
  .Replacement.Font.Bold = True
  .MatchWildcards = True
  .Execute Replace:=wdReplaceAll
End With

Selection.HomeKey Unit:=wdStory
With Selection.Find
  .ClearFormatting
```

```
  .Replacement.ClearFormatting
  .Text = ">>" & Trim(Str(firstNum)) & "<<"
  .MatchWildcards = False
  .Replacement.Text = ""
  .Execute
End With
Selection.End = Selection.Start

End Sub

Sub PDFPagerManual()

numberAtBottomPage = True
numDashes = 20
FontSize = 24

dottedLine = ""
For i = 1 To numDashes
  dottedLine = dottedLine & Chr(150) & " "
Next i

If Asc(Selection) = Asc("<") Then Selection.MoveEnd wdCharacter, -1
Selection.Words(1).Select
startHere = Selection.Start
wasNum = Val(Selection)
nextNum = wasNum + 1
Selection.Start = Selection.End
If Asc(Selection) <> Asc("<") Then
  Selection.Start = startHere
  If numberAtBottomPage = False Then Selection.InsertBefore vbCrLf & dottedLine
  Selection.InsertBefore ">>"
  Selection.InsertAfter "<<" & vbCrLf
  If numberAtBottomPage = True Then Selection.InsertAfter dottedLine & vbCrLf
  Selection.Font.Size = FontSize
  Selection.Font.Bold = True
  Selection.Start = Selection.End
End If
gotOne = False
aveLength = 0
Do
  wasHere = Selection.Start
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = Trim(Str(nextNum))
    .MatchWildcards = False
    .Forward = True
    .Replacement.Text = ""
    .Execute
  End With
  If Selection.Find.Found = True Then
    myResponse = MsgBox("OK?", vbQuestion + vbYesNoCancel)
    hereNow = Selection.Start
    If aveLength = 0 Then
      aveLength = hereNow - wasHere
    Else
      aveLength = (hereNow - startHere) / (nextNum - wasNum)
```

```
    End If
    If (hereNow - wasHere) > 2 * aveLength Then
      Beep
        Beep
      myTime = Timer
      Do
      Loop Until Timer > myTime + 0.1
    End If
    If myResponse = vbCancel Then Selection.Find.Text = "<<": Exit Sub
    If myResponse = vbYes Then
      justHere = Selection.Start
      If numberAtBottomPage = False Then Selection.InsertBefore vbCrLf & dottedLine
      Selection.InsertBefore ">>"
      Selection.InsertAfter "<<" & vbCrLf
      If numberAtBottomPage = True Then Selection.InsertAfter dottedLine & vbCrLf
      Selection.Start = justHere
      Selection.Font.Size = FontSize
      Selection.Font.Bold = True
      Selection.Start = Selection.End
      gotOne = True
      nextNum = nextNum + 1
    End If
  Else
    Selection.Paragraphs(1).Range.Select
    Selection.Start = Selection.End
    justHere = Selection.Start
    If numberAtBottomPage = False Then Selection.InsertBefore vbCrLf & dottedLine
    Selection.InsertBefore ">>" & Trim(Str(nextNum)) & "<<" & vbCrLf
    Selection.InsertAfter "<<" & vbCrLf
    If numberAtBottomPage = True Then Selection.InsertAfter dottedLine & vbCrLf
    Selection.Start = justHere
    Selection.Font.Size = FontSize
    Selection.Font.Bold = True
    gotOne = False
  End If
Loop Until gotOne = False
Selection.Find.Text = "<<"
End Sub
```

##

Unfinished pair of selection extenders

```
Sub FindExtend()
startWas = Selection.Start
endWas = Selection.End
nowText = Selection.Find.Text
myText = InputBox("Find?", "Extend finder", nowText)

Selection.Start = Selection.End
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  If Len(myText) = 0 Then .Text = myText
  .Forward = True
  .MatchCase = False
  .Wrap = False
```

```
  .MatchWildcards = False
  .Execute
End With
Selection.End = rng.End

End Sub

Sub FindExtendUp()
startWas = Selection.Start
endWas = Selection.End
nowText = Selection.Find.Text
myText = InputBox("Find?", "Extend finder", nowText)

Selection.End = Selection.Start
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  If Len(myText) > 0 Then .Text = myText
  .Forward = False
  .MatchCase = False
  .Wrap = False
  .MatchWildcards = False
  .Execute
End With
Selection.End = endWas
End Sub
```

##########

Date: Wed, 13 Apr 2011 12:21:43 -0400
Subject: Re: [CE-L] TOOLS: search for tables?
To: COPYEDITING-L@LISTSERV.INDIANA.EDU

For tables, try this macro. It goes through the active doc, stops at each table and asks you if you want to convert to text, then does it (with tabs as separators) if you say yes. If you select Cancel, it stops the search. Otherwise it goes to the next table and asks.

To separate by commas, change "wdSeparateByTabs" to " wdSeparateByCommas ". To separate by paragraphs, change it to "wdSeparateByParagraphs".

- Jessica

```
Sub convertToText()
'
'
' find tables and convert to text

' Now let them choose for each one

Dim mytable As Table
Dim msgText As String
Dim msgStyle As String
Dim msgTitle As String
Dim Response

msgStyle = vbYesNoCancel + vbQuestion
msgText = "Do you want to convert this table to text?"
msgTitle = "Converting to text"
```

```
For Each mytable In ActiveDocument.Tables
   mytable.Select
   Response = MsgBox(msgTitle, msgStyle, msgText)
   If Response = vbCancel Then    ' User chose Cancel so we kill the process
      Exit Sub
   ElseIf Response = vbYes Then  ' convert
     Selection.Rows.convertToText Separator:=wdSeparateByTabs, NestedTables:= _
      True
   Else
   ' user picked no, so go to the next table
   End If
 Next mytable

End Sub
```

* Obtain permission before forwarding. Pix: http://bit.ly/7fUx9O
* Stuff: http://bit.ly/5PONhz FAQ: http://bit.ly/4ocsZ2
* Sub: http://bit.ly/8dsD41
* Job opportunity messages must be tagged "JOB-OP:"

```
##########################################################
Dim theWord As String
For R = 1 To 14
  Set theLine = ActiveDocument.Paragraphs(R).Range
  For c = 1 To 8
    theWord = theLine.Words(2 * c - 1)
    Set rng = ActiveDocument.Range
    With rng.Find
      .ClearFormatting
      .Replacement.ClearFormatting
      .Text = "ZCZC" & theWord
      .Replacement.Text = "D" & Trim(Str(R)) & "-" & Trim(Str(c))
      .MatchWildcards = False
      .Execute Replace:=wdReplaceOne
    End With

  Next c
Next R

##########################################################
Set wordList = ActiveDocument
wordList.Windows(1).WindowState = wdWindowStateMaximize

##########################################################

Sub Docalyse()
' Version 03.03.11
```

```
myDocStats = False

'(c)2009-11 Paul Beverley; Version dated 03.03.2011
' http://www.archivepub.co.uk/FRedit
' Paul Beverley, Nutwood, Middle Road, Norwich, NR13 5EG
' Paul@archivepub.co.uk


doFRedit = True

' Don't highlight these characters:
myExceptions = "—•º£" & Chr(145) & Chr(146) & Chr(147) & Chr(148)

' listFile = "C:\Documents and Settings\Paul\My Documents\" _
    & "DocAlyseList.doc"

listFile = "C:\Program Files\VirtualAcorn\VirtualRPC-SA" _
    & "\HardDisc4\MyFiles2\WIP\zzzTheBook\zFRedit\zzAssess.doc"

If doFRedit = False Then GoTo nextbit
' Open the FRedit list just to test if it's there
Documents.Open listFile
ActiveDocument.Close SaveChanges:=False

' Copy the original text
nextbit:
ActiveDocument.Range.Copy
ActiveDocument.TrackRevisions = False

' First assessment document
Documents.Add
ActiveDocument.TrackRevisions = False
Selection.WholeStory
Selection.LanguageID = wdEnglishUK
Selection.NoProofing = False
Application.CheckLanguage = False

Set firstDoc = ActiveDocument
Selection.TypeText Text:=vbCrLf & vbCrLf

Selection.Paste
WordBasic.AcceptAllChangesInDoc

' Second assessment document
Documents.Add
Set secondDoc = ActiveDocument
Selection.Paste
WordBasic.AcceptAllChangesInDoc
If ActiveDocument.Comments.Count > 0 Then ActiveDocument.DeleteAllComments
ActiveDocument.Range.HighlightColorIndex = wdNoHighlight
Selection.WholeStory
Selection.LanguageID = wdEnglishUK
Selection.NoProofing = False
Application.CheckLanguage = False

' Create a workspace
Documents.Add
```

```
Set scrapPad = ActiveDocument

' Go back to first assessment document
firstDoc.Activate
Selection.HomeKey Unit:=wdStory


' Count various features
thisMany = ActiveDocument.Comments.Count
If thisMany > 0 Then
  Selection.TypeText Text:="Number of comments:  " _
    & thisMany & vbCrLf & vbCrLf
  ActiveDocument.DeleteAllComments
End If

fNotes = ActiveDocument.Footnotes.Count
If fNotes > 0 Then
  Selection.TypeText Text:="Number of footnotes:  " _
    & fNotes & vbCrLf & vbCrLf
End If

eNotes = ActiveDocument.Endnotes.Count
If eNotes > 0 Then
  Selection.TypeText Text:="Number of endnotes:  " _
    & eNotes & vbCrLf & vbCrLf
End If

bMarks = ActiveDocument.Bookmarks.Count
If bMarks > 0 Then
  Selection.TypeText Text:="Number of bookmarks:  " _
    & bMarks & vbCrLf & vbCrLf
  For Each myBM In ActiveDocument.Bookmarks
    myBM.Delete
  Next
End If

tboxes = ActiveDocument.Shapes.Count
' These get counted later

thisMany = ActiveDocument.Lists.Count
If thisMany > 0 Then
  Selection.TypeText Text:="Number of lists:  " _
    & thisMany & vbCrLf & vbCrLf
End If

thisMany = ActiveDocument.Tables.Count
If thisMany > 0 Then
  Selection.TypeText Text:="Number of tables:  " _
    & thisMany & vbCrLf & vbCrLf
End If

thisMany = ActiveDocument.Fields.Count
If thisMany > 0 Then
  Selection.TypeText Text:="Number of fields:  " _
    & thisMany & vbCrLf & vbCrLf
End If
```

```
thisMany = ActiveDocument.FormFields.Count
If thisMany > 0 Then
  Selection.TypeText Text:="Number of form fields:  " _
    & thisMany & vbCrLf & vbCrLf
End If

thisMany = ActiveDocument.Hyperlinks.Count
If thisMany > 0 Then
  Selection.TypeText Text:="Number of hyperlinks:  " _
    & thisMany & vbCrLf & vbCrLf
End If

thisMany = ActiveDocument.Indexes.Count
If thisMany > 0 Then
  Selection.TypeText Text:="Number of indexes:  " _
    & thisMany & vbCrLf & vbCrLf
End If

thisMany = ActiveDocument.SmartTags.Count
If thisMany > 0 Then
  Selection.TypeText Text:="Number of smart tags:  " _
      & thisMany & vbCrLf & _
      "(Tools--Autocorrect Options--Smart Tags" & _
      "--Remove Smart Tags)" & vbCrLf & vbCrLf
End If

thisMany = ActiveDocument.InlineShapes.Count
If thisMany > 0 Then
  Selection.TypeText Text:="Number of equations:  " _
    & thisMany & vbCrLf & vbCrLf
End If

' Record all the readability stats
If myDocStats = True Then
  DocStats = "aazz Readability Statistics" & vbCrLf
  With ActiveDocument.Content
    For i = 1 To 10
      DocStats = DocStats & .ReadabilityStatistics(i) & ": " _
        & .ReadabilityStatistics(i).Value & vbCrLf
    Next i
  End With
End If
Selection.TypeText Text:="aazz Quote characters" & vbCrLf

' Count the punctuation: first open single quotes
Set rng = ActiveDocument.Range
thisMany = -1
Do
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .MatchWildcards = False
    .Text = "^0145"
    .Replacement.Text = ""
    .Execute
  End With
  thisMany = thisMany + 1
```

```
Loop Until rng.Find.Found = False

Selection.TypeText Text:="Single open:  " _
    & thisMany & vbCrLf
ordinary = countIt

' ...closed single quotes
Set rng = ActiveDocument.Range
thisMany = -1
Do
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .MatchWildcards = False
    .Text = "^0146"
    .Replacement.Text = ""
    .Execute
  End With
  thisMany = thisMany + 1
Loop Until rng.Find.Found = False

Selection.TypeText Text:="Single closed (= apostrophe):  " _
    & thisMany & vbCrLf & vbCrLf
ordinary = countIt

' ...open double quotes
Set rng = ActiveDocument.Range
thisMany = -1
Do
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .MatchWildcards = False
    .Text = "^0147"
    .Replacement.Text = ""
    .Execute
  End With
  thisMany = thisMany + 1
Loop Until rng.Find.Found = False

Selection.TypeText Text:="Double open:  " _
    & thisMany & vbCrLf
ordinary = countIt

' ...closed double quotes
Set rng = ActiveDocument.Range
thisMany = -1
Do
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .MatchWildcards = False
    .Text = "^0148"
    .Replacement.Text = ""
    .Execute
  End With
  thisMany = thisMany + 1
```

```
Loop Until rng.Find.Found = False

Selection.TypeText Text:="Double closed:  " _
    & thisMany & vbCrLf & vbCrLf
ordinary = countIt

' ...unsexed double quotes

Set rng = ActiveDocument.Range
thisMany = -1
Do
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .MatchWildcards = False
    .Text = "^39"
    .Replacement.Text = ""
    .Execute
  End With
  thisMany = thisMany + 1
Loop Until rng.Find.Found = False

Selection.TypeText Text:="Single straight:  " _
    & thisMany & vbCrLf
ordinary = countIt

' ...unsexed single quotes
Set rng = ActiveDocument.Range
thisMany = -1
Do
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .MatchWildcards = False
    .Text = "^34"
    .Replacement.Text = ""
    .Execute
  End With
  thisMany = thisMany + 1
Loop Until rng.Find.Found = False

Selection.TypeText Text:="Double straight:  " _
    & thisMany & vbCrLf
ordinary = countIt

' Check to see what styles are used
allStyles = "aazz Styles used:" & vbCrLf
someStyles = False

StatusBar = "Press Ctrl-Break to stop.      " & _
     "Checking styles."

' If necessary, check footnotes, then endnotes,
' then textboxes.
countShapes = 0
For hit = 1 To 4
  If hit = 1 Then
```

```
    If fNotes > 0 Then
      goes = fNotes
    Else
      hit = 2
    End If
  End If
  If hit = 2 Then
    If eNotes > 0 Then
      goes = eNotes
    Else
      hit = 3
    End If
  End If
  If hit = 3 Then
    If tboxes > 0 Then
      goes = tboxes
    Else
      hit = 4
    End If
  End If
  If hit = 4 Then goes = 1
  For myGo = 1 To goes
    If hit = 1 Then
      Set rng = ActiveDocument.Footnotes(myGo).Range
    End If
    If hit = 2 Then
      Set rng = ActiveDocument.Endnotes(myGo).Range
    End If
    If hit = 3 Then
      Do
      ' Only check the text box if it has any text in it
        someText = ActiveDocument.Shapes(myGo).TextFrame.HasText
        If someText Then
          Set rng = ActiveDocument.Shapes(myGo).TextFrame.TextRange
        Else
          countShapes = countShapes + 1
          myGo = myGo + 1
        End If
      Loop Until someText Or myGo > goes
    End If
    If myGo > goes Then Exit For
    If hit = 4 Then
      Set rng = ActiveDocument.Range
    End If

' Remove all the highlighting from all text in normal style
    rng.HighlightColorIndex = wdYellow
    With rng.Find
      .ClearFormatting
      .Replacement.ClearFormatting
      .MatchWildcards = False
      .Text = ""
      .Style = "Normal"
      .Replacement.Text = ""
      .Replacement.Highlight = False
      .Execute Replace:=wdReplaceAll
    End With
```

```
    Do
      rng.Start = 0: rng.End = 0
      With rng.Find
        .ClearFormatting
        .Replacement.ClearFormatting
        .MatchWildcards = False
        .Text = "^p"
        .Highlight = True
        .Replacement.Text = "^p"
        .Replacement.Highlight = False
        .Execute
      End With
      If rng.Find.Found Then
        someStyles = True
        thisStyle = rng.Style
        rng.Start = 0: rng.End = 0
        With rng.Find
          .ClearFormatting
          .Replacement.ClearFormatting
          .MatchWildcards = False
          .Text = ""
          .Style = thisStyle
          .Replacement.Text = ""
          .Replacement.Highlight = False
          .Execute Replace:=wdReplaceAll
        End With
        If InStr(allStyles, thisStyle) = 0 Then _
            allStyles = allStyles & thisStyle & vbCrLf
      Else
        thisStyle = ""
      End If
    Loop Until thisStyle = ""
  Next myGo
Next hit

tboxes = tboxes - countShapes
If tboxes > 0 Then
  Selection.TypeText Text:="Number of text boxes:  " _
    & tboxes - countShapes & vbCrLf & vbCrLf
End If
If countShapes > 0 Then
  Selection.TypeText Text:="Number of shapes:  " _
    & countShapes & vbCrLf & vbCrLf
End If


' Highlight font attributes and any 'funny' characters
' that are not those in the exception list above.

anyBold = False
anyItalic = False
anySuper = False
anySub = False
anySymbol = False
anyUline = False
anyAtAll = False
anyColour = False
```

```
charStyles = "Normal"

For hit = 1 To 4
  If hit = 1 Then
    If fNotes > 0 Then
      goes = fNotes
    Else
      hit = 2
    End If
  End If
  If hit = 2 Then
    If eNotes > 0 Then
      goes = eNotes
    Else
      hit = 3
    End If
  End If
  If hit = 3 Then
    If tboxes > 0 Then
      goes = tboxes
    Else
      hit = 4
    End If
  End If
  If hit = 4 Then goes = 1
  For myGo = 1 To goes
    If hit = 1 Then
      Set rngThis = ActiveDocument.Footnotes(myGo).Range
      checkIt = "Footnotes: "
      myJump = 1
      rngThis.Start = rngThis.Start
      rngThis.Cut
      Set rng = scrapPad.Range
      rng.Paste
      Set rng = scrapPad.Range
   End If
    If hit = 2 Then
      Set rng = ActiveDocument.Endnotes(myGo).Range
      checkIt = "Endnotes (can be a bit slow): "
      myJump = 10
    End If
    If hit = 3 Then
      Do
        someText = ActiveDocument.Shapes(myGo).TextFrame.HasText
        If someText Then
          Set rng = ActiveDocument.Shapes(myGo).TextFrame.TextRange
          checkIt = "Text boxes: "
          myJump = 10
        Else
          myGo = myGo + 1
        End If
      Loop Until someText Or myGo > goes
    End If
    If myGo > goes Then Exit For
    If hit = 4 Then
      Set rng = ActiveDocument.Range
      checkIt = "Finally the main text: "
```

```
      myJump = 100
    End If

    startChar = rng.Start
    EndChar = rng.End
    rng.HighlightColorIndex = wdNoHighlight
    With rng.Find
      .ClearFormatting
      .Text = ""
      .Font.Name = "Symbol"
      .Execute
    End With
    If rng.Find.Found Then anySymbol = True

' The colours used for the different attributes are
' set below and can be changed to taste.

    For chars = startChar + 1 To EndChar
      rng.Start = chars - 1
      rng.End = chars
      myText = rng
      If myText = "" Then myText = " "
      If Asc(myText) > 128 And InStr(myExceptions, myText) = 0 Then
        rng.HighlightColorIndex = wdYellow
      End If
      If rng.Font.Name = "Symbol" Then
        rng.HighlightColorIndex = wdBrightGreen
        anySymbol = True
      End If
      If rng.Font.Superscript Then
        rng.HighlightColorIndex = wdPink
        anySuper = True
      End If
      If rng.Font.Subscript Then
        rng.HighlightColorIndex = wdPink
        anySub = True
      End If
      If rng.Font.Bold = True Then
        rng.HighlightColorIndex = wdTurquoise
        anyBold = True
      End If
      If rng.Font.Italic = True Then
        rng.HighlightColorIndex = wdTurquoise
        anyItalic = True
      End If
      If rng.Font.Underline = True Then
        rng.HighlightColorIndex = wdRed
        anyUline = True
      End If
'rng.Select
'dksdhl = rng.Start
'zkl = rng.End
' Here we see if any character has a style attached to it.
' If it's not a paragraph style, it must be a character style.

      If rng.Font.Color > 0 Then anyColour = True
      thisStyle = rng.Style
```

```
      If InStr(allStyles, thisStyle) = 0 And _
          InStr(charStyles, thisStyle) = 0 Then _
         charStyles = charStyles & thisStyle & vbCrLf
      charLeft = EndChar - chars

' Let the user know how things are progressing
      If charLeft Mod myJump = 0 Then
        If hit < 4 Then
          StatusBar = "Press Ctrl-Break to stop.      " _
              & checkIt & goes - myGo + 1 & " — " & charLeft
        Else
          StatusBar = "Press <trl-Break to stop.      " _
              & "Finally the main text: " & " — " & charLeft
        End If
      End If
    Next chars
    If hit = 1 Then
      rng.Start = 0
      rng.Cut
      rngThis.Paste
    End If
  Next myGo
Next hit

StatusBar = ""


' Sort the style names into alphabetical order
' and display the results.

Selection.HomeKey Unit:=wdStory
If someStyles = True Then
  Set rng = ActiveDocument.Range
  Selection.TypeText Text:=allStyles & vbCrLf
  rng.End = Selection.End
  rng.ConvertToTable
  rng.Sort CaseSensitive:=False
  rng.Rows.ConvertToText
Else
  Selection.TypeText Text:="aazz No styles used!" & vbCrLf
End If


If Len(charStyles) > 8 Then
  rng.Start = Selection.Start
  typeIt = "aazz Character styles used:" & vbCrLf
  Selection.TypeText typeIt
  charStyles = Right(charStyles, Len(charStyles) - 6)
  Selection.TypeText charStyles & vbCrLf
  rng.End = Selection.End
  rng.ConvertToTable
  rng.Sort CaseSensitive:=False
  rng.Rows.ConvertToText
End If

' If no attributes at all are found, say so ...
anyAtAll = anyBold Or anyItalic Or anySymbol Or _
```

```
    anyColour Or anyUline
If anyAtAll = False Then
  typeIt = "aazz No font attributes used!" & vbCrLf
  Selection.TypeText typeIt

' ... otherwise list the attributes encountered
Else
  typeIt = vbCrLf & "aazz Font attributes used:" & vbCrLf
  Selection.TypeText typeIt
  If anyBold Then Selection.TypeText ("Bold" & vbCrLf)
  If anyItalic Then Selection.TypeText ("Italic" & vbCrLf)
  If anyUline Then Selection.TypeText ("Underline" & vbCrLf)
  If anySymbol Then Selection.TypeText ("Symbol font" & vbCrLf)
  If anyColour Then Selection.TypeText ("Coloured text" & vbCrLf)
End If
Selection.TypeText vbCrLf & vbCrLf

Selection.TypeText DocStats & vbCrLf & vbCrLf

Selection.HomeKey Unit:=wdStory
' Remove a stray carriage return
' (to do with the sorting of styles)
' Selection.Delete

Set rng = ActiveDocument.Range

' Add a heading style to the results section
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .MatchWildcards = True
  .Text = "aazz (*)^13"
  .Replacement.Text = "\1^p"
  .Replacement.Style = "Heading 2"
  .Execute Replace:=wdReplaceAll
End With

scrapPad.Close SaveChanges:=False

If doFRedit = False Then GoTo lastBit

' Open the FRedit list
Documents.Open listFile
Set theList = ActiveDocument

If tboxes > 0 Then Selection.TypeText Text:="| textboxes = yes" & vbCrLf
If fNotes > 0 Then Selection.TypeText Text:="| footnotes = yes" & vbCrLf
If eNotes > 0 Then Selection.TypeText Text:="| endnotes = yes" & vbCrLf

secondDoc.Activate
Selection.HomeKey Unit:=wdStory

' Run FRedit and close the Fredit list
Call FRedit
theList.Close SaveChanges:=False

lastBit:
```

```
' End with document one on screen
firstDoc.Activate
Selection.HomeKey Unit:=wdStory
End Sub

##########################################################

First version

Sub HighlightLister()
' Version 26.05.11
' List all the highlight colours used



allHighs = ""
mixCol = 9999999
Set rng = ActiveDocument.Range
theEnd = rng.End
' Set rng2 = ActiveDocument.Range
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Highlight = True
  .Text = ""
  .Replacement.Text = ""
  .MatchWildcards = False
End With

wasCol = 0


Do
  rng.Find.Execute
  If rng.Find.Found = True Then
    stopNow = False
    thisCol = rng.HighlightColorIndex
    If thisCol < mixCol Then
'     rng.Select
      Select Case thisCol
        Case 0: 'Do nowt
        Case wdYellow: col = "Yellow"
        Case wdBrightGreen: col = "BrightGreen"
        Case wdGreen: col = " Green"
        Case wdPink: col = "Pink"
        Case wdRed: col = "Red"
        Case wdBlue: col = "Blue"
        Case wdGray25: col = "Gray25"
        Case wdGray50: col = "Gray50"
        Case wdTurquoise: col = "Turquoise"
        Case wdTeal: col = "Teal"
        Case wdDarkBlue: col = "DarkBlue"
        Case wdDarkYellow: col = "DarkYellow"
        Case wdDarkRed: col = "DarkRed"
        Case wdViolet: col = "Violet"
      Case Else
        ch.Select
```

```
        col = "A colour not on the list!"
      End Select
      If InStr(allHighs, col) = 0 Then allHighs = allHighs & col & ","
      wasCol = thisCol

    Else
      For Each wd In rng.Words
'           wd.Select
        thisCol = wd.HighlightColorIndex
        If thisCol < mixCol Then
          Select Case thisCol
            Case 0: 'Do nowt
            Case wdYellow: col = "Yellow"
            Case wdBrightGreen: col = "BrightGreen"
            Case wdGreen: col = " Green"
            Case wdPink: col = "Pink"
            Case wdRed: col = "Red"
            Case wdBlue: col = "Blue"
            Case wdGray25: col = "Gray25"
            Case wdGray50: col = "Gray50"
            Case wdTurquoise: col = "Turquoise"
            Case wdTeal: col = "Teal"
            Case wdDarkBlue: col = "DarkBlue"
            Case wdDarkYellow: col = "DarkYellow"
            Case wdDarkRed: col = "DarkRed"
            Case wdViolet: col = "Violet"
          Case Else
            ch.Select
            col = "A colour not on the list!"
          End Select
          If InStr(allHighs, col) = 0 Then allHighs = allHighs & col & ","
          wasCol = thisCol
        Else
          wasCol = 0
          For Each ch In wd.Characters
'             ch.Select
            thisCol = ch.HighlightColorIndex
            If thisCol <> wasCol Then
              Select Case thisCol
                Case 0: 'Do nowt
                Case wdYellow: col = "Yellow"
                Case wdBrightGreen: col = "BrightGreen"
                Case wdGreen: col = " Green"
                Case wdPink: col = "Pink"
                Case wdRed: col = "Red"
                Case wdBlue: col = "Blue"
                Case wdGray25: col = "Gray25"
                Case wdGray50: col = "Gray50"
                Case wdTurquoise: col = "Turquoise"
                Case wdTeal: col = "Teal"
                Case wdDarkBlue: col = "DarkBlue"
                Case wdDarkYellow: col = "DarkYellow"
                Case wdDarkRed: col = "DarkRed"
                Case wdViolet: col = "Violet"
              Case Else
                ch.Select
                col = "A colour not on the list!"
```

```
            End Select
            If InStr(allHighs, col) = 0 Then allHighs = allHighs & col & ","
            wasCol = thisCol
          End If
        Next ch
        wasCol = 0
       End If
     Next wd
     wasCol = 0
   End If
'    StatusBar = "Characters to go: " & Str(theEnd - rng.End)
  Else
    stopNow = True
  End If
Loop Until stopNow = True



Selection.HomeKey Unit:=wdStory
Selection.TypeText Text:=Replace(allHighs, ",", vbCrLf)
Selection.Start = 0
Selection.Style = "Normal"
Selection.Sort
Selection.End = 0

Exit Sub

#########################################################

Second version

Sub HighlightLister()
' Version 26.05.11
' List all the highlight colours used
Dim gotCol(99) As Boolean

mixCol = 9999999
Set rng = ActiveDocument.Range
theEnd = rng.End
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Highlight = True
  .Text = ""
  .Replacement.Text = ""
  .MatchWildcards = False
End With


Do
  rng.Find.Execute
  If rng.Find.Found = True Then
    stopNow = False
    thisCol = rng.HighlightColorIndex
    If thisCol < mixCol Then
  '    rng.Select
      gotCol(thisCol) = True
```

```
    Else
      For Each wd In rng.Words
  '         wd.Select
        thisCol = wd.HighlightColorIndex
        If thisCol < mixCol Then
          gotCol(thisCol) = True
        Else
          For Each ch In wd.Characters
    '         ch.Select
            thisCol = ch.HighlightColorIndex
            gotCol(thisCol) = True
          Next ch
        End If
      Next wd
    End If
'    StatusBar = "Characters to go: " & Str(theEnd - rng.End)
  Else
    stopNow = True
  End If
Loop Until stopNow = True

Selection.HomeKey Unit:=wdStory
For i = 1 To 16
  If gotCol(i) = True Then Selection.TypeText Text:=Str(i) & vbCrLf
Next i

End Sub

###########################################################
Version 3

Sub Test()
' Sub HighlightLister()
' Version 26.05.11
' List all the highlight colours used
Dim gotCol(99) As Boolean

mixCol = 9999999
Set rng = ActiveDocument.Range
theEnd = rng.End
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Highlight = True
  .Text = ""
  .Replacement.Text = ""
  .MatchWildcards = False
End With

Dim gotCol(99) As Boolean

Do
  rng.Find.Execute
  If rng.Find.Found = True Then
    stopNow = False
    thisCol = rng.HighlightColorIndex
    If thisCol < mixCol Then
```

```
'      rng.Select
       gotCol(thisCol) = True
     Else
       For Each ch In rng.Characters
         thisCol = ch.HighlightColorIndex
         gotCol(thisCol) = True
       Next ch
 '             rng.Select
     End If
'    StatusBar = "Characters to go: " & Str(theEnd - rng.End)
   Else
     stopNow = True
   End If
Loop Until stopNow = True

Selection.HomeKey Unit:=wdStory
For i = 1 To 16
  If gotCol(i) = True Then Selection.TypeText Text:=Str(i) & vbCrLf
Next i

Exit Sub
Selection.TypeText Text:=Replace(allHighs, ",", vbCrLf)
Selection.Start = 0
Selection.Style = "Normal"
Selection.Sort
Selection.End = 0



Exit Sub




' Keeeeeepppp ttthhhiisss
Selection.HomeKey Unit:=wdStory

Selection.Paragraphs(1).Range.Select

Selection.Sentences(1).Select

Selection.Words(1).Select
endChars = Right(Selection, 2)
If Asc(endChars) = 148 Or Asc(endChars) = 146 Then
  Selection.MoveEnd , -2
End If

' Keeeeeepppp ttthhhiisss
' Find the para numbers of first and last number
i = 0
totParas = ActiveDocument.Paragraphs.Count
For Each para In ActiveDocument.Paragraphs
  i = i + 1
  If para.Range.Start = startHere Then firstPara = i
  If para.Range.Start = startHere Then lastPara = i: Exit For
Next para
```

```
' ActiveDocument.Paragraphs(firstPara).Range.Select


End Sub


#######################################################
Sub MultiChoiceRandomize()
' Version 25.05.11
' Randomize four-choice questions and answers
myColor = wdTurquoise

oldColour = Options.DefaultHighlightColorIndex
Options.DefaultHighlightColorIndex = myColor
myResponse = MsgBox("Cursor in question file?", vbQuestion + vbYesNo)
If myResponse = vbNo Then Exit Sub
qMax = InputBox("How many questions?", "MultiChoice Randomize", "")
If qMax = 0 Then Exit Sub


Selection.HomeKey Unit:=wdStory
Set qnFile = ActiveDocument
For Each doc In Documents
  If Left(doc.Name, 6) <> "Normal" And doc.Name <> ActiveDocument.Name Then
    Set ansFile = doc
    Exit For
  End If
Next doc

ansFile.Activate
Selection.HomeKey Unit:=wdStory

Randomize
For i = 1 To qMax
' Choose random number
  rNum = ((Rnd * 4) Mod 4)

  If rNum > 0 Then
    qnFile.Activate
  ' Find the question
    With Selection.Find
     .ClearFormatting
     .Replacement.ClearFormatting
     .Text = "^p" & Trim(Str(i)) & ". "
     .Replacement.Text = ""
     .MatchWildcards = False
     .Execute
    End With
    Selection.Start = Selection.End
  ' Find the A answer
    With Selection.Find
     .ClearFormatting
     .Replacement.ClearFormatting
     .Text = "^pA. "
     .Replacement.Text = ""
     .MatchWildcards = False
     .Execute
```

```
  End With
  startQn = Selection.Start + 1
  Selection.Start = Selection.End
  Selection.Paragraphs(1).Range.Select
  Selection.Range.HighlightColorIndex = myColor
  Selection.Cut
' Move down rnd steps & paste it
  Selection.MoveDown Unit:=wdParagraph, Count:=rNum
  Selection.Paste

  'Move back up to (new) first answer & re-letter the options
  Selection.End = startQn + 1
  Selection.Start = startQn
  For j = 1 To rNum + 1
    Selection.TypeText Text:=Chr(64 + j)
    Selection.MoveDown Unit:=wdParagraph, Count:=1
    Selection.End = Selection.Start + 1
  Next j

' Move to answer file & find question number
  ansFile.Activate
  With Selection.Find
   .ClearFormatting
   .Replacement.ClearFormatting
   .Text = "^p" & Trim(Str(i)) & ". "
   .Replacement.Text = ""
   .MatchWildcards = False
   .Execute
  End With
  Selection.Start = Selection.End

  With Selection.Find
   .ClearFormatting
   .Replacement.ClearFormatting
   .Text = "^pA. "
   .Replacement.Text = ""
   .MatchWildcards = False
   .Execute
  End With
  startQn = Selection.Start + 1
  Selection.Start = Selection.End

  With Selection.Find
   .ClearFormatting
   .Replacement.ClearFormatting
   .Text = "^pB. "
   .Replacement.Text = ""
   .MatchWildcards = False
   .Execute
  End With
  Selection.End = Selection.Start + 1
  Selection.Start = startQn
' Cut the A answer
  Selection.Range.HighlightColorIndex = myColor
  Selection.Cut

' Select where to put it
```

```
    Select Case rNum
      Case 1: myText = "^pC. "
      Case 2: myText = "^pD. "
      Case 3: myText = "^pContent"
    End Select

    With Selection.Find
     .ClearFormatting
     .Replacement.ClearFormatting
     .Text = myText
     .Replacement.Text = ""
     .MatchWildcards = False
     .Execute
    End With
    Selection.End = Selection.Start + 1
    Selection.Start = Selection.End
    Selection.Paste

    'Move back up to first answer and re-letter them
    Selection.End = startQn - 1
    Selection.Start = startQn - 1
    For j = 1 To rNum + 1
      With Selection.Find
       .ClearFormatting
       .Replacement.ClearFormatting
       .Text = "^13[ABCD]. "
       .Replacement.Text = ""
       .MatchWildcards = True
       .Execute
      End With
      Selection.Start = Selection.Start + 1
      Selection.End = Selection.Start + 1
      Selection.TypeText Text:=Chr(64 + j)
    Next j
  End If
Next i
Selection.HomeKey Unit:=wdStory
qnFile.Activate
Selection.HomeKey Unit:=wdStory
Options.DefaultHighlightColorIndex = oldColour
End Sub

#########################################################
Random numbers
Dim tots(4)
For i = 1 To 13000
a = Rnd
b = ((Rnd * 4) Mod 4)
tots(b) = tots(b) + 1

Next i
a = tots(0)
b = tots(1)
c = tots(2)
d = tots(3)
ie = tots(4)
bnc = 0
```

```
Exit Sub

#####################################################

Jane Ward's macro: superscript <> square brackets

Sub BIBbracket()
' Version 21.05.11
' Add brackets and de-superscript
For Each wd In ActiveDocument.Words
 thisWD = wd
 BIBhere = InStr(thisWD, "BIB")
 If BIBhere > 0 Then
   wd.Select
   Selection.Start = Selection.Start + BIBhere
   BIBstart = Selection.Start
   If Selection.Font.Superscript = True Then
   ' We've found a superscript BIB without []
     Do
       Selection.MoveStart wdCharacter, -1
       Num = Asc(Selection)
     Loop Until Num < 48 Or Num > 57
     Selection.MoveStart wdCharacter, 1
     startHere = Selection.Start
     Selection.InsertBefore Text:="["
     Selection.MoveRight Unit:=wdWord, Count:=3
     thisChar = Selection
     Selection.MoveStart wdCharacter, 1
     Selection.TypeText Text:="]"
     Selection.MoveLeft Unit:=wdWord, Count:=1
     Selection.Delete
     Selection.MoveRight Unit:=wdWord, Count:=1
     Selection.TypeText Text:=thisChar
     Selection.Start = startHere
     Selection.Font.Superscript = False
     Selection.Start = Selection.End
     If Asc(Selection) = 150 Then
       Selection.MoveEnd wdCharacter, 1
       Selection.Font.Superscript = False
     End If
   Else
   ' We've found a superscript BIB WITH []
     Selection.Start = Selection.End
     Do
       Selection.MoveRight Unit:=wdCharacter, Count:=1
     Loop Until Asc(Selection) = Asc("]")
     Selection.Start = BIBstart
     Selection.Font.Superscript = False
   End If
 End If
Next wd
End Sub



#####################################################
```

```
' Find the para numbers of first and last number
i = 0
totParas = ActiveDocument.Paragraphs.Count
For Each para In ActiveDocument.Paragraphs
  i = i + 1
  If para.Range.Start = startHere Then firstPara = i
  If para.Range.Start = startHere Then lastPara = i: Exit For
Next para
' ActiveDocument.Paragraphs(firstPara).Range.Select
```

##

Fiddling with URLs

```
Selection.HomeKey Unit:=wdStory
gotOne = False
Do
  With Selection.Find
   .ClearFormatting
   .Replacement.ClearFormatting
   .Text = "[hw][tw][tw][p.]"
   .Wrap = False
   .Replacement.Text = ""
   .MatchWildcards = True
   .Execute
  End With
  gotOne = Selection.Find.Found

  If gotOne = True And Selection.Font.Color <> wdColorBlue Then
    urlStart = Selection.Start
  ' Find the end of the URL
    Selection.Start = Selection.End
    With Selection.Find
     .ClearFormatting
     .Replacement.ClearFormatting
     .Text = "[^13^32,^2]"
     .Wrap = False
     .Replacement.Text = ""
     .Forward = True
     .MatchWildcards = True
     .Execute
    End With
    Selection.Start = urlStart
    Selection.End = Selection.End - 1
    myText = Selection
    ActiveDocument.Hyperlinks.Add Anchor:=Selection.Range, _
        Address:=myText, TextToDisplay:=myText
    Selection.End = Selection.End + 2
    Selection.Start = Selection.End
  End If
Loop Until gotOne = False
```

##

Date: Tue, 05 Apr 2011 11:49:42 +1200
Subject: Re: New challenge - try this macro...
To: Paul Beverley <paul@archivepub.co.uk>

Hi Paul

Thanks again for your assistance.

By adapting your two recent macros I've now worked out a combination one that deals with the several issues I wanted to correct: first, delete hyperlinks from any non-URL / non-email addresses, such as author names; second, fix up existing hyperlinks by deleting http from the text of 'http+www' URLs (publisher's style) and adding http to the address of URLs where it was missing (I sometimes find that the hyperlink address defines an absolute path to a document on the author's hard drive rather to the actual URL); third add hyperlinks to URLs that don't have them - in the same style as the preceding.

FYI I have attached my final FixURLs macro, together with the list of random URLs that I used to test it out.

By following your macros I think I should be able to adapt this one to handle endnotes, but will sing out if I get stuck.

Happy 'concreting'

Thiers

```
Sub FixURLs()

'Delete and highlight hyperlinks that are not URLs or email addresses
myColour = 0
myColour = wdGray25
linksTotal = ActiveDocument.Hyperlinks.Count
If linksTotal > 0 Then
  For i = linksTotal To 1 Step -1
    myText = ActiveDocument.Hyperlinks(i).TextToDisplay
    If InStr(myText, "www") = 0 And InStr(myText, "http") = 0 _
        And InStr(myAddress, "mailto") = 0 Then
      Set rng = ActiveDocument.Hyperlinks(i).Range
      If myColour > 0 Then rng.HighlightColorIndex = myColour
      ActiveDocument.Hyperlinks(i).Delete
    End If
  Next i
End If


'Fix existing hyperlinked URLs
linksTotal = ActiveDocument.Hyperlinks.Count
If linksTotal > 0 Then
  For i = linksTotal To 1 Step -1
    myText = ActiveDocument.Hyperlinks(i).TextToDisplay
    'delete http from http://www TextToDisplay
    If InStr(myText, "www") > 0 And InStr(myText, "http") > 0 Then
       ActiveDocument.Hyperlinks(i).TextToDisplay = Mid$(myText, 8, Len(myText))
    Else
    'add http to www Address if not present
    If InStr(myText, "www") > 0 And InStr(myText, "http") = 0 Then
       ActiveDocument.Hyperlinks(i).Address = "http://" & myText
    End If
```

```
    End If
  Next i
End If


'Hyperlink any unlinked URLs
Selection.HomeKey Unit:=wdStory
gotOne = False
Do
  With Selection.Find
   .ClearFormatting
   .Replacement.ClearFormatting
   .Text = "[hw][tw][tw][p.]"
   .Wrap = False
   .Replacement.Text = ""
   .MatchWildcards = True
   .Execute
  End With
  gotOne = Selection.Find.Found

  If gotOne = True And Selection.Font.Color <> wdColorBlue Then
    urlStart = Selection.Start
 ' Find the end of the URL
    Selection.Start = Selection.End
    With Selection.Find
     .ClearFormatting
     .Replacement.ClearFormatting
     .Text = "[^13^32,^2]"
     .Wrap = False
     .Replacement.Text = ""
     .Forward = True
     .MatchWildcards = True
     .Execute
    End With
    Selection.Start = urlStart
    Selection.End = Selection.End - 1
    myText = Selection
       'link http://www URLs
       If InStr(myText, "http") > 0 And InStr(myText, "www") > 0 Then
          strText = Mid$(myText, 8, Len(myText))
          ActiveDocument.Hyperlinks.Add Anchor:=Selection.Range, _
             Address:=myText, TextToDisplay:=strText
       Else
       'link www URLs
       If InStr(myText, "http") = 0 And InStr(myText, "www") > 0 Then
          strAddress = "http://" & myText
          ActiveDocument.Hyperlinks.Add Anchor:=Selection.Range, _
             Address:=strAddress, TextToDisplay:=myText
       Else
       'link http:// URLs
          ActiveDocument.Hyperlinks.Add Anchor:=Selection.Range, _
             Address:=myText, TextToDisplay:=myText
       End If
       End If

    Selection.End = Selection.End + 2
    Selection.Start = Selection.End
```

```
  End If
Loop Until gotOne = False

End Sub
```

Test URLS

W = www URL
H = http URL
HW = http://www URL
- = without hyperlink + = with hyperlink

W- www.ihi.org

W- www.number10.gov.uk/news/latest-news/2010/05/big-society-50248

W+ www.ihi.org . (This one retains the absolute address path rather than the true URL address)

W+ www.library.nhs.uk/Improvement/ViewResource.aspx?resID=325326

H- http://24.125.155.132/scholar?q=cache

H+ http://scholar?q=cache:MS45tOwCYzMJ

HW+ http://www.scholar?q=cache:MS45tOwCYzMJ

HW+ http://www.thersa.org

HW- http://www.ihi.org .

HW- http://www.number10.gov.uk/news/latest-news/2010/05/big-society-50248

HW- http://www.scholar?q=cache:MS45tOwCYzMJ:scholar.google.com/

##########

```
On Error Resume Next

' If the file is already opened by another process,
' and the specified type of access is not allowed,
' the Open operation fails and an error occurs.
Open strFileName For Binary Access Read Lock Read As #1
Close #1

' If an error occurs, the document is currently open.
If Err.Number <> 0 Then
   FileLocked = True
   Err.Clear
End If
```

```
#####################################################

Sub TrudeauFandR()
' Version 18.02.11
' Find something specific and do things to each one
Selection.HomeKey Unit:=wdStory
Do
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = "([.\?\!])  ([A-Z])"
    .Wrap = False
    .Replacement.Text = ""
    .Forward = True
    .MatchWildcards = True
    .Execute
  End With

  If Selection.Find.Found = True Then
  ' Move the start of the selection one space right
  ' i.e. just past the punctuation mark
    Selection.Start = Selection.Start + 1
  ' Put the end of the selection one space to the right of this
  ' i.e. select the first space character
    Selection.End = Selection.Start + 1
  ' Delete it
    Selection.Delete
    stopNow = False
  Else
    stopNow = True
  End If
  Selection.Start = Selection.End
Loop Until stopNow = True

Selection.HomeKey Unit:=wdStory
Do
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = "([0-9])-([0-9])"
    .Wrap = False
    .Replacement.Text = ""
    .Forward = True
    .MatchWildcards = True
    .Execute
  End With

  If Selection.Find.Found = True Then
  ' Move the start of the selection one space right
  ' i.e. past the first number and in front of the hyphen
    Selection.Start = Selection.Start + 1
  ' Put the end of the selection one place to the right of this
  ' i.e. select the hyphen
    Selection.End = Selection.Start + 1
  ' Type a dash, which will replace the selected hyphen
    Selection.TypeText Text:=Chr(150)
    stopNow = False
```

```
  Else
    stopNow = True
  End If
  Selection.Start = Selection.End
Loop Until stopNow = True
End Sub

#########################################################




' Find the number of the paragraph
' paraNumber = ActiveDocument.Range(0, Selection.Paragraphs(1).Range.End).Paragraphs.Count
' Select the previous paragraph
' ActiveDocument.Paragraphs(paraNumber - 1).Range.Select
Sub OpenLastJob()
' Version
' Alt-Ctrl-f10

On Error GoTo NoWorries
If ActiveDocument.Name = RecentFiles(1) Then RecentFiles(2).Open
NoWorries:
RecentFiles(1).Open
Application.GoBack


End Sub

#########################################################
Sub DisplayQuote()
' Version 13.10.10
myTrack = ActiveDocument.TrackRevisions
ActiveDocument.TrackRevisions = False
Selection.HomeKey Unit:=wdLine
Selection.TypeText Text:="<DQ>"
Selection.MoveDown Unit:=wdParagraph, Count:=1
Selection.MoveLeft Unit:=wdCharacter, Count:=1
Selection.TypeText Text:="<\DQ>"

Selection.MoveUp Unit:=wdParagraph, Count:=1
Selection.MoveDown Unit:=wdParagraph, Count:=1, Extend:=wdExtend

Selection.Range.Font.Italic = False


With Selection.ParagraphFormat
  .LeftIndent = CentimetersToPoints(0.95)
  .SpaceBeforeAuto = False
  .SpaceAfterAuto = False
End With
With Selection.ParagraphFormat
  .RightIndent = CentimetersToPoints(1)
  .SpaceBeforeAuto = False
  .SpaceAfterAuto = False
End With
```

```
Exit Sub

Selection.MoveUp Unit:=wdParagraph, Count:=1
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = """
  .Forward = True
  .MatchWildcards = False
  .Execute
End With
ActiveDocument.TrackRevisions = myTrack

End Sub

########################################################

Sub PrefixAdder()
' Version 05.02.11
' Add prefix to all section numbers

'docName = ActiveDocument.Name
'i = InStr(docName, "Annex_")
'If i > 0 Then
'  thisPrefix = Mid(docName, i + 6, 1)
'Else
'  thisPrefix = ""
'End If
'Prefix = InputBox("Prefix?", "PrefixAdder", thisPrefix)
'If Prefix = "" Then Exit Sub

myFind = "9."
myPrefix = "D1."

myTrack = ActiveDocument.TrackRevisions
ActiveDocument.TrackRevisions = False

Set rng = ActiveDocument.Content
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "^m"
  .Replacement.Text = "^m^p"
  .Replacement.Highlight = False
  .MatchWildcards = False
  .Execute Replace:=wdReplaceAll
End With

For Each para In ActiveDocument.Paragraphs
  myText = para
  If Left(myText, Len(myFind)) = myFind Then
    para.Range.InsertBefore myPrefix
  End If
Next para
Set rng = ActiveDocument.Content
With rng.Find
  .ClearFormatting
```

```
  .Replacement.ClearFormatting
  .Text = "^m^p"
  .Replacement.Text = "^m"
  .Replacement.Highlight = False
  .MatchWildcards = False
  .Execute Replace:=wdReplaceAll
End With
Beep
ActiveDocument.TrackRevisions = myTrack

End Sub

#########################################################
Sub OpenLastJob()
' Version
'<shift-alt-ctrl-O>

RecentFiles(1).Open
Application.GoBack

End Sub

#########################################################
Sub InstantFindFormat()

' Version 10.09.10
' Ctrl-Shift-Alt-#

hereNow = Selection.Start
isSuper = Selection.Font.Superscript
thisBit = Trim(Selection)
If Selection.End = Selection.Start Then
  thisBit = ""
  Selection.MoveEnd , 1
End If
isSuper = Selection.Font.Superscript
isSub = Selection.Font.Subscript
isItalic = Selection.Font.Italic
isBold = Selection.Font.Bold
Selection.MoveStart , 10
Selection.End = Selection.Start
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Wrap = False
  If isSuper Then .Font.Superscript = True
  If isSub Then .Font.Subscript = True
  If isItalic Then .Font.Italic = True
  If isBold Then .Font.Bold = True
  .Text = thisBit
  .Replacement.Text = thisBit
  .MatchWildcards = False
  .MatchCase = False
  .Forward = True
  .Execute
End With
If Selection.End = hereNow Then Beep
```

```
If Selection.Start = hereNow + 10 Then
  Beep
  Selection.Start = hereNow
  Selection.End = hereNow
End If

' Leave F&R dialogue in a sensible state
Selection.Find.Wrap = wdFindContinue

End Sub

Sub InstantFindFormatUp()

' Version 10.09.10
' Ctrl-Shift-Alt-]

hereNow = Selection.Start
thisBit = Trim(Selection)
If Selection.End = Selection.Start Then
  thisBit = ""
  Selection.MoveEnd , 1
End If
isSuper = Selection.Font.Superscript
isSub = Selection.Font.Subscript
isItalic = Selection.Font.Italic
isBold = Selection.Font.Bold
Selection.MoveStart , -10
Selection.End = Selection.Start
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Wrap = False
  .Forward = False
  If isSuper Then .Font.Superscript = True
  If isSub Then .Font.Subscript = True
  If isItalic Then .Font.Italic = True
  If isBold Then .Font.Bold = True
  .Text = thisBit
  .Replacement.Text = thisBit
  .MatchCase = False
  .MatchWildcards = False
  .Execute
End With
If Selection.Start = hereNow Then Beep
If Selection.Start = hereNow - 10 Then
  Beep
  Selection.Start = hereNow
  Selection.End = hereNow
End If

'Add these two to leave F&R dialogue in a sensible state
Selection.Find.Forward = True
Selection.Find.Wrap = wdFindContinue

End Sub

######################################################
```

```
Sub PreToRepFromTop()
' Version 03.12.10
' Shift-Alt-H
myText = Selection
If Asc(myText) <> 32 Then myText = Trim(myText)
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .MatchWildcards = off
End With

Selection.HomeKey Unit:=wdStory

With Dialogs(wdDialogEditReplace)
  .Find = myText
  .Replace = myText
  .MatchCase = True
  .Execute
End With
CommandBars("Menu Bar").Controls("&Edit").Controls("R&eplace...").Execute
End Sub

Sub PrepToRepTitle()
' Version
'<shift-alt-H>
  myText$ = Trim(Selection)
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .MatchWildcards = off
  End With

  Selection.HomeKey Unit:=wdStory

  On Error GoTo ReportIt

  With Dialogs(wdDialogEditReplace)
    .Find = myText$
    .Replace = Left(myText$, 1) + LCase(Right(myText$, Len(myText$) - 1))
    .MatchCase = True
    .Show
  End With

FinishHere:
Exit Sub

ReportIt:
  MsgBox Err.Description
  Resume FinishHere
End Sub

#######################################################
Sub LinesInACell()
' Version
Dim myCount As Long
myCount = 1
Do
```

```
    Selection.MoveRight Unit:=wdCell, Count:=2
    Selection.MoveLeft Unit:=wdCell
    Selection.MoveLeft Unit:=wdCharacter
    Selection.TypeText Text:="|"
    myCount = myCount + 1
Loop Until myCount = 10
End Sub

#########################################################

Sub MacroDbase()
' Version 18.01.11
oldFind = Selection.Find.Text
oldReplace = Selection.Find.Replacement.Text

Selection.HomeKey Unit:=wdStory
allText = ""
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .MatchWildcards = False
  .Text = "anaerobic 4"
  .MatchCase = True
  .Wrap = False
  .Execute
End With
Selection.Start = Selection.End
Do
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .MatchWildcards = False
    .Text = "^pSub"
    .Font.Name = "Courier New"
    .MatchCase = True
    .Wrap = False
    .Execute
  End With

  If Selection.Find.Found = True Then
    gotOne = True
    Selection.MoveRight Unit:=wdWord, Count:=1
    Selection.MoveRight Unit:=wdWord, Count:=1, Extend:=wdExtend
    mName = Selection
    Selection.MoveRight Unit:=wdWord, Count:=5
    Selection.MoveRight Unit:=wdWord, Count:=1, Extend:=wdExtend
    If Asc(Selection) > 47 And Asc(Selection) < 58 Then
      Selection.MoveRight Unit:=wdWord, Count:=4, Extend:=wdExtend
      mDate = Selection
    Else
      mDate = ""
    End If
    Selection.MoveEnd wdCharacter, 1
    Selection.Start = Selection.End
    Selection.EndKey Unit:=wdLine, Extend:=wdExtend
    If Len(Selection) > 3 Then
      Selection.Start = Selection.Start + 2
```

```
      Selection.End = Selection.End - 1
      mDescrip = Selection
    Else
      mDescrip = "Blah blah blah"
    End If
    Selection.Start = Selection.End
    allText = allText & mName & Chr(9) & mDate & Chr(9) & mDescrip & vbCrLf
    allText = allText & "zz" & mDate & "zz" & mName & Chr(9) & mDate & Chr(9) & mDescrip & vbCrLf
  Else
    gotOne = False
  End If
Loop Until gotOne = False

Documents.Add
Selection.TypeText Text:=allText
Set rng = ActiveDocument.Content

' Switch the date order
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "zz([0-9][0-9]).([0-9][0-9]).([0-9][0-9])zz"
  .Replacement.Text = "zz\3.\2.\1zz"
  .Forward = True
  .Wrap = wdFindContinue
  .Format = False
  .MatchWildcards = True
  .Execute Replace:=wdReplaceAll
End With

Selection.WholeStory
Selection.Sort ExcludeHeader:=False, FieldNumber:="Column 1", _
    SortFieldType:=wdSortFieldAlphanumeric, SortOrder:=wdSortOrderAscending, _
    Separator:=wdSortSeparateByCommas, SortColumn:=False, caseSensitive:=True, _
    SubFieldNumber:="Paragraphs"

Selection.HomeKey Unit:=wdStory
' Find the first zzdatezz
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "zz([0-9][0-9]).([0-9][0-9]).([0-9][0-9])zz"
  .Replacement.Text = ""
  .MatchWildcards = True
  .Execute
End With
Selection.End = Selection.Start
Selection.InsertBefore Text:="Sorted in date order" & vbCrLf
Selection.Style = "Heading 2"

Selection.HomeKey Unit:=wdStory
Selection.TypeText Text:="Alphabetic order"
Selection.MoveUp Unit:=wdLine, Count:=1, Extend:=wdExtend
Selection.Style = "Heading 2"

Set rng = ActiveDocument.Content
```

```
' Remove the reverse dates
With rng.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "zz([0-9][0-9]).([0-9][0-9]).([0-9][0-9])zz"
  .Replacement.Text = ""
  .Forward = True
  .Wrap = wdFindContinue
  .Format = False
  .MatchWildcards = True
  .Execute Replace:=wdReplaceAll
End With

Selection.WholeStory
Selection.ConvertToTable Separator:=wdSeparateByTabs
Selection.Tables(1).Style = "Table Grid"
Selection.Tables(1).AutoFitBehavior (wdAutoFitContent)
With Selection.Find
  .Text = oldFind
  .Replacement.Text = oldReplace
  .MatchWildcards = False
End With
Selection.HomeKey Unit:=wdStory
ActiveDocument.SaveAs _
    FileName:="C:\Program Files\VirtualAcorn\VirtualRPC-
SA\HardDisc4\MyFiles2\WIP\zzzTheBook\MacroList.doc", _
    FileFormat:=wdFormatDocument
End Sub

#######################################################
theEnd = ActiveDocument.Content.End
revMax = ActiveDocument.Revisions.Count
For i = 1 To revMax
' For i = revMax To 1 Step -1
  Set rng = ActiveDocument.Revisions.Item(i)
  myType = ActiveDocument.Revisions.Item(i).FormatDescription
  If myType = "Formatted: Not Highlight" Then
    ActiveDocument.Revisions.Item(i).Accept
  End If
StatusBar = "Getting rid of 'Formatted: Not Highlight'..." _
      & Str(revMax - i)
 rng.Select
Next i

' Selection.EndKey Unit:=wdStory
' ActiveWindow.View.ShowRevisionsAndComments = nowShowMarks
ActiveDocument.TrackRevisions = nowTrack
Exit Sub

theEnd = ActiveDocument.Content.End
started = False
For Each rev In ActiveDocument.Range.Revisions
' revMax = ActiveDocument.Revisions.Count
' For i = 1 To revMax
' For i = revMax To 1 Step -1
'  wasStart = rng.Start
'  wasHere = rng.End
```

```
'  Set rng = ActiveDocument.Revisions.Item(i)
  Set rng = rev.Range
' GoTo missIt
'  nowHere = rng.End
'  nowStart = rng.Start
'  If nowHere <> wasHere Then
'  myType = ActiveDocument.Revisions.Item(i).FormatDescription
 myType = rev.FormatDescription
  If myType = "Formatted: Not Highlight" Then rng.Revisions.AcceptAll
'  If myType = "Formatted: Not Highlight" Then
'ActiveDocument.Revisions.Item(i).Accept
'  End If
'missIt:
  StatusBar = "Getting rid of 'Formatted: Not Highlight'..." _
      & Str(theEnd - rng.End)
'StatusBar = "Getting rid of 'Formatted: Not Highlight'..." _
      & Str(revMax - i)
' rng.Select
Next rev



#########################################################

List commands

1. On the Tools menu, point to Macro, and then click Macros.
2. In the Macros in box, click Word commands.
3. In the Macro name box, click ListCommands.
4. Click Run.
5. In the List Commands dialog box, click Current menu and keyboard settings.


#########################################################
Exit Sub
Selection.Range.PasteSpecial DataType:=wdPasteText


On Error GoTo NotText
Set MyData = New DataObject
MyData.GetFromClipboard
strClip = MyData.GetText
Selection.TypeText Text:=strClip

NotText:
If Err <> 0 Then
  MsgBox "Data on clipboard is not text."
End If
Exit Sub


  Selection.PasteAndFormat (wdFormatSurroundingFormattingWithEmphasis)
  Selection.PasteAndFormat (wdFormatPlainText)
  Selection.PasteAndFormat (wdPasteDefault)

Selection.PasteAndFormatSelection.PasteAndFormat (wdPasteUnformatted)
```

```
#####################################################
  ActiveWindow.ActivePane.SmallScroll Down:=2

#####################################################
' Select current word
Selection.Words(1).Select

' Select current paragraph
Selection.Paragraphs(1).Range.Select

' Find the number of the paragraph
MsgBox ActiveDocument.Range(0, Selection.Paragraphs(1).Range.End).Paragraphs.Count


Sub FindParaAndWordNumber ()
' Version 09.11.10
' This finds the number of the current paragraph
' and the current word

Set rng = ActiveDocument.Range
rng.End = Selection.Start
myPara = rng.Paragraphs.Count
ActiveDocument.Paragraphs(myPara).Range.HighlightColorIndex = wdYellow
myWord = rng.Words.Count
ActiveDocument.Words(myWord).HighlightColorIndex = wdRed
Exit Sub


#####################################################
' Sub RenumberSuperscript()
' Version 18.12.10

Selection.HomeKey Unit:=wdStory
i = 1
Do
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .MatchWildcards = False
    .Text = ""
    .Font.Superscript = True
    .Replacement.Text = ""
    .Wrap = False
    .Execute
  End With

  KeepGoing = Selection.Find.Found
  OKchars = "0123456789" & Chr(21)
  ' Chr(21) is because sometimes a rogue end field
  ' gets left behind with the footnote marker
  goodOne = False
  foundText = Selection
  For char = 1 To Len(foundText)
    If InStr(OKchars, Mid(foundText, char, 1)) > 0 Then goodOne = True
  Next char
  If KeepGoing = True And goodOne = True Then
```

```
    Selection.TypeText Text:=Trim(Str(i))
    i = i + 1
  End If
Loop Until KeepGoing = False
Exit Sub

#########################################################
Sub UnRaiser()
' Version 15.12.10
Selection.WholeStory
Selection.Font.Position = 0
End Sub



Sub mandy()
Selection.HomeKey Unit:=wdStory
Do
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .MatchWildcards = False
    .Font.Superscript = True
    .Text = ""
    .Wrap = False
    .Execute
  End With
  gotOne = Selection.Find.Found
  hereNow = Selection.Start
  hereEnd = Selection.End
  actual = Selection
  Selection.MoveStart wdCharacter, -2
  Selection.MoveEnd wdCharacter, 2
  If gotOne = True Then
    thisText = Selection
    For i = 1 To Len(thisText)
    MsgBox (Asc(Mid(thisText, i, 1)))

    Selection.Delete
    Selection.TypeText Text:=actual
    Selection.Start = hereNow
    Selection.Font.Superscript = True
  End If
Loop Until gotOne = False

End Sub


#########################################################
Sub SpellChWordAny()
' Version 06.12.10
Selection.MoveEnd wdWord, 1
Selection.MoveStart wdWord, -1
If CheckSpelling(Selection, _
     MainDictionary:=Languages(Selection.LanguageID).NameLocal) = False Then
  ActiveDocument.CheckSpelling
Else
```

```
  Selection.End = Selection.Start
  beep
End If
End Sub


#####################################################
' Version 12.11.10
myWds = ",that|which,which|that,last|past,past|final,final,previous,previous|last,"
myWds = myWds & "like|such as,Like|As with,than|from,"
myWds = myWds & "England|the UK,Holland|the Netherlands,"
myWds = myWds & "Continuously|Continually,Continually|Continuously,Due|Owing,"
myWds = myWds & "Continuous|Continual,Continual|Continuous,"
myWds = myWds & "continuously|continually,continually|continuously,due|owing,"
myWds = myWds & "continuous|continual,continual|continuous,however|but,"
myWds = myWds & "as|because,an|one,a|one,An|One,A|One,is|are,are|is,to|from,"
myWds = myWds & "°| degrees,may|might,might|may,ad hoc|occasional,so|therefore,"
myWds = myWds & "&|and,instantly|instantaneously,for|because,"
myWds = myWds & "Added|Improved,"
myWds = myWds & "degree|!°,degrees|!°,percent|!%,per cent|!%,"
' Use 'per cent' or 'percent' for UK/US spelling
If Selection.LanguageID = wdEnglishUK Then
  myWds = myWds & "%| per cent,"
Else
  myWds = myWds & "%| percent,"
End If

Selection.End = Selection.Start
If Asc(Selection) = 32 Then
  Selection.MoveStart wdCharacter, 2
  Exit Sub
End If
Selection.MoveLeft Unit:=wdWord, Count:=1
Selection.MoveRight Unit:=wdWord, Count:=1, Extend:=wdExtend
If Right(Selection, 1) = " " Then Selection.MoveEnd , -1
' check if this "word" includes a close single curly quote
If Right(Selection, 1) = Chr(146) Then Selection.MoveEnd , -1
thisWd = Selection

' Look through the list of words
WordPos = InStr(myWds, thisWd & "|")
If WordPos > 0 Then
  myWds = Right(myWds, Len(myWds) - WordPos - Len(thisWd))
  newWd = Left(myWds, InStr(myWds, ",") - 1)
  ' If it starts with "!", delete it and delete
  ' the previous character in the text
  If Asc(newWd) = 33 Then
    newWd = Right(newWd, Len(newWd) - 1)
    Selection.Start = Selection.Start - 1
  End If
  Selection.TypeText Text:=newWd
  Selection.MoveLeft Unit:=wdCharacter, Count:=1
Else
' If no word found, assume that an added 'that' is needed
  Selection.Start = Selection.End
  nextChar = Selection
  If nextChar = "," Then Selection.MoveRight Unit:=wdCharacter, Count:=1
```

```
  Selection.TypeText Text:=" that"
End If

#########################################################

Sub ZtoS()
' Version 24.07.10
' Ctrl-Alt-Shift-z

oldFind = Selection.Find.Text
oldReplace = Selection.Find.Replacement.Text

Selection.End = Selection.Start
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "iz"
  .MatchWildcards = False
  .Execute
End With

Selection.MoveStart , 1

If Selection = "z" Then
  Selection.TypeText Text:="s"
Else
  Selection.TypeText Text:="S"
End If

With Selection.Find
  .Text = oldFind
  .Replacement.Text = oldReplace
End With

End Sub

#########################################################
To: "Paul Beverley" <paul@archivepub.co.uk>
Subject: "Copy Text into the F&R Box" / PrepareToReplace() macro
Date: Thu, 25 Nov 2010 15:40:51 -0500

Hi, Paul.

I tried using your free macro PrepareToReplace() , but it doesn't
seem to work right. Suppose I keep pressing the "Find" button. It
keeps cycling through the document again and again, and only
after the dialog box is manually closed does the message that no
more instances can be found pop up.

I was looking for something else in a collection of macros from
WordTips , and I happened to come upon the technique of launching
a dialog box as a commandbar button rather than as a dialog. I
was then able to make the Find box behave properly.

My code is shown below. It's a bit more involved than your macro
and is designed to be a replacement for the built-in Ctrl+H
keystroke and to offer choices based on whether there is a valid selection,
```

but you can see the technique and borrow from it appropriately to fit with yours.

A couple of lines are commented out because I decided not to be bothered with setting the "Match Case" option.

Regards,
Shmuel

```
Sub ReplaceSelected()
'Show Find/Replace dialog pre-filled with the current selection
'Designed as replacement for Ctrl+H keystroke
'(c) Shmuel Gerber, November 2010
Dim reason As String, myText As String, msg As String, title As String
Dim myChoice As VbMsgBoxResult
   If Selection.Start = Selection.End Then GoTo Built_In       'nothing selected, skip rest of macro
   reason = ""                          'reason for not using selected text
   If Selection.Type <> wdSelectionNormal Then reason = "Not a 'normal' selection."
   If Selection.Range.ComputeStatistics(wdStatisticParagraphs) > 0 _
      Then reason = "Selection includes paragraph break."
   If Selection.End - Selection.Start > 255 Then reason = "Selected text is too long to find."
   If reason <> "" Then
      title = "Find/Replace cannot use selected text"
      msg = reason & " Find/Replace within selection?"
      Select Case MsgBox(msg, vbOKCancel + vbQuestion, title)
         Case vbOK: GoTo Built_In
         Case vbCancel: Exit Sub
      End Select
   End If
   'If none of the above:
   myText = Selection
   title = "Find/Replace selected text?"
   msg = _
      "Yes = Start Find/Replace using selected text:" & vbNewLine & "«" & myText & "»" & vbNewLine &
vbNewLine & _
      "No  = Find/Replace within selection" & vbNewLine & vbNewLine & _
      "Cancel = Continue editing"
   Select Case MsgBox(msg, vbYesNoCancel + vbQuestion, title)
      Case vbNo: GoTo Built_In
      Case vbCancel: Exit Sub
   End Select
   'Last choice was "Yes":
'    myChoice = MsgBox("Match Case?", vbYesNoCancel, "Find/Replace option")
'    If myChoice = vbCancel Then Exit Sub
   'based on Macros4Editors PrepareToReplace() by Paul Beverley:
   Selection.End = Selection.Start     'unselect and search from start of selection
   With Dialogs(wdDialogEditReplace)   'fill in the dialog box fields
      .Find = myText
      .Replace = myText
'       If myChoice = vbYes Then .MatchCase = True
'       If myChoice = vbNo Then .MatchCase = False
      .Execute
   End With
   Selection.End = Selection.Start 'Word selects first instance; unselect again
   GoTo Built_In

   Exit Sub
```

```
Built_In:
    'Show Word's built-in Replace dialog
    CommandBars("Menu Bar").Controls("&Edit").Controls("R&eplace...").Execute
End Sub


-----
No virus found in this message.
Checked by AVG - www.avg.com
Version: 10.0.1170 / Virus Database: 426/3277 - Release Date: 11/24/10



############################################################
Sub zCommonWordSwitch()
' Version 12.11.10
myWds = ",that|which,which|that,last|past,past|final,final,previous,previous|last,"
myWds = myWds & "like|such as,Like|As with,than|from,"
myWds = myWds & "England|the UK,Holland|the Netherlands,"
myWds = myWds & "Continuously|Continually,Continually|Continuously,Due|Owing,"
myWds = myWds & "Continuous|Continual,Continual|Continuous,"
myWds = myWds & "continuously|continually,continually|continuously,due|owing,"
myWds = myWds & "continuous|continual,continual|continuous,however|but,"
myWds = myWds & "as|because,an|one,a|one,An|One,A|One,is|are,are|is,to|from,"
myWds = myWds & "°| degrees,may|might,might|may,ad hoc|occasional,so|therefore,"
myWds = myWds & "&|and,instantly|instantaneously,for|because,"
myWds = myWds & "Added|Improved,"
myWds = myWds & "degree|!°,degrees|!°,percent|!%,per cent|!%,"
' Use 'per cent' or 'percent' for UK/US spelling
If Selection.LanguageID = wdEnglishUK Then
  myWds = myWds & "%| per cent,"
Else
  myWds = myWds & "%| percent,"
End If

Selection.End = Selection.Start
If Asc(Selection) = 32 Then
  Selection.MoveStart wdCharacter, 2
  Exit Sub
End If
Selection.MoveLeft Unit:=wdWord, Count:=1
Selection.MoveRight Unit:=wdWord, Count:=1, Extend:=wdExtend
If Right(Selection, 1) = " " Then Selection.MoveEnd , -1
' check if this "word" includes a close single curly quote
If Right(Selection, 1) = Chr(146) Then Selection.MoveEnd , -1
thisWd = Selection

' Look through the list of words
WordPos = InStr(myWds, thisWd & "|")
If WordPos > 0 Then
  myWds = Right(myWds, Len(myWds) - WordPos - Len(thisWd))
  newWd = Left(myWds, InStr(myWds, ",") - 1)
  ' If it starts with "!", delete it and delete
  ' the previous character in the text
  If Asc(newWd) = 33 Then
    newWd = Right(newWd, Len(newWd) - 1)
    Selection.Start = Selection.Start - 1
```

```
  End If
  Selection.TypeText Text:=newWd
  Selection.MoveLeft Unit:=wdCharacter, Count:=1
Else
' If no word found, assume that an added 'that' is needed
  Selection.Start = Selection.End
  nextChar = Selection
  If nextChar = "," Then Selection.MoveRight Unit:=wdCharacter, Count:=1
  Selection.TypeText Text:=" that"
End If
End Sub
```

##########

Geoff Hart shows a pair of macros that switch between two user names in his book "Effective Onscreen Editing," which is where I found out about them. (See http://www.geoff-hart.com/books/eoe/onscreen-book.htm)

The macros are simple. Let's assume you want to switch between a user name called "Copyeditor" and your own name. Insert the following code into your Visual Basic Editor (You can give the macros any names you wish, of course):

```
Sub CopyeditorUserName()
    Application.UserName = "Copyeditor"
    Application.UserInitials = "CE"
End Sub

Sub MyOwnUserName()
    Application.UserName = "Shmuel Gerber"
    Application.UserInitials = "SG"
End Sub
```

##########

```
' Jacq Harvey
Do
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = " ("
    .Forward = True
    .Wrap = False
    .Format = False
    .MatchWildcards = False
    .Execute
  End With
  If Selection.Find.Found = False Then Exit Sub
  Selection.TypeParagraph


  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = "). "
    .Replacement.Text = "^t"
    .Forward = True
```

```
    .Wrap = False
    .Format = False
    .MatchWildcards = False
    .Execute Replace:=wdReplaceOne
  End With

  If Selection.Find.Found = False Then
    With Selection.Find
      .ClearFormatting
      .Replacement.ClearFormatting
      .Text = ") "
      .Replacement.Text = "^t"
      .Forward = True
      .Wrap = False
      .Format = False
      .MatchWildcards = False
      .Execute Replace:=wdReplaceOne
    End With
  End If
  Selection.Start = Selection.End
  Selection.MoveDown Unit:=wdParagraph, Count:=1
Loop Until Selection.Find.Found = False


#########################################################
Sub TrimNoteSeparator()

' Macro recorded 9/28/2010 by P Beverley

  ActiveDocument.TrackRevisions = False

  ActiveWindow.View.Type = wdNormalView
  ActiveWindow.View.SplitSpecial = wdPaneEndnotes
  ActiveWindow.View.SplitSpecial = wdPaneEndnoteSeparator
  Selection.EndKey Unit:=wdLine
  Selection.EndKey Unit:=wdStory, Extend:=wdExtend
  Selection.TypeBackspace
  ActiveWindow.View.SplitSpecial = wdPaneEndnoteContinuationSeparator
  Selection.EndKey Unit:=wdLine
  Selection.EndKey Unit:=wdStory, Extend:=wdExtend
  Selection.TypeBackspace
  ActiveWindow.ActivePane.Close
  ActiveWindow.View.Type = wdPrintView

  ActiveDocument.TrackRevisions = True
End Sub


#########################################################

autolistoff


Just remembered I tweaked mine so that the macro turns off autonumbering for you:

Application.Options.AutoFormatAsYouTypeApplyBulletedLists = False
Application.Options.AutoFormatAsYouTypeApplyNumberedLists = False
```

```
ActiveDocument.ConvertNumbersToText
```

If you wanted, you could have another macro on your QAT to turn autonumbering back on afterwards:

```
Sub AutonumbersOn()
'
' Turns autonumbering back on after running the
' ConvertAutNumbersToText macro

Application.Options.AutoFormatAsYouTypeApplyBulletedLists = True
Application.Options.AutoFormatAsYouTypeApplyNumberedLists = True

End Sub
```

##########

To find any Greek character from your normal (i.e. non-Symbol) font, do a wildcard search for:
[<Greek Capital Letter Alpha>-<Greek Small Letter Omega>]
Where <Greek Capital Letter Alpha> is obtained by holding down the Alt key and typing 913 on the numeric keypad (with Num Lock on) and <Greek Small Letter Omega> is obtained by Alt-969. (I used the hexadecimal-to-decimal converter at http://www.statman.info/conversions/hexadecimal.html to find the decimal equivalent of the hexademical number that the Insert Symbol dialog box displays.)

The website says: "Since all characters from decorative fonts (Symbol-, Wingdings-fonts ...) are kept in a special code page from &HF000 to &HF0FF, you can search for them with [Alt61472-Alt61695]."

```
Sub Greekfind()
   Selection.Find.ClearFormatting
   With Selection.Find
      .Text = "[" & ChrW(894) & "-" & ChrW(974) & "]"
      .Replacement.Text = ""
      .Forward = True
      .Wrap = wdFindContinue
      .Format = False
      .MatchCase = False
      .MatchWholeWord = False
      .MatchAllWordForms = False
      .MatchSoundsLike = False
      .MatchWildcards = True
   End With
   Selection.Find.Execute
End Sub


Sub Symbolfind()
   Selection.Find.ClearFormatting
   Selection.Find.Replacement.ClearFormatting
   With Selection.Find
      .Text = "[" & ChrW(61472) & "-" & ChrW(61695) & "]"
      .Replacement.Text = ""
      .Forward = True
      .Wrap = wdFindContinue
      .Format = False
      .MatchCase = False
      .MatchWholeWord = False
      .MatchAllWordForms = False
      .MatchSoundsLike = False
```

```
        .MatchWildcards = True
    End With
    Selection.Find.Execute
End Sub
```

Date: Wed, 11 Aug 2010 13:40:38 +0100
From: Paul Beverley <paul@archivepub.co.uk>
To: SfEPLine@yahoogroups.com
Subject: Re: [SfEPLine] Searching for Symbol font and normal Greek characters

> To find any Greek character from your normal (i.e. non-Symbol) font, do a wildcard search for:
>
> [<Greek Capital Letter Alpha>-<Greek Small Letter Omega>]
>

Thanks for these, Andrew. Most helpful.

If FRedit users want to do the same sort of thing, you can use, say,

~[<Greek Capital Letter Alpha>-<Greek Small Letter Omega>]|&

(where - as before - <Greek Capital Letter Alpha> is obtained etc, etc)

and highlight the line in a particular colour, and then all those Greek
chars will be highlighted, so you can see which they are.

> (I used the hexadecimal-to-decimal converter at http://www.statman.
> info/conversions/hexadecimal.html to find the decimal equivalent of
> the hexademical number that the Insert Symbol dialog box displays.)
>
> The website says: "Since all characters from decorative fonts
> (Symbol-, Wingdings-fonts ...) are kept in a special code page from
> &HF000 to &HF0FF, you can search for them with [Alt61472-Alt61695]."

As it stands, FRedit can only look for these Symbols & Wingdings one
at a time, so you can't use this clever trick (not yet, anyway :-).

So the FRedit script line:

<&HF067>|^&

if highlighted, would highlight all the Symbol deltas.

(Note, the <> brackets are actually what you type into the FRedit
script, literally.)

>
> In my testing so far, this seems to work!
>
> I have recorded simple macros with shortcut keys for both of these searches, and these are given below.
>
> Andrew

>
>
> Sub Greekfind()
>   Selection.Find.ClearFormatting
>   With Selection.Find
>   .Text = "[" & ChrW(894) & "-" & ChrW(974) & "]"

I see you've extended the range here. Above you say 913 - 969, so
what have you added?

All the best,

Paul

Paul Beverley (Archive Publications)

Editing & proofreading services 01603-722544 - http://www.archivepub.co.uk

'Macros for editors' available free: http://www.archivepub.co.uk/TheBook

##
To: SfEPLine@yahoogroups.com
Date: Fri, 25 Jun 2010 10:00:09 +0100
Subject: Re: [SfEPLine] Re: navigating bookmarked references in
 =?iso-8859-1?Q?Word=A0?= 2007

Shift-F5 can be useful but I find it a bit limited so I have created some macros and keyboard shortcuts to set (or move if it already exists) and to go to two bookmarks: "Edited_to-here" and "Second_Edited_to_here". I might use the former for the actual editing point and the latter for the references at the end of the document.

You can of course set whatever keyboard shortcuts you want, but my scheme is Ctrl-F11, F11, Ctrl-F10 and F10, respectively, for the four self-explanatory macros below.

Andrew

```
Sub edited_to_here_bookmark()
    With ActiveDocument.Bookmarks
        .Add Range:=Selection.Range, Name:="Edited_to_here"
        .DefaultSorting = wdSortByName
        .ShowHidden = False
    End With
        Selection.GoTo What:=wdGoToBookmark, Name:="Edited_to_here"
    Selection.Find.ClearFormatting
    With Selection.Find
        .Text = ""
        .Replacement.Text = ""
        .Forward = True
        .Wrap = wdFindContinue
        .Format = False
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
```

```
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With
End Sub

Sub go_to_edited_to_here_bookmark()
    Selection.GoTo What:=wdGoToBookmark, Name:="Edited_to_here"
    Selection.Find.ClearFormatting
    With Selection.Find
        .Text = ""
        .Replacement.Text = ""
        .Forward = True
        .Wrap = wdFindContinue
        .Format = False
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With
End Sub

Sub second_edited_to_here_bookmark()
    With ActiveDocument.Bookmarks
        .Add Range:=Selection.Range, Name:="Second_Edited_to_here"
        .DefaultSorting = wdSortByName
        .ShowHidden = False
    End With
        Selection.GoTo What:=wdGoToBookmark, Name:="Second_Edited_to_here"
    Selection.Find.ClearFormatting
    With Selection.Find
        .Text = ""
        .Replacement.Text = ""
        .Forward = True
        .Wrap = wdFindContinue
        .Format = False
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With
End Sub

Sub go_to_second_edited_to_here_bookmark()
    Selection.GoTo What:=wdGoToBookmark, Name:="Second_Edited_to_here"
    Selection.Find.ClearFormatting
    With Selection.Find
        .Text = ""
        .Replacement.Text = ""
        .Forward = True
        .Wrap = wdFindContinue
        .Format = False
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
```

```
      .MatchAllWordForms = False
   End With
End Sub
```

At 09:45 on 25/06/2010 (Friday), Hazel Reid wrote:
>Andrew said < what about Shift-F5, which moves the cursor back to the
>previous editing point(s) in the document? When you reopen a file, it can
>also move the cursor to the place where it was when the file was last
>saved.>
>
>Andrew, that is brilliant. I was just at the point of opening the book I'm
>working on and it went straight to the point where I stopped last night!
>Sheer magic. Thanks.
>
>Hazel

##

```
Sub FootnoteConverter()
' Version 29.05.10
' Remember current highlight colour ...
OldColour = Options.DefaultHighlightColorIndex
' ... and select preferred colour
Options.DefaultHighlightColorIndex = wdTurquoise

' Add a text number next to each footnote number
numFoots = ActiveDocument.Footnotes.Count
For i = 1 To numFoots
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = "^f"
    .Replacement.Text = "^&" & Replace(Str(i), " ", "")
```

```
    .Replacement.Highlight = True
    .Wrap = wdFindContinue
    .MatchWildcards = False
    .Execute Replace:=wdReplaceOne
  End With
  Selection.MoveRight Unit:=wdWord, Count:=1
Next

' Find the place where you want to put the notes
With Selection.Find
  .Text = "References"
  .Style = ActiveDocument.Styles("Heading 2")
  .Forward = True
  .MatchWildcards = False
  .Execute
End With
Selection.MoveLeft Unit:=wdCharacter, Count:=2

' Prepare to put notes at the end of the text
' Selection.EndKey Unit:=wdStory
Selection.TypeParagraph

' Create "Notes" heading
Selection.Style = ActiveDocument.Styles("Heading 2")
Selection.TypeText Text:="Notes" & vbCrLf
Selection.ClearFormatting

' Copy each of the footnotes, give it a number
' and paste it at the end of the main text
For i = 1 To numFoots
  Set rng = ActiveDocument.Footnotes(i).Range
  rng.Copy
  Selection.TypeText Text:=Replace(Str(i), " ", "") & " "
  Selection.Paste
  Selection.TypeParagraph
Next

' Delete all the footnotes
For Each fn In ActiveDocument.Footnotes
  fn.Delete
Next

Options.DefaultHighlightColorIndex = OldColour

End Sub

#######################################################
Sub NotesReembed()
' Version 24.08.10
Selection.HomeKey Unit:=wdLine
firstChar = Selection
If firstChar <> "1" Then
  myResponse = MsgBox("Is this the first line of the notes?", _
   vbQuestion + vbYesNo)
  If myResponse = vbNo Then Exit Sub
End If
```

```
'Put a marker at the beginning of the footnotes
Selection.InsertAfter Text:="zxczxc" & vbCrLf

'If line starts with a tab, strip them
If Asc(firstChar) = 99 Then
  Selection.HomeKey Unit:=wdLine
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = "^p^t"
    .Replacement.Text = "^p"
    .Wrap = False
    .Forward = True
    .Execute Replace:=wdReplaceAll
  End With
End If
Do
  Selection.HomeKey Unit:=wdStory
  ' Find the footnote marker
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = "zxczxc"
    .Replacement.Text = ""
    .Wrap = wdFindContinue
    .Forward = True
    .Execute
  End With
  myStart = Selection.End + 1
  Selection.Start = myStart

  ' Get the note number
  Selection.End = myStart + 5
  myNote = Val(Selection)

  'Give up if you've reached the end
  If myNote = 0 Then Exit Do

  ' Select the footnote
  Selection.End = myStart
  Selection.MoveDown Unit:=wdParagraph, Count:=1, Extend:=wdExtend
  myPara = Selection

  ' Find the first space, i.e. after the note number
  spacePlace = InStr(myPara, " ")
  Selection.MoveEnd , -1
  Selection.MoveStart , spacePlace
  Selection.Copy
  Selection.Start = myStart
  ' Delete the used footnote
  Selection.MoveEnd , 1
  Selection.Delete

  ' Find the note citation (superscript number)
  Selection.HomeKey Unit:=wdStory
  With Selection.Find
    .ClearFormatting
```

```
    .Replacement.ClearFormatting
    .Text = Trim(Str(myNote))
    .Font.Superscript = True
    .Replacement.Font.Superscript = False
    .Replacement.Text = ""
    .Wrap = wdFindContinue
    .Forward = True
    .Execute
  End With
  ' Delete the superscript number and add a footnote
  Selection.Delete
  With ActiveDocument.Range(Start:=ActiveDocument.Content.Start, End:= _
    ActiveDocument.Content.End)
    With .FootnoteOptions
      .Location = wdBottomOfPage
      .NumberingRule = wdRestartContinuous
      .StartingNumber = 1
      .NumberStyle = wdNoteNumberStyleArabic
    End With
    .Footnotes.Add Range:=Selection.Range, Reference:=""
  End With
  ' Paste in the text of the footnote
  Selection.Paste
Loop Until myNote = 0

' Tidy up and go home
Selection.HomeKey Unit:=wdStory
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "zxczxc"
  .Replacement.Text = ""
  .Wrap = wdFindContinue
  .Forward = True
  .Execute Replace:=wdReplaceOne
End With
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = ""
End With
Selection.EndKey Unit:=wdStory

End Sub

######################################################
Sub FReditInstructions()
Documents.Open filename:="C:\Program Files\VirtualAcorn\VirtualRPC-
SA\HardDisc4\MyFiles2\WIP\zzzTheBook\Macros4Editors.doc"
Selection.HomeKey Unit:=wdStory
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "1. How the Macro Works"
  .Replacement.Text = ""
  .MatchWildcards = False
  .Execute
```

```
End With
startHere = Selection.Start
Selection.End = Selection.Start
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "5. List of Short Script"
  .Replacement.Text = ""
  .MatchWildcards = False
  .Execute
End With
Selection.End = Selection.Start
Selection.Start = startHere
Selection.Copy
Documents.Open filename:="C:\Program Files\VirtualAcorn\VirtualRPC-
SA\HardDisc4\MyFiles2\WIP\zzzFRedit\1_Instructions.doc"
Selection.HomeKey Unit:=wdStory
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "1. How the Macro Works"
  .Replacement.Text = ""
  .MatchWildcards = False
  .Execute
End With
startHere = Selection.Start
Selection.End = Selection.Start
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "Well, that's it."
  .Replacement.Text = ""
  .MatchWildcards = False
  .Execute
End With
Selection.End = Selection.Start
Selection.Start = startHere
Selection.Delete
Selection.Paste
End Sub

Sub FReditMacro()
Documents.Open filename:="C:\Program Files\VirtualAcorn\VirtualRPC-
SA\HardDisc4\MyFiles2\WIP\zzzTheBook\Macros4Editors.doc"
Selection.HomeKey Unit:=wdStory
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "Sub FRedit()^p"'
  .Replacement.Text = ""
  .MatchWildcards = False
  .Execute
End With
startHere = Selection.Start
Selection.End = Selection.Start
With Selection.Find
  .ClearFormatting
```

```
  .Replacement.ClearFormatting
  .Text = "Textalyse Macro"
  .Replacement.Text = ""
  .MatchWildcards = False
  .Execute
End With
Selection.End = Selection.Start
Selection.Start = startHere
Selection.Font.Reset
Selection.TypeBackspace
Selection.MoveUp Unit:=wdLine, Count:=1
Selection.TypeParagraph
Selection.Paste

Documents.Open filename:="C:\Program Files\VirtualAcorn\VirtualRPC-
SA\HardDisc4\MyFiles2\WIP\zzzTheBook\zzFRedit\2_Macro.doc"
Selection.WholeStory
Selection.Delete
Selection.Paste

Selection.HomeKey Unit:=wdStory
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "Sub FRedit()^p"
  .Replacement.Text = ""
  .MatchWildcards = False
  .Execute Replace:=wdReplaceOne
End With

With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "End Sub"
  .Replacement.Text = ""
  .MatchWildcards = False
  .Execute Replace:=wdReplaceOne
End With

Selection.HomeKey Unit:=wdStory

End Sub

######################################################
Sub RevisionCount()
TrackState = ActiveDocument.TrackRevisions
ActiveDocument.TrackRevisions = False

Dim numDelete As Long, numInsert As Long, _
    numFormat As Long, numOther As Long, _
    numChanges As Long

numChanges = ActiveDocument.Revisions.Count
i = numChanges
For Each rev In ActiveDocument.Revisions
'  ActiveDocument.Revisions(i).Range.Select
  i = i - 1
```

```
  JLJ = rev.Range
  vChange = rev.Type
  Select Case vChange
    Case wdRevisionDelete: numDelete = numDelete + 1
    Case wdRevisionInsert: numInsert = numInsert + 1
    Case wdRevisionParagraphProperty: numFormat = numFormat + 1
    Case wdRevisionSectionProperty: numFormat = numFormat + 1
    Case wdRevisionStyleDefinition: numFormat = numFormat + 1
    Case wdRevisionTableProperty: numFormat = numFormat + 1
    Case wdRevisionProperty: numRevProp = numRevProp + 1
  Case Else
    numOther = numOther + 1
' Not implemented:
xcvxc = wdNoRevision: xcvxc = wdRevisionReconcile
xcvxc = wdRevisionConflict: xcvxc = wdRevisionDisplayField
xcvxc = wdRevisionParagraphNumber
xcvxc = wdRevisionReplace: xcvxc = wdRevisionStyle
  End Select
  If i Mod 10 = 0 Then StatusBar = Str(i)
Next rev

Selection.EndKey Unit:=wdStory
' Have to do it again. Weird!
Selection.EndKey Unit:=wdStory
' If you are at the site of a revision,
' it doesn't work first time!


Selection.TypeText Text:=vbCrLf & "Total:  " & Str(numChanges) & vbCrLf
Selection.TypeText Text:="Deletes: " & Str(numDelete) & vbCrLf
Selection.TypeText Text:="Inserts:" & Str(numInsert) & vbCrLf
Selection.TypeText Text:="Formats:" & Str(numFormat) & vbCrLf
Selection.TypeText Text:="Revision property:" & Str(numRevProp) & vbCrLf
Selection.TypeText Text:="Other:" & Str(numOther) & vbCrLf

numMissing = numChanges - numOther - numDelete - numInsert _
    - numFormat - numRevProp

StatusBar = ""

ActiveDocument.TrackRevisions = TrackState
Exit Sub

########################################################

Sub ItalicCount()
' Version 22.07.2010

Selection.EndKey Unit:=wdStory
theEnd = Selection.Start

' copy all the footnotes to the end of the text
If ActiveDocument.Footnotes.Count > 0 Then
  For Each fn In ActiveDocument.Footnotes
    fn.Range.Copy
    Selection.Paste
  Next
```

```
End If

' copy all the endnotes to the end of the text
If ActiveDocument.Endnotes.Count > 0 Then
  For Each fn In ActiveDocument.Endnotes
    fn.Range.Copy
    Selection.Paste
  Next
End If

' copy all the textboxes to the end of the text
Set rng = ActiveDocument.Range
rng.Start = rng.End
If ActiveDocument.Shapes.Count > 0 Then
  For Each shp In ActiveDocument.Shapes
    If shp.TextFrame.HasText Then
      Set rng2 = shp.TextFrame.TextRange
      rng2.Copy
      rng.Paste
      rng.Start = rng.End
    End If
  Next
End If

Selection.HomeKey Unit:=wdStory
totItalic = 0
totRoman = 0
totChars = ActiveDocument.Characters.Count
For Each myChar In ActiveDocument.Characters
  If myChar.Font.Italic = True Then
    totItalic = totItalic + 1
  Else
    totRoman = totRoman + 1
  End If
If totItalic Mod 100 = 0 Then StatusBar = _
     "    Press <Ctrl-Break> to stop.      " _
    & "Remaining:  " & Int((totChars - totItalic - totRoman) / 100)
Next

Selection.EndKey Unit:=wdStory
Selection.Start = theEnd
Selection.Delete
Selection.HomeKey Unit:=wdStory

StatusBar = ""
MsgBox ("Italic: " & totItalic & vbCrLf & vbCrLf _
    & "Roman: " & totRoman)

End Sub
```

##########

From: http://wordribbon.tips.net/Pages/T008922_Examining_Tracked_Changes_in_a_Macro.html

Examining Tracked Changes in a Macro (2007/10)

Summary: The Track Changes feature in Word is very handy when you need to see what edits are made to a document. Using a macro you can even access the changes to see what they are. Here's how to get at the most elemental of the change information. (This tip works with MS Word 2007, and Word 2010. You can find a version of this tip for the older menu interface of Word here: Examining Tracked Changes in a Macro.)

If you need to develop a macro to process a document in which changes have been tracked (using the Track Changes feature), you may wonder how you can determine the number of changes in the document and how you can look at each of the changes, programmatically. It isn't that tough to do if you remember that the changes are organized in using the Revisions collection. This means that you can determine the number of changes using this code:

```
iNumChanges = ActiveDocument.Revisions.Count
```

Just like any other collection, you can step through each member of the Revisions collection and figure out various information about the change represented in the member. While the details of what properties and methods belong to the Revisions collection is much too voluminous for this tip, you can determine the type of each change by looking at the Type property, in this manner:

```
vChange = ActiveDocument.Revisions(1).Type
```

At this point vChange will be equal to one of 14 possible revision types. These revision types can be referenced by the built-in constants wdNoRevision, wdRevisionDelete, wdRevisionInsert, wdRevisionParagraphProperty, wdRevisionReconcile, wdRevisionSectionProperty, wdRevisionStyleDefinition, wdRevisionConflict, wdRevisionDisplayField, wdRevisionParagraphNumber, wdRevisionProperty, wdRevisionReplace, wdRevisionStyle, and wdRevisionTableProperty.

Additional information can be found in Word's Help system or by searching the Internet for the phrase "revisions collection".

##########
Examining Tracked Changes in a Macro (2002/3)

Summary: The Track Changes feature in Word is very handy when you need to see what edits are made to a document. Using a macro you can even access the changes to see what they are. Here's how to get at the most elemental of the change information. (This tip works with Microsoft Word 97, Word 2000, Word 2002, and Word 2003. You can find a version of this tip for the ribbon interface of Word (Word 2007 and later) here: Examining Tracked Changes in a Macro.)

If you need to develop a macro to process a document in which

changes have been tracked (using the Track Changes feature), you may wonder how you can determine the number of changes in the document and how you can look at each of the changes, programmatically. It isn't that tough to do if you remember that the changes are organized in using the Revisions collection. This means that you can determine the number of changes using this code:

```
iNumChanges = ActiveDocument.Revisions.Count
```

Just like any other collection, you can step through each member of the Revisions collection and figure out various information about the change represented in the member. While the details of what properties and methods belong to the Revisions collection is much too voluminous for this tip, you can determine the type of each change by looking at the Type property, in this manner:

```
vChange = ActiveDocument.Revisions(1).Type
```

At this point vChange will be equal to one of 14 possible revision types. These revision types can be referenced by the built-in constants wdNoRevision, wdRevisionDelete, wdRevisionInsert, wdRevisionParagraphProperty, wdRevisionReconcile, wdRevisionSectionProperty, wdRevisionStyleDefinition, wdRevisionConflict, wdRevisionDisplayField, wdRevisionParagraphNumber, wdRevisionProperty, wdRevisionReplace, wdRevisionStyle, and wdRevisionTableProperty.

Additional information can be found in Word's Help system or by searching the Internet for the phrase "revisions collection".

```
#######################################################
Sub JumpUp2()
'
' JumpUp2 Macro
' Macro recorded 7/12/2010 by P Beverley
'
  With ActiveDocument.Bookmarks
    .Add Range:=Selection.Range, Name:="ComeBackHere2"
    .DefaultSorting = wdSortByName
    .ShowHidden = True
  End With
  Selection.HomeKey Unit:=wdStory
  Selection.GoTo What:=wdGoToBookmark, Name:="ComeBackHere2"
  Selection.Find.ClearFormatting
  With Selection.Find
    .Text = "Needham"
    .Replacement.Text = "emporia"
    .Forward = True
    .Wrap = wdFindContinue
    .Format = False
    .MatchCase = False
    .MatchWholeWord = False
```

```
    .MatchWildcards = False
    .MatchSoundsLike = False
    .MatchAllWordForms = False
  End With
End Sub

#######################################################
' Sub MultiFileComment()
myResponse = MsgBox("Comment collector: Word" & vbCrLf & _
     "Have you opened the first file?", vbQuestion + vbYesNo)
If myResponse = vbNo Then Exit Sub

myFolder = ActiveDocument.Path
ActiveDocument.Close SaveChanges:=False

Set myFileSystem = CreateObject("Scripting.FileSystemObject")
Set myFileList = myFileSystem.GetFolder(myFolder).Files

If Word.Documents.Count = 0 Then Documents.Add
Set myList = ActiveDocument
blankLine = vbCrLf & vbCrLf
FilesTotal = 0
For Each myFile In myFileList
  myFileType = Right(myFile, 4)
  If myFileType = ".doc" Or myFileType = "docx" Or myFileType = ".rtf" Then
    Set myDoc = Application.Documents.Open(filename:=myFile.Path, ReadOnly:=True)

    If ActiveDocument.Comments.Count >= 1 Then
      ActiveDocument.StoryRanges(wdCommentsStory).Copy
      myList.Activate
      myDocName = myDoc.Name
      Selection.TypeText Text:=myDocName & vbCrLf
      Selection.MoveUp Unit:=wdLine, Count:=1, Extend:=wdExtend
      Selection.Font.Bold = True
      Selection.EndKey Unit:=wdStory
      Selection.Paste
      Selection.TypeText Text:=vbCrLf & vbCrLf
    End If

    myDoc.Close SaveChanges:=wdDoNotSaveChanges
    FilesTotal = FilesTotal + 1
  End If
Next
MsgBox ("Files checked" + Str(FilesTotal))


' End Sub

#######################################################

Soundlike for spotting links between words - didn't work.

Dim myCol(20)
OldColour = Options.DefaultHighlightColorIndex
myCol(1) = wdYellow
myCol(2) = wdTurquoise
myCol(3) = wdBrightGreen
```

```
myCol(4) = wdPink
myCol(5) = wdRed
myCol(6) = wdBlue
myCol(7) = wdLightGray
myCol(8) = wdDarkGray
myColTotal = 8
nowCol = 1

For Each p In ActiveDocument.Range.Paragraphs
  Options.DefaultHighlightColorIndex = myCol(nowCol)
  nowCol = (nowCol Mod myColTotal) + 1
  If Len(p) > 5 And InStr(p, "-") = 0 Then
    p.Range.Select
    thisWordStart = Selection.Start
    tabPos = InStr(p, Chr(9))
    myWord = Left(p, tabPos - 1)
    Set rng = ActiveDocument.Range
    rng.Start = Selection.End
    With rng.Find
      .ClearFormatting
      .Replacement.ClearFormatting
      .Highlight = False
      .MatchWildcards = False
      .Text = myWord
      .Wrap = False
      .Replacement.Text = "^&"
      .MatchWholeWord = True
      .MatchSoundsLike = True
      .Replacement.Highlight = True
      .Execute Replace:=wdReplaceOne
    End With
    fgkd = Selection.End
    fgd = rng.End
    If rng.End = ActiveDocument.Range.End Then
      Selection.Range.HighlightColorIndex = wdColorAutomatic
    Else
      rng.Start = thisWordStart
      With rng.Find
        .Execute Replace:=wdReplaceOne
'       .Wrap = wdFindContinue
      End With
    End If
  End If
Next

Options.DefaultHighlightColorIndex = OldColour


#########################################################
checkLength = 100
For Each p In ActiveDocument.Range.Paragraphs
  paraText = p
  If Left(paraText, 5) = "Table" Then
    Set rng = p.Range
    startTitle = rng.Start
    rng.Start = rng.End
    startTable = rng.End
```

```
    With rng.Find
      .ClearFormatting
      .Replacement.ClearFormatting
      .MatchWildcards = False
      .Text = "^p"
      .Replacement.Text = ""
      .Execute
    End With
    rng.Start = startTable
    tableText = rng
    If InStr(tableText, Chr(7)) Then
      ' This is a table
      titleText = paraText
      ' then chop "Table x.y" out
      rng.Start = startTitle
      ' now chop title = table out
      rng.Cut
      ' now type in "[<table x.y here]
      rng.Select
      Selection.typeafter
      ' rng.Select
    End If
  End If
Next
```

##########

Date: Tue, 29 Jun 2010 09:21:19 -0400
Subject: Re: [CE-L] Tools Query: Macro to count the number of changes in Track Changes
To: COPYEDITING-L@LISTSERV.INDIANA.EDU

The Revisions collection has a count property, making it easy to count the TOTAL number of revisions using a single command:

```
Sub BigDumbErrorCounter()
MsgBox ActiveDocument.Revisions.Count
End Sub
```

This macro looks only at the body of the document. If you want to include revisions in headers, footers, textboxes and so on, you have to do more work with document ranges. And keep in mind that changing something (deleting and overwriting with something else) counts as two changes - an insertion and a deletion.

If you want to categorize the revisions, you have to examine each revision and categorize them and count in the different categories. Or you could do fancy tricks with showing different types of revisions and reviewers, accepting what's visible, redoing the count, comparing the numbers, then undoing the accept and the hide.

- Jessica
* Obtain permission before forwarding. Pix: http://bit.ly/7fUx9O
* Stuff: http://bit.ly/5PONhz FAQ: http://bit.ly/4ocsZ2
* Sub: http://bit.ly/8dsD41 Community: http://bit.ly/80YUUd

##########

Date: Mon, 28 Jun 2010 21:09:28 -0700
Subject: Re: [CE-L] TOOLS: Changing cases
To: COPYEDITING-L@LISTSERV.INDIANA.EDU

Julie Vaughn wrote:
> Using Word for the Mac on my MacBook Pro, Shift-F3 toggles among the
> three case options (uppercase, lowercase, and sentence or title case,
> the latter depending on the text selected).

If you want more refined capping without toggling individual words,
here's one of Jack Lyon's useful freebies:

```
Sub NewTrueTitleCase()
'Created by Jack M. Lyon
'updated by Hilary Powers to ignore head levels
'http://www.editorium.com
    Selection.Range.Case = wdTitleWord
    For Each wrd In Selection.Range.Words
     Select Case Trim(wrd)
      Case "A", "An", "As", "At", "And", "But", _
      "By", "For", "From", "In", "Into", "Of", _
      "On", "Onto", "Or", "The", _
      "To", "Unto", "With"
      wrd.Case = wdLowerCase
     End Select
    Next wrd
    wrdCount = Selection.Range.Words.Count
    Selection.Range.Words(1).Case = wdTitleWord
    Selection.Range.Words(wrdCount - 1).Case = wdTitleWord
    strLength = Selection.Range.Characters.Count
    For i = 1 To strLength
     If Selection.Range.Characters(i) = ":" Then
      Selection.Range.Characters(i + 2).Case = wdTitleWord
     End If
    Next i
End Sub
```

It takes a selection and applies true title case, lowercasing
prepositions and articles except at the beginning and end, and following
a colon.

##

```
' Changes the initial letter after a colon to lowercase
oldFind = Selection.Find.Text
oldReplace = Selection.Find.Replacement.Text
rng = ActiveDocument.Range
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .MatchWildcards = True
  .Text = "<to>[a-zA-Z ]@ly>"
  .Replacement.Text = ""
  .Execute
End With

While Selection.Find.Found
  If Selection.End - Selection.Start < 20 Then
```

```
    Selection.Range.HighlightColorIndex = wdPink
  End If
  Selection.Find.Execute
Wend

With Selection.Find
  .Text = oldFind
  .Replacement.Text = oldReplace
  .MatchWildcards = False
End With
```

##

There is a property Document.ShowRevisions that can be toggled programmatically. Setting this property to False is the same as setting the menu item for revision display to "Final"--in other words, deletions are not displayed and insertions are displayed as normal (unmarked) text. But there are also other properties that come into play when the .ShowRevisions property is set to True thast detremine how the revisions are actually displayed. it would actually require a small decision tree to completely describe the behavior when .ShowRevisions is True.

##

```
' Go to the middle of a selection.

midPoint = Int((Selection.Start + Selection.End) / 2)
Selection.Start = midPoint
Selection.End = midPoint
```

##

While something is found, do something...

```
>     Selection.Find.ClearFormatting
>     With Selection.Find
>         .Text = "page TK"
>         .Forward = True
>         .Wrap = wdFindContinue
>         .Format = False
>         .MatchCase = True
>         .MatchWholeWord = False
>         .MatchWildcards = False
>         .MatchSoundsLike = False
>         .MatchAllWordForms = False
>     End With

While Selection.Find.Found
    Selection.MoveEnd Unit:=wdCharacter, Count:=-2
    Selection.InsertAfter Text:="*"
    Selection.Find.Execute
```

```
Wend


#########################################################
Find if a file is is a directory

If Dir(strWindowsTempDirPath, vbDirectory) <> "" Then

#########################################################

Sub FindHiLi99()
' No idea what this is!
varExists = False
For Each v In ActiveDocument.Variables
  If v.Name = "hColour" Then varExists = True: Exit For
Next v

If varExists = False Then ActiveDocument.Variables.Add "hColour", 0
searchColour = ActiveDocument.Variables("hColour")

' If no text is selected, search for next highlight
If Selection.Start = Selection.End Then GoTo FindNext

' If some text is selected, see what colour it is;
' then go find more text of that colour.
selColour = Selection.Range.HighlightColorIndex
If selColour > 0 Then
  searchColour = selColour
  ActiveDocument.Variables("hColour") = searchColour
  GoTo FindNext
End If

Selection.Start = Selection.End
Set rng = ActiveDocument.Text
Do
  With rng.Find
   .ClearFormatting
   .Replacement.ClearFormatting
   .Text = ""
   .Highlight = True
   .Wrap = False
   .Replacement.Text = ""
   .Forward = True
   .MatchWildcards = False
   .Execute
  End With

  rng.Start = rng.End
  myResponse = MsgBox("This colour? (Cancel = any colour)", vbQuestion + vbYesNoCancel)
  If myResponse = vbCancel Then
    ActiveDocument.Variables("hColour") = 0
    Exit Sub
  End If
  If rng.Find.Found = False Then rng.Select: Exit Sub
Loop Until myResponse = vbYes
searchColour = rng.Range.HighlightColorIndex
ActiveDocument.Variables("hColour") = searchColour
```

```
If myResponse = vbYes Then GoTo finish

FindNext:
Set rng = ActiveDocument.Content
rng.Start = Selection.Start
If searchColour = 0 Then
  With rng.Find
   .ClearFormatting
   .Replacement.ClearFormatting
   .Text = ""
   .Highlight = True
   .Wrap = False
   .Replacement.Text = ""
   .Forward = True
   .MatchWildcards = False
   .Execute
  End With
Else
  foundHlight = False
  Do
    With rng.Find
     .ClearFormatting
     .Replacement.ClearFormatting
     .Text = ""
     .Highlight = True
     .Wrap = False
     .Replacement.Text = ""
     .Forward = True
     .MatchWildcards = False
     .Execute
    End With
    thisColour = rng.HighlightColorIndex
    If thisColour = searchColour Or thisColour > 100 Then
       foundHlight = True
    End If
  Loop Until foundHlight = True Or rng.Find.Found = False
  If thisColour > 100 Then
    rng.Start = rng.Start - 1
    rng.End = rng.Start + 1
    Do
      hereNow = rng.Start
      rng.Start = rng.Start + 1
      rng.End = rng.End + 1
      thisColour = rng.Range.HighlightColorIndex
    Loop Until thisColour = searchColour Or rng.Start = hereNow
    Do
      hereNow = rng.Start
      rng.End = rng.End + 1
      thisColour = rng.Range.HighlightColorIndex
    Loop Until thisColour <> searchColour Or rng.Start = hereNow
      rng.End = rng.End - 1
  End If
End If

finish:
rng.Start = rng.End
```

```
'With Selection.Find
' .ClearFormatting
' .Replacement.ClearFormatting
' .Text = ""
' .Replacement.Text = ""
'End With


End Sub

########################################################################


' Extend from cursor to the selected text
' Sounds useful!

Selection.Extend
Selection.Extend Character:=")"


########################################################################
Date: Fri, 02 Apr 2010 14:21:25 +0100
To: Paul Beverley <paul@archivepub.co.uk>
Subject: Re: [SfEPLine] Word macro needed

  Paul

Your VB code uses "New DataObject", which needs the MS Forms 2.0 Object
Library to be loaded - not always the case, and wasn't so on my PC
(Windows 7 and MS Office 2003). You can load this manually from the VB
editor (Tools > References), but I added the following routine to load
it via code:

//On Error Resume Next
ActiveWorkbook.VBProject.References.AddFromGuid _
GUID:="{0D452EE1-E08F-101A-852E-02608C4D0BB4}", _
major:=2, minor:=0

Just thought I'd mention it as I got a run error when I tried your macro
for the first time.

Regards

*Rich Cutler*
Project Manager

----------------------------------------------------------------------
/*Helius* /
Unit 8, 15 Lincoln Cottages, Brighton BN2 9UJ, UK
T: +44 (0)1273 570536
F: +44 (0)8700 940882
www.helius.biz <http://www.helius.biz>
----------------------------------------------------------------------
########################################################################


Sub EndnotesToText()
```

```
Dim aendnote As Endnote

For Each aendnote In ActiveDocument.Endnotes
ActiveDocument.Range.InsertAfter vbCr & aendnote.Index & vbTab & aendnote.Range
aendnote.Reference.InsertBefore "a" & aendnote.Index & "a"
Next aendnote

For Each aendnote In ActiveDocument.Endnotes
aendnote.Reference.Delete
Next aendnote

Selection.Find.ClearFormatting
Selection.Find.Replacement.ClearFormatting

With Selection.Find.Replacement.Font
.Superscript = True
End With
With Selection.Find
.Text = "(a)([0-9]{1,})(a)"
.Replacement.Text = "\2"
.Forward = True
.Wrap = wdFindContinue
.Format = True
.MatchWildcards = True
End With
Selection.Find.Execute Replace:=wdReplaceAll

End Sub
```

Date: Fri, 02 Apr 2010 13:12:46 +0100
To: Paul Beverley <paul@archivepub.co.uk>
Subject: Re: [SfEPLine] Word macro needed

Hi Paul:

> When you say "look into the problem" it sounds as if I may not have
> made it clear that that version I sent you does do what you asked for;
> it does detect those funny characters on page 10 of your sample text.

Sorry for my brief, and poorly worded, reply yesterday - had to rush out. It does indeed do what you say, and I'm very grateful for you taking the time to help me. I've got a macro that find Symbol font characters, and I'll append your macro to it, as those strange characters behave more like Symbol than Unicode!

BTW, here are some snippets of code that you may find useful - some you my already have.

-=Rich=-

```
'Pastes text from the clipboard without its formatting.
'
'I assign the shortcut key [Shift][insert] to this

Public Sub MAIN()
WordBasic.EditPasteSpecial IconNumber:=0, Link:=0, DisplayIcon:=0,
Class:="Word.Document.6", DataType:="Text",
IconFileName:="C:\PROGRA~1\MSOFFI~1\WINWORD\WINWORD.EXE",
Caption:="Microsoft Word Document"
End Sub

----------

Sub CutGraphics()
'
' Cut and highlight frames (incl. graphics)

    WordBasic.ScreenUpdating 0

    Options.DefaultHighlightColorIndex = wdTurquoise

    Selection.Find.ClearFormatting
    Selection.Find.Replacement.ClearFormatting
    Selection.Find.Replacement.Highlight = True
    With Selection.Find
       .Text = "^g"
       .Replacement.Text = "X"
       .Forward = True
       .Wrap = wdFindContinue
       .Format = True
       .MatchCase = False
       .MatchWholeWord = False
       .MatchWildcards = False
       .MatchSoundsLike = False
       .MatchAllWordForms = False
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

    Selection.Find.ClearFormatting
    Selection.Find.Replacement.ClearFormatting

    Selection.Find.Highlight = True
    With Selection.Find.Replacement.Font
       .Position = 0
    End With
    Selection.Find.Replacement.Highlight = True
    With Selection.Find
       .Text = "X"
       .Replacement.Text = ""
       .Forward = True
       .Wrap = wdFindContinue
       .Format = True
       .MatchCase = False
       .MatchWholeWord = False
       .MatchWildcards = False
```

```
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

    Selection.Find.ClearFormatting
    Selection.Find.Replacement.ClearFormatting

    Options.DefaultHighlightColorIndex = wdYellow

End Sub


-------------------------------------------------

Sub TableStopStrip()

' Deletes stops at end of paras in table entries

    WordBasic.ScreenUpdating 0

    Selection.Find.ClearFormatting
    Selection.Find.Replacement.ClearFormatting
    With Selection.Find
        .Text = ".^p"
        .Replacement.Text = "^p"
        .Forward = True
        .Wrap = wdFindContinue
        .Format = False
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

' Resets Find/replace dialog

    Selection.Find.ClearFormatting
    Selection.Find.Replacement.ClearFormatting

        With Selection.Find
        .Text = ""
        .Replacement.Text = ""
        .Forward = True
        .Wrap = wdFindContinue
        .Format = False
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With

' Following written by RW at Prufrock 9/9/2002
```

```
Dim myTable As Table
Dim myCell As Cell
For Each myTable In ActiveDocument.Tables
    For Each myCell In myTable.Range.Cells
        myCell.Select
        Selection.Collapse Direction:=wdCollapseEnd
        Selection.Move Unit:=wdCharacter, Count:=-2
        If Selection.Text = "." Then Selection.Delete
    Next myCell
Next myTable

End Sub

---------------------------------------------------------------------

Sub ExtractTables()
'//////////////
'This macro extracts tables to another document.
'//////////////
Dim FirstDoc, SecondDoc As Document
Dim i As Long
Dim TableNumber As String
Dim TableRg As Range

    'Save a reference to the original document
    Set FirstDoc = ActiveDocument
    i = FirstDoc.Tables.Count
    If i = 0 Then
        MsgBox "No table found in this document. Macro will stop"
        End
    End If
    i = 0

    'Create new doc where to put the tables
    Set SecondDoc = Documents.Add(Template:="Normal", NewTemplate:=False)
    ActiveWindow.View.Type = wdNormalView

    'Process tables

    For Each myTable In FirstDoc.Tables
        i = i + 1
        TableNumber = "Table__" & Format(i, "0000")

        'Set range to table text
        Set TableRg = myTable.Range

        'paste contents of table at end of new doc
        Set PasteRg = SecondDoc.Range   'start with whole doc,
        PasteRg.Collapse wdCollapseEnd  'then collapse to end
        PasteRg.InsertParagraphBefore
        PasteRg.InsertBefore _
            "**********" & TableNumber & "********"
        PasteRg.InsertParagraphBefore
        Set PasteRg = SecondDoc.Range   'start with whole doc,
        PasteRg.Collapse wdCollapseEnd  'then collapse to end
        PasteRg.FormattedText = TableRg.FormattedText
```

```
        'In the main doc, Table with reference
        Set TableRg = myTable.Range  'move range to table
        TableRg.Collapse wdCollapseEnd 'move off it to the right
        TableRg.InsertBefore _
           "**********Insert " & TableNumber & " here!********"
        TableRg.InsertParagraphAfter
        myTable.Delete
    Next myTable


End Sub
```

##
##

Clear Tracking from Format Changes
Word 2002+ will insist on tracking changes in formatting if you're tracking anything at all, which makes clearing the tracking after author review much more difficult. If you want to get all the formatting out of play at once, the following macro will do the job. (This is my newest toy, created after years of intermittent nibbling at the problem. I keep it on Alt+Ctrl+Shift+F.)

```
Sub AcceptFormatChanges()
' Macro cobbled up 8/14/06 by Hilary Powers
' Based on commands revealed by Keri Morgret
' Use only on file that opens with all markup visible!
' Next four lines toggle all display of changes off.
WordBasic.ShowFormatting
WordBasic.ShowInkAnnotations
WordBasic.ShowInsertionsAndDeletions
WordBasic.ShowComments
' Toggle display of formatting back on.
WordBasic.ShowFormatting
' Approve formatting changes.
WordBasic.AcceptAllChangesShown
' Next three lines toggle everything else back on.
WordBasic.ShowInsertionsAndDeletions
WordBasic.ShowComments
WordBasic.ShowInkAnnotations
End Sub
```

##

Date: Fri, 19 Mar 2010 10:00:58 -0400
Subject: Re: [CE-L] Uncheck Formatting info in Track Changes permanently
To: COPYEDITING-L@LISTSERV.INDIANA.EDU

Erin is looking for a way to permanently stop Word 2003 from showing formatting comments.

Interesting. The setting I would think governs this does not seem to work.

Go to Tools, Options, Security and uncheck Make hidden markup visible when opening or saving.

It makes no difference.

Of course you could use an AutoOpen macro that sets these things for you each time you open a doc.

Here's a simple one:

```
Sub AutoOpen

   ActiveDocument.ActiveWindow.View.ShowFormatChanges = False

End Sub
```

- Jessica

Or you could use this macro occasionally to accept all the formatting changes and leave other changes in place. The idea came from Hilary Powers, and I updated to use Word 2003 properties rather than WordBasic commands, and to restore your original settings rather than toggling everything back on. Caveat: I haven't tested this much, so you'd probably want to save the doc first. Or try it out on a junk copy.

```
Sub AcceptFormatChanges()
 ' robustified by Jessica Weissman, based on ideas from Hilary Power and
Keri Morgret
   Dim bShowInk As Boolean, bShowFormat As Boolean, bShowInsDel As
Boolean, bShowComments As Boolean
 ' save current settings for the show stuff

  bShowInk = ActiveDocument.ActiveWindow.View.ShowInkAnnotations
  bShowFormat = ActiveDocument.ActiveWindow.View.ShowFormatChanges
  bShowInsDel =
ActiveDocument.ActiveWindow.View.ShowInsertionsAndDeletions
  bShowComments = ActiveDocument.ActiveWindow.View.ShowComments

 ' hide other stuff and show format changes
    With ActiveDocument.ActiveWindow.View
       .ShowFormatChanges = True
       .ShowInkAnnotations = False
       .ShowComments = False
       .ShowInsertionsAndDeletions = False
    End With
' Approve formatting changes.
     ActiveDocument.AcceptAllRevisionsShown

' Restore original settings
   With ActiveDocument.ActiveWindow.View
    .ShowFormatChanges = bShowFormat
    .ShowInkAnnotations = bShowInk
    .ShowInsertionsAndDeletions = bShowInsDel
    .ShowComments = bShowComments
   End With
```

Date: Fri, 19 Mar 2010 10:38:25 -0400
Subject: Re: [CE-L] Uncheck Formatting info in Track Changes permanently
To: COPYEDITING-L@LISTSERV.INDIANA.EDU

Somehow a line was cut from the macro in the original posting. Here it is with the final line restored.

- Jessica, who did not see Hilary's response before posting mine

```
Sub AcceptFormatChanges()
 ' robustified and updated by Jessica Weissman, based on ideas from
Hilary Powers and Keri Morgret
   Dim bShowInk As Boolean, bShowFormat As Boolean, bShowInsDel As
Boolean, bShowComments As Boolean

' save current settings for the show stuff

  bShowInk = ActiveDocument.ActiveWindow.View.ShowInkAnnotations
  bShowFormat = ActiveDocument.ActiveWindow.View.ShowFormatChanges
  bShowInsDel =
ActiveDocument.ActiveWindow.View.ShowInsertionsAndDeletions
  bShowComments = ActiveDocument.ActiveWindow.View.ShowComments

 ' hide other stuff and show format changes

    With ActiveDocument.ActiveWindow.View
       .ShowFormatChanges = True
       .ShowInkAnnotations = False
       .ShowComments = False
       .ShowInsertionsAndDeletions = False
    End With

' Approve formatting changes - they're the only revisions showing now
     ActiveDocument.AcceptAllRevisionsShown

' Restore original settings
   With ActiveDocument.ActiveWindow.View
    .ShowFormatChanges = bShowFormat
    .ShowInkAnnotations = bShowInk
    .ShowInsertionsAndDeletions = bShowInsDel
    .ShowComments = bShowComments
   End With
End Sub
```



```
######################################################################
Sub zzPosPub()

funnyCode = "zzBlank"
' used as the code to mean 'leave the Find/Replace box blank
```

```
maxLines = 750
' the maximum number of lines in your list

ReDim findArray(maxLines), replaceArray(maxLines) As String
ReDim findArray(maxLines), replaceArray(maxLines) As String
ReDim hlArray(maxLines) As WdColorIndex
ReDim styleArray(maxLines, 2) As String
ReDim funct(maxLines, 16) As Boolean

Dim myFile(20) As String

Dim fText, rText, fStyle, rStyle, allLine, changeTo As String
Dim startAtListFile, Fopposite, Ropposite As Boolean
Dim fBold, rBold, fItalic, rItalic As Boolean
Dim fSuper, rSuper, fSub, rSub, fUline, rUline, fSmall, rSmall As Boolean
Dim mchWild, mchCase As Boolean, docs As Integer


' Remember the existing highlight colour
OldColour = Options.DefaultHighlightColorIndex
' On Error GoTo ReportIt

' ################################################
 ActiveWindow.View.Type = wdPrintView

 ActiveWindow.ActivePane.View.Zoom.Percentage = 150

 Selection.WholeStory
 Selection.Style = ActiveDocument.Styles("Normal")

 With ActiveDocument.Styles("Normal").ParagraphFormat
   .SpaceAfter = 9
 End With


 Selection.WholeStory
 With Selection.Font
  .Name = "Times New Roman"
  .Size = 13
  .Bold = False
  .Color = wdColorAutomatic
  .Italic = False
  .Underline = False
 End With

 Selection.WholeStory
 Selection.LanguageID = wdEnglishUK

 Application.ScreenUpdating = False


 ' Selection.HomeKey Unit:=wdStory
  theText = ActiveDocument.Name
 docs = Word.Documents.Count
 theList = ""
For i = 1 To docs
   thisFile = Documents(i).Name
```

```
  If thisFile = "PosPublist.doc" Then
    theList = "PosPublist.doc"
    i = docs
  End If
Next
If theList = "" Then
  PreEditList = "C:\Program Files\VirtualAcorn\VirtualRPC-SA\HardDisc4\MyFiles\Positive\PosPublist.doc"
  Documents.Open FileName:=PreEditList
  theList = "PosPublist.doc"
End If

GoTo theStart


' ###############################################



' Remember which is the currently active file
startFile = ActiveDocument.Name
Set rng = ActiveDocument.Content
rng.End = rng.Start + 50
With rng.Find
  .Text = "|"
  .Execute
End With

' If you find a "|", the starting file is a list file
startIsListFile = rng.Find.Found

' How many Word files are loaded?
docs = Word.Documents.Count
If docs = 1 Then MyError = 10: GoTo myErrorReport

' First, count how many of the files are F&R lists
' and how many are texts to be worked on.

NoOfLists = 0
NoOfTexts = 0
For i = 1 To docs
  myFile(i) = Documents(i).Name
  Documents(myFile(i)).Activate
  Set rng = ActiveDocument.Content

  ' Finding a '|' in the first 50 characters means it's a list
  rng.End = rng.Start + 50
  With rng.Find
    .Text = "|"
    .Execute
  End With
  If rng.Find.Found Then
    NoOfLists = NoOfLists + 1
    theList = myFile(i)
  Else
    NoOfTexts = NoOfTexts + 1
    theText = myFile(i)
  End If
```

```
Next i

If NoOfLists > 1 And NoOfTexts = 1 And Not startIsListFile _
   Then MyError = 8: GoTo myErrorReport
If NoOfLists = 1 And NoOfTexts > 1 And startIsListFile _
   Then MyError = 7: GoTo myErrorReport
If NoOfLists > 1 And NoOfTexts > 1 Then MyError = 9: _
   GoTo myErrorReport

theStart:
' At this point, theList holds the name of the F&R list
' and theText is the file to work on.

Documents(theList).Activate

' Check that last line is blank, and add CrLf if not
'lastPara = ActiveDocument.Paragraphs.Count
'Set rng = ActiveDocument.Paragraphs(lastPara).Range
'Ftext = rng
'If Len(Ftext) > 1 Then rng.InsertAfter vbCr
'????????????????????????????????????????????????????????????????????????????????


' Create the list of F&Rs
FRitem = 0: hilight = 1: Ignore = False
' FRitem is used to count the actual lines that are F&R commands

lastLine = ActiveDocument.Paragraphs.Count
For i = 1 To lastLine
 ' Look through the list for things that aren't actual F&Rs
   Do
      Do
         Set rng = ActiveDocument.Paragraphs(i).Range
         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, "light") > 0 Then
          ' Check for | Highlight command
            Select Case LCase(Right(allLine, 3))
               Case "= 0": hilight = 0
               Case "= 1": hilight = 1
               Case " no": hilight = 0
               Case "yes": hilight = 1
               Case Else
               MyError = 3: GoTo myErrorReport
            End Select
         End If
         If InStr(allLine, "nore") > 0 Then
          ' Check for | Ignore command
            Select Case LCase(Right(allLine, 3))
               Case "yes": Ignore = True
               Case " no": Ignore = False
```

```
            Case Else: MyError = 4: GoTo myErrorReport
          End Select
        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, 2) = "##" Then i = lastLine + 1

  ' It's an F&R line, so check for highlighting
    If i <= lastLine And Len(allLine) > 1 Then
      lineStart = rng.Start
      lineEnd = rng.End

      rng.End = lineStart + 1
      hilightColour = rng.HighlightColorIndex * hilight
      rng.End = lineEnd

      If hilightColour > 7 Then MyError = 1: GoTo myErrorReport

     ' Has it got a pad character 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
       ' What style is the Find in?
        fStyle = rng.Style
        If fStyle = "Normal" Then fStyle = ""

        rng.End = lineStart + 1
       ' Check format & type colour of first char of Find
        Fopposite = (rng.Font.Color = wdColorAutomatic)
        fItalic = rng.Italic
        fBold = rng.Bold
        fSuper = rng.Font.Superscript
        fSub = rng.Font.Subscript
        fUline = rng.Underline
        fSmall = rng.Font.SmallCaps

        i = i + 1
        Set rng = ActiveDocument.Paragraphs(i).Range
        rng.End = rng.End - 1
        lineEnd = rng.End
        rText = rng
        If InStr(rText, "|") > 0 Then MyError = 2: GoTo myErrorReport
       ' What style is the Replace in?
        rStyle = rng.Style
        If rStyle = "Normal" Then rStyle = ""
        rng.Start = lineEnd - 1

       ' Check format & type colour of last char of Find
        Ropposite = (rng.Font.Color = wdColorAutomatic)
        rItalic = rng.Italic
        rBold = rng.Bold
        rSuper = rng.Font.Superscript
```

```
      rSub = rng.Font.Subscript
      rUline = rng.Underline
      rSmall = rng.Font.SmallCaps

    Else
     ' It's all on one line, so no style info
      fStyle = ""
      rStyle = ""
     ' Chop up the line into F and R
      fText = Left(allLine, padPosition - 1)
      rText = Right(allLine, Len(allLine) - padPosition)

      rng.End = lineStart + 1
     ' Check format & type colour of first char of Find
      Fopposite = (rng.Font.Color = wdColorAutomatic)
      fItalic = rng.Italic
      fBold = rng.Bold
      fSuper = rng.Font.Superscript
      fSub = rng.Font.Subscript
      fUline = rng.Underline
      fSmall = rng.Font.SmallCaps

      rng.End = lineEnd
      rng.Start = lineEnd - 1
     ' Check format & type colour of last char of Find
      Ropposite = (rng.Font.Color = wdColorAutomatic)
      rItalic = rng.Italic
      rBold = rng.Bold
      rSuper = rng.Font.Superscript
      rSub = rng.Font.Subscript
      rUline = rng.Underline
      rSmall = rng.Font.SmallCaps
    End If
  ' Clip off MatchCase and Wildcard indicators
     mchWild = False
     mchCase = True
    If Left(fText, 1) = "¬" Then
      fText = Right(fText, Len(fText) - 1)
      mchCase = False
    End If
    If Left(fText, 1) = "~" Then
      fText = Right(fText, Len(fText) - 1)
      mchWild = True
    End If
    If Left(fText, 1) = "¬" Then
      fText = Right(fText, Len(fText) - 1)
      mchCase = False
    End If
    FRitem = FRitem + 1
  ' Save all the F&R info in arrays
    findArray(FRitem) = fText
    replaceArray(FRitem) = rText
    styleArray(FRitem, 1) = fStyle
    styleArray(FRitem, 2) = rStyle
    hlArray(FRitem) = hilightColour
    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) = Fopposite
      funct(FRitem, 16) = Ropposite
    ' ^p is not allowed in wildcard searches!
      If mchWild And InStr(fText, "^" & "p") > 0 _
         Then MyError = 5: GoTo myErrorReport
    ' You can't do case insensitive AND wildcard
      If mchWild And mchCase = False Then _
         MyError = 6: GoTo myErrorReport
   End If
Next i

lastItem = FRitem

'#################################
' ActiveDocument.Close
'#################################

Windows(theText).Activate
colourOffset = 7

Application.ScreenUpdating = True



' Get the data out of the arrays
For FRitem = 1 To lastItem
   fText = findArray(FRitem)
   rText = replaceArray(FRitem)
   fStyle = styleArray(FRitem, 1)
   rStyle = styleArray(FRitem, 2)
   hilightColour = hlArray(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)
```

```
    Fopposite = funct(FRitem, 15)
    Ropposite = funct(FRitem, 16)

  ' funnyCode means Ftext is to be blank
    If fText = funnyCode Then fText = ""
  ' A blank in the R text?
    If Left(rText, Len(funnyCode)) = funnyCode Then
     ' If so, are there any ChangeTo words?
       changeTo = Right(rText, Len(rText) - Len(funnyCode))
       rText = ""
    End If

    Set rng = ActiveDocument.Content

  ' Do the F&R with the appropriate conditions set
    With rng.Find
      .ClearFormatting
      .Text = fText
      If fBold Then .Font.Bold = Fopposite
      If fItalic Then .Font.Italic = Fopposite
      If fSuper Then .Font.Superscript = Fopposite
      If fSub Then .Font.Subscript = Fopposite
      If fUline Then .Font.Underline = Fopposite
      If fSmall Then .Font.SmallCaps = fSmall
      If fStyle > "" Then .Style = fStyle
      If Ignore = True Then .Highlight = False
      With .Replacement
        .ClearFormatting
        .Text = rText
        If rBold Then .Font.Bold = Ropposite
        If rItalic Then .Font.Italic = Ropposite
        If rSuper Then .Font.Superscript = Ropposite
        If rSub Then .Font.Subscript = Ropposite
        If rUline Then .Font.Underline = Ropposite
        If rSmall Then .Font.SmallCaps = rSmall
        If rStyle > "" Then .Style = rStyle
        If hilightColour > 0 And rText > "" Then
          .Font.Color = hilightColour + colourOffset
        End If
        If InStr(changeTo, "bold") Then .Font.Bold = True
        If InStr(changeTo, "italic") Then .Font.Italic = True
        If InStr(changeTo, "super") Then .Font.Superscript = True
        If InStr(changeTo, "sub") Then .Font.Subscript = True
        If InStr(changeTo, "under") Then .Font.Underline = True
      End With
      .MatchWildcards = mchWild
      .MatchCase = mchCase
      .Execute Replace:=wdReplaceAll
    End With
  Next

  ' In the F&Rs above, a highlight was signalled
  ' by using an obscure font colour unlikely to be
  ' used in the text, so now we convert each font
  ' colour back to a highlight colour.

  If hilight = 1 Then
```

```
  For HLcolour = 2 To 7
    Options.DefaultHighlightColorIndex = HLcolour

    Set rng = ActiveDocument.Content
    With rng.Find
      .ClearFormatting
      .Font.Color = HLcolour + colourOffset
      .Replacement.ClearFormatting
      .Replacement.Font.Color = wdColorAutomatic
      .Replacement.Highlight = True
      .Text = ""
      .Replacement.Text = ""
      .Execute Replace:=wdReplaceAll
    End With
   Next
End If

'########################################
 Selection.EndKey Unit:=wdStory
 Selection.TypeParagraph
 Selection.TypeParagraph
 Selection.TypeText Text:="Auto-edited by P Beverley, "
 Selection.InsertDateTime DateTimeFormat:="dd MMMM yyyy.", InsertAsField:=False
 Selection.TypeParagraph

 Selection.HomeKey Unit:=wdStory

'########################################

FinishHere:
' Restore highlight colour to normal
Options.DefaultHighlightColorIndex = OldColour
Exit Sub

' Warn the user about problems that the macro has detected
myErrorReport:
If MyError <= 6 Then rng.Select
Selection.HomeKey Unit:=wdLine
Select Case MyError
   Case 1: myPrompt = "Is the whole line highlighted?"
   Case 2: myPrompt = "No matching replace text"
   Case 3: myPrompt = "A |Highlight line should say 'yes' or 'no'."
   Case 4: myPrompt = "An |Ignore line should say 'yes' or 'no'"
   Case 5: myPrompt = "Sorry, Word can't use ^p in a wildcard search." _
      & vbCrLf & vbCrLf & "On Word for Mac, try [^13]." & vbCrLf & _
      vbCrLf & "On Word for Windows, try ^13."
   Case 6: myPrompt = "Sorry, Word can't do case insensitive searches with wildcards."
   Case 7: myPrompt = "Too many files." & vbCrLf & vbCrLf & _
      "Please switch to the text file and rerun the macro."
   Case 8: myPrompt = "Too many files." & vbCrLf & vbCrLf & _
      "Please switch to the F&R list file and rerun the macro."
   Case 9: myPrompt = "Too many open files." & vbCr & vbCr & _
      "Please close unused files and rerun the macro."
   Case 10: myPrompt = "Please create or load an F&R list."
   Case Else: myPrompt = "Progam error; please inform Paul Bev."
End Select
MsgBox myPrompt, vbOKOnly + vbExclamation, "FRedit"
```

```
Exit Sub

' Errors that Word generates end up here
ReportIt:
' If we've found the F&R list, select it
If theList > "" Then
   Documents(theList).Activate
   Set rng = ActiveDocument.Range

 ' and look for the current line in the F&R list
 ' which is probably where the problem lies.
   Selection.HomeKey Unit:=wdStory
   With Selection.Find
      .ClearFormatting
      .Text = fText & "|" & rText
      .Forward = True
      .MatchWildcards = False
      .Execute
   End With

 ' If we couldn't find it, maybe it's a two-line F&R
   If Not Selection.Find.Found Then
      With Selection.Find
         .ClearFormatting
         .Text = fText & vbCrLf & rText
         .Forward = True
         .MatchWildcards = False
         .Execute
      End With
   End If
End If
Selection.HomeKey Unit:=wdLine

' Display Word's error message
MsgBox Err.Description, vbExclamation, "Error from Word"
Resume FinishHere


End Sub

#########################################################
Sub EndnoteDeleteDoubleSpace()
  For i = 1 To ActiveDocument.Endnotes.Count
    Set rng = ActiveDocument.Endnotes(i).Range
    rng.Start = rng.Start - 1
    rng.End = rng.Start + 1
    mychar = rng
    myASC = 0
    If Len(mychar) > 0 And i > 1 Then myASC = Asc(mychar)
    If myASC = 13 Or myASC = 11 Then rng.Delete

    Set rng = ActiveDocument.Endnotes(i).Range
    rng.Start = rng.End - 1
    KeepGoing = True
    Do
      mychar = rng
      If Len(mychar) > 0 Then
```

```
      myASC = Asc(mychar)
    Else
      myASC = 0
    End If
    If myASC = 13 Or myASC = 11 Or myASC = 32 Then
      rng.Delete
      rng.Start = rng.Start - 1
    Else
      KeepGoing = False
    End If
  Loop Until KeepGoing = False
 Next
End Sub
```

##
ScreenUpdating 1/0
Screenrefresh

You can convert footnotes to endnotes and viceversa:

Insert -> Footnotes -> Options
Page 354 says:

"Make sure you're in Normal view"

"You can convert endnotes to footnotes or vice versa by clicking the
Insert menu and then clicking ”Footnote,• ”Options,• and ”Convert.•"

but I can't get it to work.

##
FindThis macros done with rng, not Selection:

```
Sub FindThisFwdOO()
' 17.02.10
' <Shift-Ctrl-Alt-Down>
If Selection.Start = Selection.End Then MsgBox ("Select text"): Exit Sub
thisBit = Selection
If Asc(thisBit) <> 32 Then thisBit = Trim(Selection)
Set rng = ActiveDocument.Range
rng.Start = Selection.End
With rng.Find
  .Wrap = False
  .Text = thisBit
  .MatchCase = False
  .Forward = True
  .Execute
End With
rng.Select
'Add the find item to the F&R dialogue
Selection.Find.Text = thisBit
End Sub

Sub FindThisBackOO()
' 17.02.10
' <Shift-Ctrl-Alt-Up>
If Selection.Start = Selection.End Then MsgBox ("Select text"): Exit Sub
```

```
thisBit = Selection
If Asc(thisBit) <> 32 Then thisBit = Trim(Selection)
Set rng = ActiveDocument.Range
rng.Start = Selection.End
With rng.Find
  .Wrap = False
  .Text = thisBit
  .MatchCase = False
  .Forward = False
  .Execute
End With
rng.Select
'Add the find item to the F&R dialogue
Selection.Find.Text = thisBit
End Sub


Sub FindThisFwdMarkOO()
' 17.02.10
' <Ctrl-Alt-Shift-Num-Plus>
If Selection.Start = Selection.End Then MsgBox ("Select text"): Exit Sub
thisBit = Selection
If Asc(thisBit) <> 32 Then thisBit = Trim(Selection)
Selection.Start = Selection.End
Selection.TypeText ("[[[")
Set rng = ActiveDocument.Range
rng.Start = Selection.End
With rng.Find
  .Wrap = False
  .Text = thisBit
  .MatchCase = False
  .Forward = True
  .Execute
End With
rng.Select
'Add the find item to the F&R dialogue
Selection.Find.Text = thisBit

End Sub

Sub FindThisBackMarkOO()
' 17.02.10
' <Ctrl-Alt-Shift-Num-Plus>
If Selection.Start = Selection.End Then MsgBox ("Select text"): Exit Sub
thisBit = Selection
If Asc(thisBit) <> 32 Then thisBit = Trim(Selection)
Selection.Start = Selection.End
Selection.TypeText ("[[[")
Set rng = ActiveDocument.Range
rng.Start = Selection.End
With rng.Find
  .Wrap = False
  .Text = thisBit
  .MatchCase = False
  .Forward = False
  .Execute
End With
```

```
rng.Select
'Add the find item to the F&R dialogue
Selection.Find.Text = thisBit
End Sub
```

##

Find Count with and without trimming

```
Sub FindCount()
' 08.02.10
' <Alt-?>
If Selection.Start = Selection.End Then
  MsgBox ("Select some text")
  Exit Sub
End If

thisBit = Selection

Set rng = ActiveDocument.Range
thisMany = -1
Do
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .MatchCase = False
    .Text = thisBit
    .Replacement.Text = ""
    .Execute
  End With
  thisMany = thisMany + 1
Loop Until rng.Find.Found = False

' If selected text contained spaces, make that clear ...
showThis = Replace(thisBit, " ", "_") & " – " & thisMany

Set rng = ActiveDocument.Range
thisMany = -1
Do
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = thisBit
    .MatchCase = True
    .Replacement.Text = ""
    .Execute
  End With
  thisMany = thisMany + 1
Loop Until rng.Find.Found = False

' If selected text contained spaces, make that clear ...
```

```
showThis = showThis & "  (" & thisMany & ")   "

'... and count without any outer spaces
If thisBit <> Trim(thisBit) Then
  thisBit = Trim(thisBit)

  Set rng = ActiveDocument.Range
  thisMany = -1
  Do
    With rng.Find
      .ClearFormatting
      .Replacement.ClearFormatting
      .Text = thisBit
      .MatchCase = False
      .Replacement.Text = ""
      .Execute
    End With
    thisMany = thisMany + 1
  Loop Until rng.Find.Found = False
  showThis = showThis & "        " & thisBit & " – " & thisMany

  Set rng = ActiveDocument.Range
  thisMany = -1
  Do
    With rng.Find
      .ClearFormatting
      .Replacement.ClearFormatting
      .MatchCase = True
      .Text = thisBit
      .Replacement.Text = ""
      .Execute
    End With
    thisMany = thisMany + 1
  Loop Until rng.Find.Found = False
  showThis = showThis & "  (" & thisMany & ")   "
End If

MsgBox showThis & showThisCase
End Sub

#####################################################

Changing things pargraph by paragraph

Sub JustifyOFF()
For Each myPara In ActiveDocument.Paragraphs
 If myPara.Range.ParagraphFormat.Alignment = wdAlignParagraphJustify Then
   myPara.Range.ParagraphFormat.Alignment = wdAlignParagraphLeft
 End If
Next
End Sub



#####################################################
Sub Macro1()
' Was used for indexing
```

```
'
If Left(ActiveDocument.Name, 5) = "Index" Then
  Selection.MoveDown Unit:=wdLine, Count:=1
  Selection.EndKey Unit:=wdLine
  Windows("FReditList2").Activate
    Selection.MoveDown Unit:=wdParagraph, Count:=2
    Selection.MoveUp Unit:=wdParagraph, Count:=1, Extend:=wdExtend
  hhh = Selection
  hhh = Left(hhh, Len(hhh) - 1)
  Windows("TextAll").Activate
  Selection.HomeKey Unit:=wdStory
  Selection.Find.ClearFormatting
  With Selection.Find
    .Text = hhh
    .Replacement.Text = ""
    .Forward = True
    .Wrap = wdFindContinue
    .Format = False
    .MatchCase = False
    .MatchWholeWord = False
    .MatchWildcards = False
    .MatchSoundsLike = False
    .MatchAllWordForms = False
  End With
  Selection.Find.Execute
Else
  Windows("Index_Edited_01").Activate
End If
End Sub


##########################################################
Numbering the paragraphs

Sub NumberParas()
For p = 1 To ActiveDocument.Paragraphs.Count
  ActiveDocument.Paragraphs(p).Range.Select
  Selection.MoveLeft Unit:=wdCharacter
  Selection.TypeText Text:=LTrim(Str(p)) & ") "
Next p
End Sub


##########################################################


Macro to create ASCII code sheet

Sub test()
mySkip = 45
For i = 33 To 32 + mySkip
  fText = Str(i) & vbTab & Chr(i) & vbTab
  fText = fText & Str(i + mySkip) & vbTab & Chr(i + mySkip) & vbTab
  fText = fText & Str(i + 2 * mySkip) & vbTab & Chr(i + 2 * mySkip) & vbTab
  fText = fText & Str(i + 3 * mySkip) & vbTab & Chr(i + 3 * mySkip) & vbTab
  lastSkip = i + 4 * mySkip
  If lastSkip < 256 Then
```

```
    fText = fText & Str(lastSkip) & vbTab & Chr(lastSkip) & vbTab
  End If
  fText = fText & vbCrLf
  Selection.InsertAfter fText
Next i

End Sub
```

##

Sue Peter's macro to tag A, B, C and D headings

```
For i = 1 To ActiveDocument.Paragraphs.Count
  Set rng = ActiveDocument.Paragraphs(i).Range
  rng.End = rng.End - 1
  myCode = Left(rng.Style, 1)
  mySecond = Mid(rng.Style, 2, 1)
  If Asc(myCode) > 64 And Asc(myCode) < 69 And mySecond = "" Then
    rng.InsertAfter "</" & myCode & "hd>"
    rng.InsertBefore "<" & myCode & "hd>"
  End If
Next
```

##

Centres the chapter heads

```
Sub ChapterHeads()
Do
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = "<+CH>"
    .Replacement.Text = ""
    .Forward = True
    .Wrap = False
    .MatchWildcards = False
    .Execute
  End With
  If Selection.Find.Found = True Then
    GoGo = True
    Selection.ParagraphFormat.Alignment = wdAlignParagraphCenter
  Else
    GoGo = False
  End If
Loop Until GoGo = False

End Sub
```

##

```
  DocName = ActiveDocument.FullName
  NewDocName = Replace(DocName, ".doc", "_COM.doc")

  If ActiveDocument.Comments.Count >= 1 Then
    ActiveDocument.StoryRanges(wdCommentsStory).Copy
```

```
  End If

  Documents.Add
  Selection.Paste
  ActiveDocument.SaveAs FileName:=NewDocName
End Sub



theList = ""
Set rng = ActiveDocument.Content
totWords = rng.Words.Count
myPrompt = "Press <Ctrl-Break> to stop.    Words to go: "
StatusBar = myPrompt & totWords
For wrd = 1 To totWords
  thisWord = rng.Words(wrd)
  If Len(thisWord) < 2 Then GoTo nextWord

  thisWord = Trim(thisWord)
  If Asc(thisWord) > 90 Then GoTo nextWord

  part = Right(thisWord, Len(thisWord) - 1)
  If LCase(part) = part Then
    StatusBar = myPrompt & totWords - wrd
    GoTo nextWord
  End If
  If InStr(theList, thisWord) > 0 Then GoTo nextWord

' Then and only then add it to the list
  theList = theList & thisWord & vbCrLf
nextWord:
Next wrd
Documents.Add
Selection.InsertAfter Text:=theList
Selection.WholeStory
Selection.Sort
Selection.HomeKey Unit:=wdStory
Selection.Delete


Fiddle with different styles, adding indent to the paragraph
style following (or not):

Sub Harvey()
  StyleList = "Heading 1, Heading 2, Heading 3, and any more you want"
  NoIndentStyle = "Body Text"
  IndentStyle = "Body Text First Indent"

  For i = 1 To (ActiveDocument.Paragraphs.Count - 1)
    StyleNow = ActiveDocument.Paragraphs(i).Range.Style
    StyleNext = ActiveDocument.Paragraphs(i + 1).Range.Style

    If InStr(StyleList, StyleNow) And StyleNext = IndentStyle Then
      ActiveDocument.Paragraphs(i + 1).Range.Style = NoIndentStyle
    End If
  Next
```

End Sub

Select a paragraph:

```
Selection.MoveUp Unit:=wdParagraph, Count:=1
Selection.MoveDown Unit:=wdParagraph, Count:=1, Extend:=wdExtend
```

More needs to be transferred into this file and then the
source file put into zzBits.

==========

```
Sub FindAdverbs()
Dim i As Integer
Dim CurrentString As String
For i = 1 To ActiveDocument.Words.Count
CurrentString = Trim(ActiveDocument.Words(i).Text)
If Right(CurrentString, 2) = "ly" Then
With ActiveDocument.Words(i)
.Italic = Not .Italic
.Bold = Not .Bold
End With
End If
Next i
End Sub
```

This

==========

==========

```
Sub Exclude()
Dim sAddWords As String
sAddWord = Trim(Selection.Text)
ChangeFileOpenDirectory _
"C:\Program Files\Common Files\Microsoft Shared\Proof\"
' The file name in the next line should be changed so it
```

Click Here for More Tips Collections: http://store.VitalNews.com Page 237 WordTips: The Macros Working With Word‘s Tools

```
' reflects the proper exclude file name for your system
Documents.Open FileName:="mssp2_en.exc", _
ConfirmConversions:=False, ReadOnly:=False, _
AddToRecentFiles:=False, PasswordDocument:="", _
PasswordTemplate:="", Revert:=False, _
WritePasswordDocument:="", WritePasswordTemplate:="", _
Format:=wdOpenFormatAuto
Selection.TypeText Text:=sAddWord
Selection.TypeParagraph
ActiveDocument.Close SaveChanges:=wdSaveChanges, _
OriginalFormat:=wdOriginalDocumentFormat
```

==========

This first macro is mine - it finds the directory of the currently active document.

```
Sub Test()
myFullFileName = ActiveDocument.FullName
myFileName = ActiveDocument.Name
If myFullFileName = myFileName Then
  MsgBox ("File not yet saved.")
Else
' This returns the filepath of the directory containing
' the active document
  myDirName = Left(myFullFileName, InStr(myFullFileName, myFileName) - 2)
  MsgBox myDirName
End If
End Sub
```

From: http://www.word.mvps.org/FAQs/MacrosVBA/ModifyRecordedMacro.htm

Suppose you record a macro that opens a document and then does something to it, such as changing the view. The beginning of the macro may look like this: 1

```
Sub Macro2()
  Documents.Open FileName:= "Lorem.doc" , _
      ConfirmConversions:= False , _
      ReadOnly:= False , AddToRecentFiles:= False , _
      PasswordDocument:= "" , PasswordTemplate:= "" , _
      Revert:= False , WritePasswordDocument:= "" , _
      WritePasswordTemplate:= "" , _
      Format:=wdOpenFormatAuto, XMLTransform:= ""
  ' more code, for example ...
  ActiveWindow.View = wdPrintView
End Sub
```

The recorded macro for opening a document, like the recording of the Format > Font dialog, contains unneeded things. In this case, they are parameters that contain information about the file, such as a password. The only parameter that's necessary is the FileName. You can remove the other parameters from the command, and Word will use its default values for them.

A more important problem is that every time you run the recorded macro, it will open the same document. This may be what you intend, but more likely you want the macro to let you choose which document to open.

One way to get the file's name into the macro is to display an input box, where you can type it in. The InputBox function shows a message box with a text entry field, and its result is the name that you type into the field.

```
Sub Macro2A()
  Dim MyFileName As String
  MyFileName = InputBox( "Enter file name to open:" , _
     "Open a Document" )
  If MyFileName <> "" Then
    Documents.Open FileName:=MyFileName
    ' more code, for example ...
    ActiveWindow.View = wdPrintView
  End If
End Sub
```

But this isn't very friendly. There's a good chance of making a typing mistake. And if the document isn't in the currently active folder, you have to type the entire path to it. A better idea is to use the File > Open dialog that's already built into Word, which lets you browse to and select the proper document. When you click the OK button in the dialog, Word opens the selected document. The macro is simpler, too, because it doesn't need a separate Documents.Open statement&8212;the dialog handles it all for you.

```
Sub Macro2B()
  If Dialogs(wdDialogFileOpen).Show = - 1 Then
    ' more code, for example ...
    ActiveWindow.View = wdPrintView
  End If
End Sub
```

The word "Dialogs" in this code refers to a list of all of Word's built-in dialogs. Each dialog has a name that starts with "wdDialog". In this case, wdDialogFileOpen is the name of the built-in File > Open dialog, and the expression "Dialogs(wdDialogFileOpen)" selects that particular dialog from the list. To see all of the possible names, press F2 in the VBA editor to display the Object Browser, type wdDialog into the search box, and press Enter.

The word ".Show" refers to a method of the dialog. A method is an action that can be done&8212;the .Show method causes the dialog to appear and execute (carry out its function).

Many methods also have a value after they execute, which tells the macro something about what just happened (this is called "returning" the value). In this case, if you click the OK button in the dialog then the .Show method returns the value &8211;1, but if you click the Cancel button or the X in the title bar then .Show returns the value 0. The VBA help topic for each method tells you what values that method can return and what they mean. You can use the returned value in an If statement, as in Macro2B, to decide what to do.

Similar changes to recorded code let you make macros that save files to variable locations, search for variable strings, and

many other unrecordable variations. You can find out more at Getting help with calling Word's built-in dialogs using VBA.

==========

SEARCHING FOR NON-BLACK TEXT

When searching for text in a document, it is easy to search for formats such as “bold” or “italic.” You can also search for “not bold” and “not italic.” You can search for text of a specific color, but Word does not allow you to search for text other than a specific color. For instance, you cannot search for text that is “not black.”

If you need to search for text that is not black, then the best way to do so is with a macro. Consider the following macro, which prompts you for your search text, and then looks for the first non-black instance of that text.

```
Sub FindNotBlack()
  With Selection.Find
  .ClearFormatting
  .Text = InputBox(prompt:="Enter the search text.", _
  Title:="Find Nonblack Text")
  Do While .Execute
  With Selection.Font
  If (.Color <> wdColorAutomatic) And _
  (.Color <> wdColorBlack) Then
  MsgBox "Found"
  Exit Sub
  End If
  End With
  Loop
  End With
End Sub
```

Note that the macro checks to see if the color of the matched text is different from black (wdColorBlack) and from the automatic color (wdColorAutomatic). This is because the automatic color is black on most systems.

==========

FINDING TEXT BOXES

Word has a powerful search and replace capability that lets you search for virtually anything in your document. Word even includes codes you can use to search for special items. (Click the Special button in the Find and Replace dialog to see what codes are available.) One thing you cannot search for, however, is text boxes. There is no special code that allows you find text boxes.

You can, however, use a macro to look through a document and stop when it finds a text box. The following macro stops on each text box it finds and asks the user if that is the text box wanted.

```
Sub SearchTextBox()
Dim shp As Shape
Dim sTemp As String
Dim iAnswer As Integer
For Each shp In ActiveDocument.Shapes
If shp.Type = msoTextBox Then
shp.Select
Selection.ShapeRange.TextFrame.TextRange.Select
```

```
sTemp = Selection.Text
sTemp = Left(sTemp,20)
iAnswer = MsgBox("Box contains text beginning with:" & vbCrLf _
& sTemp & vbCrLf & "Stop here?", vbYesNo, "Located Text Box")
If iAnswer = vbYes Then Exit For
End If
Next
End Sub

==========================================
Sub Test()
Dim v As Variable
varExists = False
For Each v In ActiveDocument.Variables
  If v.Name = "tColour" Then varExists = True: Exit For
Next v

If varExists = False Then ActiveDocument.Variables.Add "tColour", 7

searchColour = ActiveDocument.Variables("tColour")

' If no text is selected, search for next highlight
If Selection.Start = Selection.End Then GoTo FindNext

' If some text is selected, see what colour it is;
' then go find more text of that colour.
selColour = Selection.Font.Color
searchColour = selColour
ActiveDocument.Variables("tColour") = searchColour

FindNext:
Selection.Start = Selection.End
HlightFound = False
With Selection.Find
 .ClearFormatting
 .Replacement.ClearFormatting
 .Text = ""
 .Font.Color = searchColour
 .Wrap = wdFindContinue
 .Replacement.Text = ""
 .Forward = True
 .MatchWildcards = False
 .Execute
End With

End Sub

Sub FindHighlight()
Dim v As Variable
varExists = False
For Each v In ActiveDocument.Variables
  If v.Name = "hColour" Then varExists = True: Exit For
Next v

If varExists = False Then ActiveDocument.Variables.Add "hColour", 7

searchColour = ActiveDocument.Variables("hColour")
```

```
' If no text is selected, search for next highlight
If Selection.Start = Selection.End Then GoTo FindNext

' If some text is selected, see what colour it is;
' then go find more text of that colour.
selColour = Selection.Range.HighlightColorIndex
If selColour > 0 Then
  searchColour = selColour
  ActiveDocument.Variables("hColour") = searchColour
  GoTo FindNext
End If
Selection.Start = Selection.End

Do
  With Selection.Find
   .ClearFormatting
   .Replacement.ClearFormatting
   .Text = ""
   .Highlight = True
   .Wrap = wdFindContinue
   .Replacement.Text = ""
   .Forward = True
   .MatchWildcards = False
   .Execute
  End With

  myResponse = MsgBox("This colour?", vbQuestion + vbYesNoCancel)
  If myResponse = vbCancel Then Exit Sub
  If Selection.Find.Found = False Then Exit Sub
Loop Until myResponse = vbYes
searchColour = Selection.Range.HighlightColorIndex
ActiveDocument.Variables("hColour") = searchColour

FindNext:
Selection.Start = Selection.End
HlightFound = False
Do
  With Selection.Find
   .ClearFormatting
   .Replacement.ClearFormatting
   .Text = ""
   .Highlight = True
   .Wrap = wdFindContinue
   .Replacement.Text = ""
   .Forward = True
   .MatchWildcards = False
   .Execute
  End With
'  Selection.Start = Selection.End - 1
  thisColour = Selection.Range.HighlightColorIndex
  If thisColour = searchColour Or thisColour > 100 Then
     HlightFound = True
  End If
 ' MsgBox thisColour
Loop Until HlightFound = True Or Selection.Find.Found = False
If thisColour > 100 Then
```

```
  Selection.Start = Selection.Start - 1
  Selection.End = Selection.Start + 1
  Do
    Selection.Start = Selection.Start + 1
    Selection.End = Selection.End + 1
    thisColour = Selection.Range.HighlightColorIndex
  Loop Until thisColour = searchColour
  Do
    Selection.End = Selection.End + 1
    thisColour = Selection.Range.HighlightColorIndex
  Loop Until thisColour <> searchColour
    Selection.End = Selection.End - 1
End If

End Sub
```

==========

If you want to do it in one easy step, however, you can use this VBA macro:

```
Sub Reverse()
Selection.StartOf Unit:=wdParagraph, Extend:=wdMove
Selection.MoveEnd Unit:=wdParagraph
Selection.ParagraphFormat.Shading.BackgroundPatternColorIndex = wdBlack
Selection.Font.ColorIndex = wdWhite
End Sub
```

All you need to do is place the insertion point somewhere within the paragraph, and then run the macro. As with most any macro, you can assign it to a toolbar button or a shortcut key so you

==========

All that rubbish below can be done with:

```
myDirName = ActiveDocument.Path
If myDirName > "" Then
  ChangeFileOpenDirectory myDirName

' Now check that this really has set the current directory.
  MsgBox CurDir
Else
  MsgBox ("File not yet saved.")
End If
```

And you can then open and save files just by the filename, knowing
which directory they are in.

```
Sub Test()
myFullFileName = ActiveDocument.FullName
myFileName = ActiveDocument.Name
If myFullFileName = myFileName Then
  MsgBox ("File not yet saved.")
Else
' This returns the filepath of the directory containing
' the active document
  myDirName = Left(myFullFileName, InStr(myFullFileName, myFileName) - 2)
' Now make this the current directory
  ChangeFileOpenDirectory myDirName
```

```
' And now you can do things in that directory!
  Documents.Open FileName:="wordtips"
' Just checking that this really has set the current directory.
  MsgBox CurDir
End If
End Sub


==========================================
Sub Test()
Dim v As Variable
varExists = False
For Each v In ActiveDocument.Variables
  If v.Name = "tColour" Then varExists = True: Exit For
Next v

If varExists = False Then ActiveDocument.Variables.Add "tColour", 7

searchColour = ActiveDocument.Variables("tColour")

' If no text is selected, search for next highlight
If Selection.Start = Selection.End Then GoTo FindNext

' If some text is selected, see what colour it is;
' then go find more text of that colour.
selColour = Selection.Font.Color
searchColour = selColour
ActiveDocument.Variables("tColour") = searchColour

FindNext:
Selection.Start = Selection.End
HlightFound = False
With Selection.Find
 .ClearFormatting
 .Replacement.ClearFormatting
 .Text = ""
 .Font.Color = searchColour
 .Wrap = wdFindContinue
 .Replacement.Text = ""
 .Forward = True
 .MatchWildcards = False
 .Execute
End With

End Sub

Sub FindHighlight()
Dim v As Variable
varExists = False
For Each v In ActiveDocument.Variables
  If v.Name = "hColour" Then varExists = True: Exit For
Next v

If varExists = False Then ActiveDocument.Variables.Add "hColour", 7

searchColour = ActiveDocument.Variables("hColour")
```

```
' If no text is selected, search for next highlight
If Selection.Start = Selection.End Then GoTo FindNext

' If some text is selected, see what colour it is;
' then go find more text of that colour.
selColour = Selection.Range.HighlightColorIndex
If selColour > 0 Then
  searchColour = selColour
  ActiveDocument.Variables("hColour") = searchColour
  GoTo FindNext
End If
Selection.Start = Selection.End

Do
  With Selection.Find
   .ClearFormatting
   .Replacement.ClearFormatting
   .Text = ""
   .Highlight = True
   .Wrap = wdFindContinue
   .Replacement.Text = ""
   .Forward = True
   .MatchWildcards = False
   .Execute
  End With

  myResponse = MsgBox("This colour?", vbQuestion + vbYesNoCancel)
  If myResponse = vbCancel Then Exit Sub
  If Selection.Find.Found = False Then Exit Sub
Loop Until myResponse = vbYes
searchColour = Selection.Range.HighlightColorIndex
ActiveDocument.Variables("hColour") = searchColour

FindNext:
Selection.Start = Selection.End
HlightFound = False
Do
  With Selection.Find
   .ClearFormatting
   .Replacement.ClearFormatting
   .Text = ""
   .Highlight = True
   .Wrap = wdFindContinue
   .Replacement.Text = ""
   .Forward = True
   .MatchWildcards = False
   .Execute
  End With

  If Selection.Range.HighlightColorIndex = searchColour Then
     HlightFound = True
  End If
Loop Until HlightFound = True Or Selection.Find.Found = False

End Sub
```

==========
they don’t use one of them.

```
Sub ScoreCard()
Dim iScore As Integer
Dim iTopScore As Integer
Dim WordList As Variant
Dim i As Integer
Dim sUnused As String
' Enter the words or phrases in the array below;
' each word or phrase in quotation marks and
' separated by commas
WordList = Array("Mr.", "jelly", "wince", _
"proper", "fix", "compound", "high and dry")
' Counts the number of words in the array
iTopScore = CInt(UBound(WordList)) + 1
iScore = iTopScore
' Counts the number of "misses"
sUnused = ""
For i = 1 To iTopScore
With Selection.Find
.Forward = True
.Wrap = wdFindContinue
.Format = False
.MatchCase = False
.MatchAllWordForms = False
.MatchWholeWord = True
.Execute FindText:=WordList(i - 1)
End With
If Selection.Find.Found = False Then
iScore = iScore - 1
sUnused = sUnused & vbCrLf & WordList(i - 1)
End If
Next i
' Displays the score
If iScore = iTopScore Then
sUnused = "All words and phrases were used."
Else
sUnused = "The following words and phrases" & _
" were not used:" & sUnused
End If
sUnused = vbCrLf & vbCrLf & sUnused
MsgBox Prompt:="The score is " & iScore & _
" of " & iTopScore & sUnused, Title:="What's the Score?"
End Sub
```

The macro displays a score for the composition and also displays any of the words or phrases th

==========

Date: Sat, 19 Sep 2009 10:43:07 +0100
From: Paul Beverley <paul@archivepub.co.uk>
To: Paul Beverley <paul.beverley@livtech.co.uk>
Subject: Bookmarks macro

```
Sub Macro2()
'
' Macro2 Macro
' Macro recorded 9/19/2009 by Paul Beverley
'
```

```
    With ActiveDocument.Bookmarks
        .Add Range:=Selection.Range, Name:="PaulsMark"
        .DefaultSorting = wdSortByName
        .ShowHidden = False
    End With
    Selection.EndKey Unit:=wdStory
    Selection.HomeKey Unit:=wdStory
    Selection.GoTo What:=wdGoToBookmark, Name:="PaulsMark"
    Selection.Find.ClearFormatting
    With Selection.Find
        .Text = "EL measure"
        .Replacement.Text = "EL measure"
        .Forward = True
        .Wrap = wdFindContinue
        .Format = False
        .MatchCase = True
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With
End Sub
```

All the best,

Paul

Paul Beverley (Archive Publications), LCGI, Advanced Member SfEP

Editing & proofreading +44-1603-722544 - http://www.archivepub.co.uk

==========

```
' First sort out which is the list of F&Rs
' and which the the text to be worked on
theList = ""
theText = ""
docs = Word.Documents.Count
' Examine the curently active file
fileName = ActiveDocument.Name
startFile = Left(fileName, InStr(fileName, ".") - 1)

ListNo = 0
textNo = 0
Select Case docs
 ' If only one doc, then find a list on disc ...
MyError = 8: GoTo myErrorReport   theText = startFile
 ' Check on disc for the list
   textFullName = ActiveDocument.FullName
   textFileName = ActiveDocument.Name
   dirName = Replace(textFullName, textFileName, "")

   listFullFilename = dirName + "PElist.doc"
   If Dir(listFullFilename) > "" Then
```

```
    ' If a PElist file exists, open it ...
      Documents.Open fileName:=listFullFilename
      If MsgBox("Is this the list?", vbQuestion _
        + vbYesNo, Title:="PreEdit") = vbNo Then Exit Sub
    Else
     ' ... otherwise create a new one
      Documents.Add
      ActiveDocument.SaveAs fileName:=listFullFilename
      Exit Sub:   ' Stop macro and let user create a list
    End If
    docs = 2
End If

For i = 1 To docs
    fileName = Application.Documents(i).Name + "."
 ' The dummy dot is in case there's a file w/o ".doc"
    fileName = Left(fileName, InStr(fileName, ".") - 1)
    Windows(fileName).Activate
    Set rng = ActiveDocument.Paragraphs(1).Range
    rng.End = rng.End - 1
    Ftext = rng
    If Left(Ftext, 8) = "| File: " Then
      ListNo = i
     ' If so, pick off the file name of the text
      If Len(Ftext) > 10 Then
        docText = Right(Ftext, Len(Ftext) - 8)
       ' Is docText the name of a loaded file
        For j = 1 To docs
          fileName = Application.Documents(j).Name + "."
          ' The dummy dot is in case there's a file w/o ".doc"
          fileName = Left(fileName, InStr(fileName, ".") - 1)
          If fileName = docText Then
            textNo = j
            j = docs + 1
            i = docs + 1
          End If
        Next j
      End If
    End If
    If i <= docs Then
      Set rng = ActiveDocument.Content
      rng.End = rng.Start + 50
      With rng.Find
        .Text = "|"
        .Execute
      End With
      If rng.Find.Found Then
        ListNo = i
        i = docs
      End If
    End If
Next i

' At this stage, we hope we've found the listNo,
' but if not, generate an error
If ListNo = 0 Then MyError = 8: GoTo myErrorReport
```

```
fileName = Application.Documents(ListNo).Name
theList = Left(fileName, InStr(fileName, ".") - 1)
Windows(theList).Activate

' If only two documents, they are Nos 1 and 2
If docs = 2 Then textNo = 3 - ListNo

' If we can't find the text, assume that it's the
'  file that was on screen when the macro was run.
If textNo = 0 Then
   theText = startFile
Else
   fileName = Application.Documents(textNo).Name
   theText = Left(fileName, InStr(fileName, ".") - 1)
End If

If theList = theText Then MyError = 7: GoTo myErrorReport

' Check that last line is blank and add CrLf if not
lastPara = ActiveDocument.Paragraphs.Count
Set rng = ActiveDocument.Paragraphs(lastPara).Range
Ftext = rng
If Len(Ftext) > 1 Then rng.InsertAfter vbCrLf

' Check and update | File: line
Set rng = ActiveDocument.Paragraphs(1).Range
Ftext = rng
If InStr(Ftext, "File:") > 0 Then rng.Delete
rng.InsertBefore "| File: " & theText & vbCrLf
Selection.HomeKey Unit:=wdStory
lastLine = ActiveDocument.Paragraphs.Count
```

==========

From: http://www.word.mvps.org/FAQs/MacrosVBA/ModifyRecordedMacro.htm

Suppose you record a macro that opens a document and then does something to it, such as changing the view. The beginning of the macro may look like this: 1

```
Sub Macro2()
  Documents.Open FileName:= "Lorem.doc" , _
     ConfirmConversions:= False , _
     ReadOnly:= False , AddToRecentFiles:= False , _
     PasswordDocument:= "" , PasswordTemplate:= "" , _
     Revert:= False , WritePasswordDocument:= "" , _
     WritePasswordTemplate:= "" , _
     Format:=wdOpenFormatAuto, XMLTransform:= ""
  ' more code, for example ...
  ActiveWindow.View = wdPrintView
End Sub
```

The recorded macro for opening a document, like the recording of the Format > Font dialog, contains unneeded

things. In this case, they are parameters that contain information about the file, such as a password. The only parameter that's necessary is the FileName. You can remove the other parameters from the command, and Word will use its default values for them.

A more important problem is that every time you run the recorded macro, it will open the same document. This may be what you intend, but more likely you want the macro to let you choose which document to open.

One way to get the file's name into the macro is to display an input box, where you can type it in. The InputBox function shows a message box with a text entry field, and its result is the name that you type into the field.

```
Sub Macro2A()
  Dim MyFileName As String
  MyFileName = InputBox( "Enter file name to open:" , _
     "Open a Document" )
  If MyFileName <> "" Then
    Documents.Open FileName:=MyFileName
    ' more code, for example ...
    ActiveWindow.View = wdPrintView
  End If
End Sub
```

But this isn't very friendly. There's a good chance of making a typing mistake. And if the document isn't in the currently active folder, you have to type the entire path to it. A better idea is to use the File > Open dialog that's already built into Word, which lets you browse to and select the proper document. When you click the OK button in the dialog, Word opens the selected document. The macro is simpler, too, because it doesn't need a separate Documents.Open statement&8212;the dialog handles it all for you.

```
Sub Macro2B()
  If Dialogs(wdDialogFileOpen).Show = - 1 Then
    ' more code, for example ...
    ActiveWindow.View = wdPrintView
  End If
End Sub
```

The word "Dialogs" in this code refers to a list of all of Word's built-in dialogs. Each dialog has a name that starts with "wdDialog". In this case, wdDialogFileOpen is the name of the built-in File > Open dialog, and the expression "Dialogs(wdDialogFileOpen)" selects that particular dialog from the list. To see all of the possible names, press F2 in the VBA editor to display the Object Browser, type wdDialog into the search box, and press Enter.

The word ".Show" refers to a method of the dialog. A method is an action that can be done&8212;the .Show

method causes the dialog to appear and execute (carry out its function).

Many methods also have a value after they execute, which tells the macro something about what just happened (this is called "returning" the value). In this case, if you click the OK button in the dialog then the .Show method returns the value &8211;1, but if you click the Cancel button or the X in the title bar then .Show returns the value 0. The VBA help topic for each method tells you what values that method can return and what they mean. You can use the returned value in an If statement, as in Macro2B, to decide what to do.

Similar changes to recorded code let you make macros that save files to variable locations, search for variable strings, and many other unrecordable variations. You can find out more at Getting help with calling Word's built-in dialogs using VBA.

==========

```
Public Sub PaulKeyBindings()

Dim kb As KeyBinding
Dim s As String
Dim tbl As Table

' Check active document for text
' and warn user.
If ActiveDocument.Content <> vbCr Then
   If MsgBox("Active doc has content. Proceed?", vbQuestion + vbYesNo) = vbNo Then Exit Sub
End If

' Print Heading
Selection.InsertAfter KeyBindings.Count & _
   " key bindings in context: " & CustomizationContext & vbCr & vbCr

' Collapse selection to end of document
Selection.Collapse wdCollapseEnd

' Insert start of table bookmark
ActiveDocument.Bookmarks.Add "StartOfTable"

' Print table heading
Selection.InsertAfter "KeyString" & vbTab & _
   "KeyCategory" & vbTab & "Command" & vbTab _
   & "KeyCode" & vbTab & "KeyCode2" _
```

```
  & vbTab & "CommandParameter" & vbCr

'Start the For loop, printing key binding data
Selection.Collapse wdCollapseEnd
For Each kb In KeyBindings
   s = kb.KeyString & vbTab & kb.KeyCategory _
      & vbTab & kb.Command & vbTab & kb.KeyCode _
      & vbTab & kb.KeyCode2 & vbTab _
      & kb.CommandParameter & vbCr
   Selection.InsertAfter s
   Selection.Collapse wdCollapseEnd
Next kb

' Collapse selection to end of document
Selection.Collapse wdCollapseEnd

' Insert end of table bookmark
ActiveDocument.Bookmarks.Add "EndOfTable"

' Select text between bookmarks
ActiveDocument.Bookmarks("StartofTable").Select
With Selection
    .ExtendMode = True
.GoTo wdGoToBookmark, , , "EndOfTable"
    .ExtendMode = False
End With

Set tbl = Selection.ConvertToTable(Separator:=wdSeparateByTabs)
tbl.Columns.AutoFit
Selection.Collapse wdCollapseEnd

End Sub



==========================================
Sub ListAllWords()
   Dim rng As Range, myWord As String

   Selection.EndKey Unit:=wdStory
   Selection.TypeText Text:="==========" & vbCrLf
   Selection.WholeStory
   Selection.End = Selection.End - 12
   Selection.Range.HighlightColorIndex = wdYellow
   Selection.EndKey Unit:=wdStory

   gogo = True
   Do While gogo = True
      Set rng = ActiveDocument.Range
      With rng.Find
         .ClearFormatting
         .MatchWildcards = True
         .Highlight = True
         .Replacement.Highlight = False
         .Text = "<*>"
         .Execute Replace:=wdReplaceOne
      End With
```

```
    gogo = rng.Find.Found
    If gogo = True Then
      myWord = rng
      Set rng = ActiveDocument.Range
      With rng.Find
        .ClearFormatting
        .Replacement.Highlight = False
        .MatchCase = False
        .Text = "<" & myWord & ">"
        .Execute Replace:=wdReplaceAll
      End With
      myWord = StrConv(myWord, vbLowerCase)
      Selection.TypeText Text:=myWord & vbCrLf
    End If
  Loop

    Selection.WholeStory
    Selection.Range.HighlightColorIndex = wdNoHighlight

  Selection.HomeKey Unit:=wdStory
  With Selection.Find
    .ClearFormatting
    .Text = "=====^p"
    .MatchWildcards = False
    .Execute
  End With

  listStart = Selection.End
  Selection.EndKey Unit:=wdStory
  Selection.Start = listStart
  Selection.Sort
  Selection.End = Selection.Start

End Sub
```

==========

OK, the following macro loads all the doc, docx and rtf files in a specified-in-the-macro folder and counts the total number of words. It doesn't use ActiveDocument.Words.Count because that seems to treat punctuation as words and so gives an inflated count (and we wouldn't want that, now, would we?!)

It has to do the count by using a wildcard search to find all of the (words consisting of letters, hyphens and apostrophes) and then follows it with a count of all (words consisting of numbers and commas), e.g. "15,000" is one single word, as are "can't" and "half-hearted".

```
Sub MultiFileCount()
  Dim fs, foc, fc, fi, fiType
  Dim foldername As String
  Dim myDoc As Document
```

```
  foldername = "c:\Full\Address\Of\Folder"

  Set fs = CreateObject("Scripting.FileSystemObject")
  Set foc = fs.GetFolder(foldername)
  Set fc = foc.Files

  WordsTotal = 0
  FilesTotal = 0
  For Each fi In fc
    fiType = Right(fi, 4)
    If fiType = ".doc" Or fiType = "docx" Or fiType = ".rtf" Then
      Set myDoc = Application.Documents.Open(filename:=fi.Path, ReadOnly:=True)

      With Selection.Find
        .ClearFormatting
        .Replacement.ClearFormatting
        .MatchWildcards = True
        .Text = "[/a-zA-Z\'\-’]@[!/a-zA-Z\'\-\’]"
        .Replacement.Text = " ^& ^0124 "
        .Replacement.Highlight = False
        .Wrap = wdFindContinue
        .Format = True
      End With

      Selection.HomeKey Unit:=wdStory
      Do While Selection.Find.Execute
        WordsTotal = WordsTotal + 1
      Loop

      With Selection.Find
        .ClearFormatting
        .Replacement.ClearFormatting
        .MatchWildcards = True
        .Text = "[!a-zA-Z][0-9,]@[!0-9,]"
        .Replacement.Text = " ^& ^0124 "
        .Replacement.Highlight = False
        .Wrap = wdFindContinue
        .Format = True
      End With
      Selection.HomeKey Unit:=wdStory
      Do While Selection.Find.Execute
        WordsTotal = WordsTotal + 1
      Loop

      myDoc.Close
      FilesTotal = FilesTotal + 1
    End If
  Next
  MsgBox ("Total words in" + Str(FilesTotal) + " files: " + Str(WordsTotal))

End Sub



==========================================
'Sub MultiFileFandR()
  Dim fs, foc, fc, fi, fiType, fiList
```

```
  Dim foldername As String
  Dim myDoc As Document

  foldername = "c:\Program Files\VirtualAcorn\VirtualRPC-SA\HardDisc4\MyFiles2\WIP"

  Set fs = CreateObject("Scripting.FileSystemObject")
  Set foc = fs.GetFolder(foldername)
  Set fc = foc.Files

  filesTotal = 0
  fiList = ""
  For Each fi In fc
    fiType = Right(fi, 4)
    If fiType = ".doc" Or fiType = "docx" Or fiType = ".rtf" Then
    fiList = fiList + vbCrLf + fi
    End If
  Next
  fiList = fiList + vbCrLf + vbCrLf + "FandR on all " + str(filesTotal) + " files?"
  MsgBox (fiList, vbYesNo) = vbNo Then Exit Sub


  For Each fi In fc
    'Do what you need to with each file
    'Each fi is a 'File object' (look it up in VBA) and you can
    'access a lot of properties on it.  You can filter out thoseOR fitype = ".wbk" OR
    'files you don't need to get data from.
    fitype = Right(fi, 4)
    If fitype = ".doc" Or fitype = "docx" Or fitype = ".rtf" Then
      Set myDoc = Application.Documents.Open(filename:=fi.Path, ReadOnly:=True)
      ' WordTotal = WordTotal + ActiveDocument.Words.Count
      ' FilesTotal = FilesTotal + 1

      With Selection.Find
        .ClearFormatting
        .Replacement.ClearFormatting
        .Text = "with"
        .Replacement.Text = "^&"
        .Wrap = wdFindContinue
        .Replacement.Highlight = true
        .MatchWildcards = True
        .Format = True
      End With
      Selection.Find.Execute Replace:=wdReplaceAll

      myDoc.Save
      myDoc.Close
    End If
  Next
  MsgBox ("Total words in" + Str(FilesTotal) + " files: " + Str(WordTotal))


==========================================
Sub PreEdit()

'(c)2009 Paul Beverley

' This version is dated 05.05.09.
```

```
' Thanks to Anna Sharman, Matthew Strawbridge and Phillip Marsden
'  for programming segments for some features.

' This version, April 2009, is for use by SfEP MEMBERS ONLY.
' Please do not pass it on to non-members.

' Show it to non-members, by all means, but they may not use it
'  unless and until they join SfEP.

' These are the default values. Change them if you prefer others.

Highlight = 3
Ignore = 0
MyColour = wdTurquoise

PreEditList = "C:\Lists\PreEdit.txt"

On Error GoTo ReportIt

Dim findarray(500) As String
Dim replacearray(500) As String
Dim colourarray(500) As WdColorIndex


' If you're going to have lists with more than 500 items then
'  increase these three lines above.

OldText = Selection.Find.Text
OldReplaceText = Selection.Find.Replacement.Text
OldMC = Selection.Find.MatchCase
OldMW = Selection.Find.MatchWildcards
OldHL = Selection.Find.Highlight

Dim findstring As String
Dim replacestring As String
Dim ReadData As String
Dim DataLine As String
Dim Title As String
Dim MyText As String

Dim ColourNow As WdColorIndex
Dim OldColour As WdColorIndex
OldColour = Options.DefaultHighlightColorIndex

Dim FS As FileSystemObject
Set FS = New FileSystemObject

Dim InputFile As TextStream
Set InputFile = FS.OpenTextFile(PreEditList, ForReading)

Do
  DataLine = InputFile.ReadLine()
  Title = DataLine
  DataLine = Replace(Title, " ", "")
  DataLine = Right(DataLine, 7)
  If DataLine = "light=0" Then Highlight = 0: ColourNow = 0
  If DataLine = "light=1" Then Highlight = 1: ColourNow = OldColour
```

```
  If DataLine = "light=2" Then Highlight = 2: ColourNow = MyColour
  If DataLine = "light=3" Then Highlight = 3: ColourNow = OldColour
  If DataLine = "gnore=1" Then Ignore = 1
Loop Until Left(Title, 1) = "#" Or InputFile.AtEndOfStream
If Title = "" Then Title = "NoSections"

If InputFile.AtEndOfStream Then
  InputFile.Close
  Set InputFile = FS.OpenTextFile(PreEditList, ForReading)
Else
   Do
     ReadData = InputFile.ReadLine()
   Loop Until (ReadData = Title) Or InputFile.AtEndOfStream
   If InputFile.AtEndOfStream Then
     InputFile.Close
     Set InputFile = FS.OpenTextFile(PreEditList, ForReading)
   End If
End If

idx = 0
While (Not InputFile.AtEndOfStream) And Title > ""
  ReadData = InputFile.ReadLine()
    If Left(ReadData, 1) = "@" Then
      ReadData = Right(ReadData, (Len(ReadData) - 1))
      highlighting = 0
    Else
      highlighting = 1
    End If
    If (ReadData > "") And (Not Left(ReadData, 1) = "|") And (Not Left(ReadData, 1) = "#") Then
    colonpos = InStr(ReadData, "|")
    findstring = Left(ReadData, colonpos - 1)
    replacestring = Right(ReadData, (Len(ReadData) - colonpos))
    findarray(idx) = findstring
    replacearray(idx) = replacestring
    colourarray(idx) = ColourNow * highlighting
    idx = idx + 1
  Else
    If Left(ReadData, 1) = "#" Then Title = ""
    If Left(ReadData, 1) = "|" Then
      DataLine = Replace(ReadData, " ", "")
      If Highlight = 3 And InStr(DataLine, "olour=") > 0 Then ColourNow = Val(Mid(DataLine, 9))
    End If
  End If
Wend

idxmax = idx - 1
InputFile.Close

Offset = 7

If idxmax >= 0 Then
 For idx = 0 To idxmax
     Selection.HomeKey Unit:=wdStory
   findstring = findarray(idx)
   replacestring = replacearray(idx)
   With Selection.Find
     If Ignore = 1 Then .Highlight = False
```

```
    If Left(findstring, 1) = "~" Then
      findstring = Right(findstring, (Len(findstring) - 1))
      .MatchWildcards = True
    Else
      .MatchWildcards = False
    End If
    If Left(findstring, 1) = "¬" Then
      findstring = Right(findstring, (Len(findstring) - 1))
      .MatchCase = False
    Else
      .MatchCase = True
    End If
    .Text = findstring
    If (Highlight > 0 And replacestring > "") Then .Replacement.Font.Color = (colourarray(idx) Mod 10) + Offset
    .Forward = True
    .Wrap = wdFindContinue
    .Replacement.Text = replacestring
  End With
    Counter = 0
    Do While Selection.Find.Execute
      Counter = Counter + 1
    Loop
    Selection.HomeKey Unit:=wdStory
    Selection.Find.Execute Replace:=wdReplaceAll
    If colourarray(idx) > 9 Then
      MyText = findstring + vbTab + Str$(Counter) + vbCrLf
      Selection.EndKey Unit:=wdStory
      Selection.TypeText Text:=MyText
    End If
 Next
End If

If Highlight = 0 Then Options.DefaultHighlightColorIndex = 0
If Highlight = 1 Then Options.DefaultHighlightColorIndex = ColourNow

For idx = 2 To 7
  Options.DefaultHighlightColorIndex = idx
  Selection.HomeKey Unit:=wdStory

  With Selection.Find
    .ClearFormatting
    .Font.Color = idx + Offset
    .Replacement.ClearFormatting
    .Replacement.Font.Color = wdColorAutomatic
    .Replacement.Highlight = True
    .Text = ""
    .Replacement.Text = ""
    .Forward = True
    .Wrap = wdFindContinue
  End With
  Selection.Find.Execute Replace:=wdReplaceAll
Next

With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "^^^^(?)"
```

```
  .Replacement.Text = "\1"
  .Replacement.Font.Superscript = True
  .Wrap = wdFindContinue
  .MatchWildcards = True
  .Format = True
End With
' Selection.Find.Execute Replace:=wdReplaceAll

With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "__(?)"
  .Replacement.Text = "\1"
  .Replacement.Font.Subscript = True
  .Wrap = wdFindContinue
  .MatchWildcards = True
  .Format = True
End With
' Selection.Find.Execute Replace:=wdReplaceAll

With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "\<\<(*)\>\>"
  .Replacement.Text = "\1"
  .Replacement.Font.Italic = True
  .Wrap = wdFindContinue
  .MatchWildcards = True
  .Format = True
End With
' Selection.Find.Execute Replace:=wdReplaceAll

With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "\[\[(*)\]\]"
  .Replacement.Text = "\1"
  .Replacement.Font.Bold = True
  .Wrap = wdFindContinue
  .MatchWildcards = True
  .Format = True
End With
' Selection.Find.Execute Replace:=wdReplaceAll

' Dummy F&R to restore F&R window

 With Selection.Find
   .ClearFormatting
   .Replacement.ClearFormatting
   .Text = OldText
   .Replacement.Text = OldReplaceText
   .Highlight = OldHL
   .MatchCase = OldMC
   .MatchWildcards = OldMW
 End With
```

```
Options.DefaultHighlightColorIndex = OldColour


FinishHere:
Exit Sub

ReportIt:
MySearch = vbCrLf + vbCrLf + "Find: " + findstring + vbCrLf + vbCrLf + "Replace: " + replacestring
Select Case Err.Number
  Case 5
    MyError = "Missing '|' in line:" + vbCrLf + vbCrLf + ReadData
  Case 53, 76
    MyError = "Can't find a pre-edit file at: " + vbCrLf + vbCrLf + PreEditList
  Case 5692
    MyError = "You can't use ^p in a wildcard search:" + vbCrLf + vbCrLf + findstring + "|" + replacestring
  Case Else
    MyError = Err.Description + MySearch
End Select

MsgBox (MyError)
Resume FinishHere


End Sub
```

==========

SetRange Method
See Also Applies To Example Specifics
Sets the starting and ending character positions for the range or selection.

Note Character position values start at the beginning of the story, with the first value being 0 (zero). All characters are counted, including nonprinting characters. Hidden characters are counted even if they're not displayed.

expression.SetRange(Start, End)

expression Required. An expression that returns a Range or Selection object.

Start Required Long. The starting character position of the range or selection.

End Required Long. The ending character position of the range or selection.

Remarks
The SetRange method redefines the starting and ending positions of an existing Selection or Range object. This method differs from the Range method, which is used to create a range, given a starting and ending position.

Example
This example selects the first 10 characters in the document.

```
Selection.SetRange Start:=0, End:=10
```

This example uses SetRange to redefine myRange to refer to the first three paragraphs in the active document.

```
Set myRange = ActiveDocument.Paragraphs(1).Range
myRange.SetRange Start:=myRange.Start, _
   End:=ActiveDocument.Paragraphs(3).Range.End
```

This example uses SetRange to redefine myRange to refer to the area starting at the beginning of the document and ending at the end of the current selection.

```
Set myRange = ActiveDocument.Range(Start:=0, End:=0)
myRange.InsertAfter "Hello "
myRange.SetRange Start:=myRange.Start, End:=Selection.End
```

This example extends the selection to the end of the document.

```
Selection.SetRange Start:=Selection.Start, _
   End:=ActiveDocument.Content.End
```

==========

Xtreme Visual Basic Talk > Legacy Visual Basic (VB 4/5/6) > VBA / Office Integration > Word, PowerPoint, Outlook, and Other Office Products > Help! How to search an MS Word textbox with a VBA macro?

PDA

View Full Version : Help! How to search an MS Word textbox with a VBA macro?

Peter Ring

09-16-2004, 09:25 AM

I'm currently writing a macro to search Word documents for certain fields.

If I do the search manually, there is no problem with searching the Word textboxes.

But when I record this procedure and run it again, the (floating, anchored) textboxes are not searched. I then have to select the textboxes one-by-one and run the macro there. I guess the problem is the object "Selection".

Simple example:

```
Selection.Find.ClearFormatting

With Selection.Find
.Text = "^d" 'Find Fields
.Replacement.Text = ""
.Forward = True
.Wrap = wdFindContinue
.Format = False
.MatchCase = False
.MatchWholeWord = False
.MatchWildcards = False
```

.MatchSoundsLike = False
.MatchAllWordForms = False
End With
Selection.Find.Execute
MsgBox Selection.Text
Selection.Find.Execute
MsgBox Selection.Text

:confused:

I'm using Word XP, but the macro should preferably work
with all MS Word versions using VBA.

Does anybody know a method to search the whole
document including the Word textboxes?

---

herilane

09-16-2004, 10:54 AM

I think you may need to run Find twice. Once for the main text, and once for textboxes. That's the only way I can think of. :(
'For the main text:
With Activedocument.Range.Find
'For the textboxes:
ActiveDocument.Shapes.SelectAll
With Selection.FindI'd be happy to hear of a better way, if anybody knows!

Peter Ring

09-20-2004, 04:11 AM

Many thanks to herilane for a good answer, which lead me
on the track of the solution.

However, it was not that simple. The solution was:

For j = 1 To ActiveDocument.Shapes.Count
ActiveDocument.Shapes(j).Select
Selection.WholeStory
GoSub DoSearch
Next j

where DoSearch is a subroutine containing the general
search procedure and handling the results.

Powered by: vBulletin v3.6.4

==========

<Word.Tips.Net> Welcome toWord.Tips.Net

[

RSS Feeds

<RSS 2.0> Daily Nuggets (a WordTip every day)

<RSS 2.0> Allen Wyatt's WordTips (the weekly classic)

Tips.Net > WordTips Home > Formatting > Styles > Printing a Full Style Sheet

Printing a Full Style Sheet

Summary: Word supports the use of styles (they are very powerful), but it doesn't provide a way to get a full-featured style sheet printed. This tip examines ways you can create your own style sheets for printing. (This tip works with Microsoft Word 97, Word 2000, Word 2002, Word 2003, and Word 2007.)

Dave would love a way to print a full-featured style sheet for his documents. He knows that he can choose to print "Styles" in the Print dialog box, but he would rather have a style sheet that shows the actual styles, such as color, size, font, etc.

Unfortunately there is no such capability in Word. You can, however, create a style sheet of your liking by using a macro. For instance, the following will insert, in the current document, the names of all the styles available in the document. Each style name is on its own line (paragraph) and is formatting using the various styles.

```
Sub ListStyleNames()
  For Each Style In ActiveDocument.Styles
    With Selection
      .Style = ActiveDocument.Styles(Style)
      .TypeText (ActiveDocument.Styles(Style).NameLocal)
      .TypeParagraph
    End With
  Next
End Sub
```

Such an approach, while handy for a concise list of styles, isn't much more informative than what can be printed using the "Styles" designation in the Print dialog box. It does, however, provide a basis upon which one can build to create a more full-featured style sheet.

The problem with creating a detailed style sheet using macros is that styles can contain a ton of information. The object model used by Word (and accessible in VBA) quickly becomes quite complex when testing styles to see what they contain. Here's just a simple example to give you the

flavor:

```
Sub SimpleStyleSheet()
   Dim sOutput As String
   Dim sTemp As String
   Dim StyleTypes(4) As String

   StyleTypes(1) = "Paragraph"
   StyleTypes(2) = "Character"
   StyleTypes(3) = "Table"
   StyleTypes(4) = "List"

   For Each Style In ActiveDocument.Styles
      sOutput = Style.NameLocal & vbCrLf
      sOutput = sOutput & "   Style type: " & StyleTypes(Style.Type) & vbCrLf
      sTemp = Style.BaseStyle
      If Len(sTemp) > 0 Then
         sOutput = sOutput & "   Based on: " & Style.BaseStyle & vbCrLf
      End If
      sOutput = sOutput & "   Font: " & Style.Font.Name
      sTemp = ""
      If Style.Font.Bold Then sTemp = sTemp & "Bold, "
      If Style.Font.Italic Then sTemp = sTemp & "Italic, "
      If Len(sTemp) > 0 Then
         sTemp = Left(sTemp, Len(sTemp) - 2)
         sOutput = sOutput & " (" & sTemp & ")"
      End If
      sOutput = sOutput & vbCrLf
      Selection.TypeText (sOutput & vbCrLf)
   Next
End Sub
```

The only thing this macro does is to list all the styles, what type of styles they are, whether they are based on a different style (and if so, what that style is named), what font is used by the style, and whether the font is bold or italic. Anyone familiar with styles will immediately understand that these few items are only a small sampling of what can be defined within a style. To check all possible style formats and print them in the style sheet would make the macro very long, indeed.

Even so, this macro might be useful as it provides an idea of how to put together your own style sheet. You just need to figure out what you want to see in the style sheet and then add the macro code to determine that information and print it out.

Tip #6748 applies to Microsoft Word versions: 97 | 2000 | 2002 | 2003 | 2007

```
==========================================
Attribute VB_Name = "Word2TWiki"

Sub Word2TWiki()
'Attribute VB_Name = "Word2Wiki"
'This function can be used to convert a Word doc to the TWiki formatting language
'For references: http://www.twiki.org/cgi-bin/view/Plugins/MsWordToTWikiMLAddOn

    Application.ScreenUpdating = False

    CleanFormattingParagraphEndings

    ConvertHeading "---+", wdStyleHeading1
    ConvertHeading "---++", wdStyleHeading2
    ConvertHeading "---+++", wdStyleHeading3
    ConvertHeading "---++++", wdStyleHeading4
    ConvertHeading "---+++++", wdStyleHeading5
    ConvertHeading "---++++++", wdStyleHeading6

    ConvertStyle "__", bold:=True, italic:=True
    ConvertStyle "*<u>", twikiCode2:="</u>*", bold:=True, underline:=wdUnderlineSingle
    ConvertStyle "==", bold:=True, fontName:="Courier New"

    ConvertStyle "*", bold:=True
    ConvertStyle "_", italic:=True
    ConvertStyle "<u>", twikiCode2:="</u>", underline:=wdUnderlineSingle
    ConvertStyle "=", fontName:="Courier New"

    ConvertLists
    ConvertHyperlinks

    ConvertTables

    WikiSaveAsHTMLAndConvertImages

    ' Copy to clipboard
    ActiveDocument.Content.Copy

    Application.ScreenUpdating = True
End Sub

Private Sub ConvertTables()
'This function was kindly provided by Merlijn van Deen <valhallasw{at}platypusnet{dot}org>
'on September 17, 2005
'MS Excel is used in order to handle merged cells

Dim excelapp, sheet As Object
Dim thisRow As Row
Dim thisCell As Cell
```

```
Dim myRange As Range

For Each thisTable In ActiveDocument.Tables
   thisTable.Select

   'Breaks don't just break TWiki tables, but also the splitting routine
   'Find and remove all breaks, they break split and/or TWiki
   With Selection.Find
      .ClearFormatting
      .Replacement.ClearFormatting
      .Wrap = wdFindStop
      .MatchCase = False
      .MatchWholeWord = True
      .MatchWildcards = False
      .MatchSoundsLike = False
      .MatchAllWordForms = False
      .Replacement.Text = " "
      .Replacement.Font.bold = False
      .Replacement.Font.italic = False

      .Execute FindText:="^n", Replace:=wdReplaceAll 'column
      .Execute FindText:="^m", Replace:=wdReplaceAll 'page
      .Execute FindText:="^b", Replace:=wdReplaceAll 'section
   End With
   'Added by Jos Maccabiani on Sep 18, 2005:
   'To preserve line breaks in the table, treat paragraph and line breaks in a
   'special way: replace with unformatted %BR%
   With Selection.Find
      .ClearFormatting
      .Replacement.ClearFormatting
      .Wrap = wdFindStop
      .MatchCase = False
      .MatchWholeWord = True
      .MatchWildcards = False
      .MatchSoundsLike = False
      .MatchAllWordForms = False
      .Replacement.Text = " %BR% "
      .Replacement.Font.bold = False
      .Replacement.Font.italic = False

      .Execute FindText:="^p", Replace:=wdReplaceAll 'paragraph
      .Execute FindText:="^l", Replace:=wdReplaceAll 'line
   End With

   thisTable.Select
   Selection.Copy

   'use excel to fix merged cells
   Set excelapp = CreateObject("Excel.Application")
      excelapp.Workbooks.Add
      Set sheet = excelapp.Worksheets.Add
         sheet.Paste

         'disable all borders, necessary to prevent extra spaces between
         ' | | in merged cells
         excelapp.Cells.Borders.LineStyle = wdNone
```

```
        For Each Cell In excelapp.Selection.Cells
           'First check if the cell is empty (or contains of spaces)
           'If so, change contents to ' '
           'This is to prevent cells from being merged in twiki

           If Len(Cell.FormulaR1C1) = 0 Then Cell.FormulaR1C1 = " "
        Next

        For Each Cell In excelapp.Selection.Cells
           'Now unmerge-and-change all cells

           Set c = Cell.Mergearea 'Cells have to be unmerged first,  but the area is needed later

           'nul is an output variable; VBA syntax checking needed  one, even though
           'Split is listed as a Method for Cell.
           'This sub splits the cells in word to their original state  (with information from Excel)
           '! a SUB will give no output normally, however, the VBA  syntax checker doesn't recognise the use of
Split as a sub
           '! The syntax checker will complain about adding nul =  _and_ about removing it when thisTable is
defined
           '! The program works without defining thisTable, so just  keep the nul = (or find a way to fix it)
           nul = thisTable.Cell(Cell.Row, Cell.Column).Split(c.Rows.Count, c.Columns.Count)

           c.UnMerge

           If c.Rows.Count > 1 Then 'rows!
              For x = 2 To c.Rows.Count
                 c.Cells(x, 1) = "^"
              Next x
           End If
        Next

        excelapp.Selection.Copy

        Set myRange = thisTable.Cell(1, 1).Range
        myRange.End = thisTable.Cell(thisTable.Rows.Count, thisTable.Columns.Count).Range.End
        myRange.Select

        'fix it, the dirty way
        Selection.Paste 'replace the table with the excel data


        'cleaning up
        Set sheet = Nothing
        excelapp.DisplayAlerts = False 'To prevent 'Do you want to  save (...)' dialog of excel
        excelapp.Quit
        Set excelapp = Nothing

        'End with the original procedure
        For Each thisRow In thisTable.Rows
           thisRow.Range.InsertBefore "|"
           thisRow.Range.InsertAfter "|"
        Next thisRow
        thisTable.ConvertToText Separator:="|"

Next thisTable
End Sub
```

```
Private Sub CleanFormattingParagraphEndings()
    With Selection.Find
        .ClearFormatting
        'Target
        .Text = "^p"
        'Replacement
        .Replacement.ClearFormatting
        .Replacement.Font.bold = False
        .Replacement.Font.italic = False
        .Replacement.Font.underline = wdUnderlineNone
        .Replacement.Font.Name = "Arial"
        .Replacement.Text = "^p"
        'Options
        .Wrap = wdFindContinue
        .Format = True
        .MatchCase = False
        .MatchWholeWord = True
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With
    Selection.Find.Execute Replace:=wdReplaceAll
End Sub

Private Sub ConvertHeading(twikiCode As String, heading As WdBuiltinStyle)
    Dim normalStyle As style
    Set normalStyle = ActiveDocument.Styles(wdStyleNormal)

    ActiveDocument.Select

    With Selection.Find

        .ClearFormatting
        .style = ActiveDocument.Styles(heading)
        .Text = ""

        .Format = True
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False

        .Forward = True
        .Wrap = wdFindContinue

        Do While .Execute
            With Selection
                If InStr(1, .Text, vbCr) Then
                    ' Just process the chunk before any newline characters
                    ' We'll pick-up the rest with the next search
                    .Collapse
                    .MoveEndUntil vbCr
                End If

                ' Don't bother to markup newline characters (prevents a loop, as well)
```

```
            If Not .Text = vbCr Then
               .InsertBefore twikiCode + " "
            End If

            .style = normalStyle
         End With
      Loop
   End With
End Sub

Private Sub ConvertStyle(twikiCode1 As String, Optional twikiCode2 As String, Optional bold As Boolean =
False, Optional italic As Boolean = False, Optional underline As WdUnderline = wdUndefined, Optional fontName
As String = "")
'This function converts styled text in Word to TWiki markup
'
'This function also solves the problem that if a word is in a style,
'and the trailing space is also in that style, then
'a space will be placed before the trailing wikiCode causing the
'effect to be ignored when first posted to TWiki.
'
'This is what this function does:
'--------------------------------
'Insert new tags
'Remove all 'loose' formatted spaces
'Remove leading spaces
'Remove trailing spaces
'Add missing spaces before
'Add missing spaces after
'Remove the inserted tags and replace by TWiki tags

   With Selection.Find
      .ClearFormatting
      .Replacement.ClearFormatting
      .MatchSoundsLike = False
      .MatchAllWordForms = False
      .Forward = True
      .Wrap = wdFindContinue
   End With

'Insert new tags
'   * find: (empty) Formatted:Style
'   * repl: <TWikiStyle>^&</TWikiStyle> Formatted:NotStyle
   With Selection.Find
      .Font.bold = bold
      .Font.italic = italic
      .Font.underline = underline
      .Font.Name = fontName
      .Text = ""
      .Replacement.Text = "<TWikiStyle>^&</TWikiStyle>"
      .Replacement.Font.bold = False
      .Replacement.Font.italic = False
      .Replacement.Font.underline = wdUnderlineNone
      .Format = True
      .MatchCase = False
      .MatchWholeWord = False
      .MatchWildcards = False
   End With
```

```
    Selection.Find.Execute Replace:=wdReplaceAll

'Remove all 'loose' formatted spaces
'   * find: <TWikiStyle> </TWikiStyle>
'   * repl: (empty) Formatting:None
    With Selection.Find
        .Text = "<TWikiStyle> </TWikiStyle>"
        .Replacement.Text = ""
        .Format = False
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

'Remove leading spaces
'   * find: \<TWikiStyle\>( @)<  (with wildcards)
'   * repl: <TWikiStyle>
    With Selection.Find
        .Text = "\<TWikiStyle\>( @)<"
        .Replacement.Text = "<TWikiStyle>"
        .Format = False
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = True
    End With
    Selection.Find.Execute Replace:=wdReplaceAll
'Remove trailing spaces
'   * find: (>)( @)(\</TWikiStyle\>)  (with wildcards)
'   * repl: </TWikiStyle>
    With Selection.Find
        .Text = "(>)( @)(\</TWikiStyle\>)"
        .Replacement.Text = "</TWikiStyle>"
        .Format = False
        .MatchCase = True
        .MatchWholeWord = True
        .MatchWildcards = True
    End With
    Selection.Find.Execute Replace:=wdReplaceAll
'Add missing spaces before
'   * find: (>)\<TWikiStyle\>  (with wildcards)
'   * repl: \1 <TWikiStyle>
    With Selection.Find
        .Text = "(>)\<TWikiStyle\>"
        .Replacement.Text = "\1 <TWikiStyle>"
        .Format = False
        .MatchCase = True
        .MatchWholeWord = True
        .MatchWildcards = True
    End With
    Selection.Find.Execute Replace:=wdReplaceAll
'Add missing spaces after
'   * find: \</TWikiStyle\>(<)  (with wildcards)
'   * repl: </TWikiStyle> \1
    With Selection.Find
        .Text = "\</TWikiStyle\>(<)"
        .Replacement.Text = "</TWikiStyle> \1"
```

```
        .Format = False
        .MatchCase = True
        .MatchWholeWord = True
        .MatchWildcards = True
    End With
    Selection.Find.Execute Replace:=wdReplaceAll
'Remove the inserted tags and replace by TWiki tags
'   * find: \<TWikiStyle\>(*)\</TWikiStyle\>   (with wildcards)
'   * repl: twikiCode\1twikiCode
    With Selection.Find
        .Text = "\<TWikiStyle\>(*)\</TWikiStyle\>"
        .Replacement.Text = twikiCode1 + "\1" + IIf(twikiCode2 = "", twikiCode1, twikiCode2)
        .Format = True
        .MatchCase = True
        .MatchWholeWord = True
        .MatchWildcards = True
    End With
    Selection.Find.Execute Replace:=wdReplaceAll

End Sub

Private Sub ConvertHyperlinks()
    Dim hyperCount As Integer

    hyperCount = ActiveDocument.Hyperlinks.Count

    For i = 1 To hyperCount
        With ActiveDocument.Hyperlinks(1)
            Dim addr As String
            addr = .Address
            text2disp = .TextToDisplay
            .Delete
            .Range.InsertBefore "[[" & addr & "]["
            .Range.InsertAfter "]]"
            '.Range.InsertBefore "[["
            '.Range.InsertAfter "|" & addr & "]]"
        End With
    Next i
End Sub

Private Sub ConvertLists()
    Dim para As Paragraph
    For Each para In ActiveDocument.ListParagraphs
        With para.Range
            If .ListFormat.ListType = wdListBullet Then
                .InsertBefore "   * "
            Else
                .InsertBefore "   1. "
            End If

            For x = 2 To .ListFormat.ListLevelNumber
                .InsertBefore "   "
            Next x

            .ListFormat.RemoveNumbers
        End With
    Next para
```

```
End Sub

Private Sub WikiSaveAsHTMLAndConvertImages()
   Dim s As Shape
   For Each s In ActiveDocument.Shapes
      If s.Type = msoPicture Then
         s.ConvertToInlineShape
      End If
   Next


   FileName = ActiveDocument.Path + "\" + ActiveDocument.Name
   FolderName = FileName + "_files"

   ActiveDocument.SaveAs FileName:=FileName + ".htm", _
            FileFormat:=wdFormatFilteredHTML, LockComments:=False, Password:="", _
            AddToRecentFiles:=True, WritePassword:="", ReadOnlyRecommended:=False, _
            EmbedTrueTypeFonts:=False, SaveNativePictureFormat:=False, SaveFormsData _
            :=False, SaveAsAOCELetter:=False

   Set fs = CreateObject("Scripting.FileSystemObject")
   If fs.FolderExists(FolderName) Then
      Set f = fs.GetFolder(FolderName)

      Dim iShape As InlineShape
      Dim sA As String, sB As String, sC As String, sD As String

      sA = "<img src="
      sB = "%ATTACHURL%/"
      sC = "></img>"
      'sD = sA & sB & """" & "myimage.jpg" & """" & sC
      Set fc = f.Files
      i = 1
      For Each f In fc
         If i <= ActiveDocument.InlineShapes.Count Then
            Set iShape = ActiveDocument.InlineShapes.Item(i)
            iShape.Range.InsertBefore sA & """" & sB & f.Name & """" & sC
            i = i + 1
         End If
      Next

      Shell "explorer.exe " + FileName + "_files", vbNormalFocus
   End If
End Sub
```

==========

```
Sub WikiSwitch()
  Set rng = ActiveDocument.Range
  With rng.Find
    .Text = "=" & "=^p"
    .Execute
  End With

  If rng.Find.Found = True Then
    Call WikiToStyles
  Else
    Call WikiToText
  End If

  Selection.HomeKey Unit:=wdStory

End Sub
Sub WikiToText()
Dim Eq2, Eq3, Tk2, Tk3, pr, npr As String
Eq2 = "=" & "=": Eq3 = Eq2 + "="
Tk2 = """ & """: Tk3 = Tk2 + """
pr = "pre": npr = "</" & pr & ">"
pr = "<" & pr & ">"

'Convert Heading 2 to <=><=><=>Title<=><=><=>
  Set rng = ActiveDocument.Range
  With rng.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .MatchWildcards = True
    .Text = "(*)^13"
    .Style = "Heading 2"
    .Replacement.Text = Eq3 & "\1" & Eq3 & "^p"
    .Replacement.Style = "Normal"
    .Execute Replace:=wdReplaceAll
  End With

'Convert Heading 1 to <=><=>Title<=><=>
  Set rng = ActiveDocument.Range
  With rng.Find
    .MatchWildcards = True
    .Text = "(*)^13"
    .Format = True
    .Style = "Heading 1"
    .Replacement.Text = Eq2 & "\1" & Eq2 & "^p"
    .Replacement.Style = "Normal"
    .Execute Replace:=wdReplaceAll
  End With

'Convert Bold to <'><'><'>word<'><'><'>
  more = True
  Do
    Set rng = ActiveDocument.Range
    With rng.Find
      .Text = ""
      .Font.Bold = True
      .Execute
```

```
    End With
    more = rng.Find.Found
    If more = True Then
      rng.Font.Bold = False
      rng.InsertBefore Tk3
      rng.InsertAfter Tk3
    End If
  Loop Until more = False

'Convert Italic to <'><'>word<'><'>
  more = True
  Do
    Set rng = ActiveDocument.Range
    With rng.Find
      .Text = ""
      .Font.Italic = True
      .Execute
    End With
    more = rng.Find.Found
    If more = True Then
      rng.Font.Italic = False
      rng.InsertBefore Tk2
      rng.InsertAfter Tk2
    End If
  Loop Until more = False

'Convert HTML sample to [pre]word[/pre]
  gogo = True
  Do
    Set rng = ActiveDocument.Range
    With rng.Find
      .Style = "HTML Sample"
      .Execute
    End With
    myStart = rng.Start
    myEnd = rng.End
    gogo = rng.Find.Found

    If gogo = True Then
    Set rng = ActiveDocument.Range
    rng.Start = myEnd
    With rng.Find
      .Style = "Default Paragraph Font"
      .Execute
    End With
    myEnd = rng.Start

    rng.Start = myStart
    rng.End = myEnd
    rng.Style = "Default Paragraph Font"

    rng.InsertBefore pr & vbCrLf
    rng.InsertAfter npr & vbCrLf
    End If

  Loop Until gogo = False
```

```
  Set rng = ActiveDocument.Range
    With rng.Find
      .MatchWildcards = False
      .Text = "^p^p" & npr
      .Replacement.Text = "^p" & npr & "^p"
      .Execute Replace:=wdReplaceAll
    End With


 End Sub
Sub WikiToStyles()
Dim Eq2, Eq3 As String
Eq2 = "=" & "=": Eq3 = Eq2 + "="
Tk2 = "'" & "'": Tk3 = Tk2 + "'"
pr = "pre": npr = "</" & pr & ">"
pr = "<" & pr & ">"

'Convert <=><=><=>Title<=><=><=> to Heading 2
  Dim rng As Range
  Set rng = ActiveDocument.Range
  With rng.Find
    .MatchWildcards = True
    .Text = Eq3 & "(*)" & Eq3 & "^13"
    .Replacement.Text = "\1^p"
    .Replacement.Style = "Heading 2"
    .Execute Replace:=wdReplaceAll
  End With

'Convert <=><=>Title<=><=> to Heading 1
  Set rng = ActiveDocument.Range
  With rng.Find
    .MatchWildcards = True
    .Text = Eq2 & "(*)" & Eq2 & "^13"
    .Replacement.Text = "\1^p"
    .Replacement.Style = "Heading 1"
    .Execute Replace:=wdReplaceAll
  End With

'Convert <'><'>word<'><'> to Bold
  Set rng = ActiveDocument.Range
  With rng.Find
    .MatchWildcards = True
    .Text = Tk3 & "(*)" & Tk3
    .Replacement.Text = "\1"
    .Replacement.Font.Bold = True
    .Execute Replace:=wdReplaceAll
  End With

'Convert <'><'>word<'><'> to Italic
  Set rng = ActiveDocument.Range
  With rng.Find
    .MatchWildcards = True
    .Text = Tk2 & "(*)" & Tk2
    .Replacement.Text = "\1"
    .Replacement.Font.Italic = True
    .Execute Replace:=wdReplaceAll
  End With
```

```
'Convert [pre]section[/pre] to HTML sample
  Do
    Set rng = ActiveDocument.Range
    fileEnd = rng.End
    With rng.Find
      .MatchWildcards = False
      .Text = "^p" & pr & "^p"
      .Replacement.Text = "^p"
      .Execute Replace:=wdReplaceOne
    End With
    codeStart = rng.End

    If rng.End <> fileEnd Then
      Set rng = ActiveDocument.Range
      With rng.Find
        .MatchWildcards = False
        .Text = "^p" & npr & "^p"
        .Replacement.Text = "^p"
        .Execute Replace:=wdReplaceOne
      End With
      codeEnd = rng.End
      rng.Start = codeStart
      rng.Style = "HTML Sample"
    End If
  Loop Until rng.End = fileEnd

End Sub

Sub prime()
Selection.InsertSymbol CharacterNumber:=8242, Unicode:=True
' Or just use insert character 2032 (hex version of 8242).
End Sub
Sub primeDouble()
Selection.InsertSymbol CharacterNumber:=8243, Unicode:=True
' Or just use insert character 2033 (hex version of 8243).

End Sub

Sub PhraseCount()
    caseSensitive = True
 ' Hide the hyphens
    Dim rng As Range
    Set rng = ActiveDocument.Range
    With rng.Find
      .ClearFormatting
      .MatchCase = False
      .MatchWildcards = False
      .Text = "-"
      .Replacement.Text = "zczc"
      .Execute Replace:=wdReplaceAll
    End With

 ' Find beginning of phrase list
    Selection.HomeKey Unit:=wdStory
    With Selection.Find
        .ClearFormatting
```

```
      .Text = "......^p"
      .MatchWildcards = False
      .Execute
    End With
    Selection.Collapse wdCollapseEnd
    listStart = Selection.End

    totAltotal = 0
    diffWords = 0
    myPhrase = "something"
    Do While Asc(myPhrase) > 13
     ' Select the next phrase
      With Selection.Find
        .ClearFormatting
        .MatchWildcards = True
        .Text = "*^13"
        .Execute
      End With

      Selection.End = Selection.End - 1
      myPhrase = Selection

      If Asc(myPhrase) > 13 Then
       ' Prepare to find the phrase
        Set rng = ActiveDocument.Range
        With rng.Find
          .ClearFormatting
          If caseSensitive = False Then
            .MatchCase = False
          Else
            .MatchCase = True
          End If
          .MatchWholeWord = True
          .MatchWildcards = False
          .Text = myPhrase
          .Execute
        End With

      ' Count the no. of occurrences
        WordsTotal = 0
        Do
          WordsTotal = WordsTotal + 1
          rng.Find.Execute
        Loop Until rng.End > listStart
      ' Type frequency figure next to the phrase
        Selection.Collapse wdCollapseEnd
        Selection.TypeText Text:=vbTab & Str(WordsTotal)
        Selection.MoveRight Unit:=wdCharacter, Count:=1
      ' Update totals
        totAltotal = totAltotal + WordsTotal
        diffWords = diffWords + 1
      End If

    Loop

  ' Restore the hyphens
    Set rng = ActiveDocument.Range
```

```
  With rng.Find
    .ClearFormatting
    .MatchCase = False
    .MatchWildcards = False
    .Text = "zczc"
    .Replacement.Text = "-"
    .Execute Replace:=wdReplaceAll
  End With

  Selection.TypeText Text:=vbCrLf & "Words counted = " & _
  Str(totAltotal) & vbCrLf & "Different words =" & Str(diffWords)

End Sub



Sub WikiConvert()
' I don't think this is used for anything (30.11.09). Is it?!

 With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = ""
  .Font.Size = 18
  .Replacement.Text = "==^&zz"
  .Replacement.Font.Bold = False
  .Wrap = wdFindContinue
  .MatchWildcards = True
  .Format = True
 End With
 Selection.Find.Execute Replace:=wdReplaceAll

 With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = ""
  .Font.Size = 14
  .Replacement.Text = "===^&zzz"
  .Replacement.Font.Bold = False
  .Wrap = wdFindContinue
  .MatchWildcards = True
  .Format = True
 End With
 Selection.Find.Execute Replace:=wdReplaceAll

 With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = ""
  .Font.Bold = True
  .Replacement.Text = """^&"""
  .Wrap = wdFindContinue
  .MatchWildcards = True
  .Format = True
 End With
 Selection.Find.Execute Replace:=wdReplaceAll
```

```
With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = ""
  .Font.Italic = True
  .Replacement.Text = ""^&""
  .Wrap = wdFindContinue
  .MatchWildcards = True
  .Format = True
End With
Selection.Find.Execute Replace:=wdReplaceAll

With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "^pzzz"
  .Replacement.Text = "===^p"
  .Wrap = wdFindContinue
  .MatchWildcards = False
  .Format = True
End With
Selection.Find.Execute Replace:=wdReplaceAll

With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = "^pzz"
  .Replacement.Text = "==^p"
  .Wrap = wdFindContinue
  .MatchWildcards = False
  .Format = True
End With
Selection.Find.Execute Replace:=wdReplaceAll

End Sub
```

To: <SfEPLine@yahoogroups.com>

Date: Wed, 21 Oct 2009 09:30:39 +0100
Subject: RE: [SfEPLine] Re: TextStat help, please

Paul wrote:

Sorry to have caused confusion: it's TextAnal that we're now asking about on Vista, not TextStat - though I can't see why a macro should be affected by the operating system.

Maybe it's the version of Word, Davina (who it was that asked the question originally)? I do know we had a minor hiccup with FRedit not working on the Mac version of Word.

I was the person who raised the query about TextStat initially (and, Paul, may I make a plea for a change of your macro's name to TextAnalysis, please?). In offline correspondence with Davina, we've established that I need to download the version of TextStat from Softpedia (the link that Paul gives for Vista users). The version that I downloaded from Matthias Huening's website (http://neon.niederlandistik.fu-berlin.de/textstat/) seems to be different, and, looking at the Softpedia details (which has the name of another developer), I'm wondering whether they are entirely different programs with the same name, hence the confusion. My problem was that I don't have a File menu on the top left as many of you do (only an Export menu, which won't give me a text file), so all the helpful suggestions from those who replied to my query (thank you!) don't help at the moment. I'm going to download the Softpedia version now and see how it goes.

Best wishes

Jacqueline

Date: Wed, 21 Oct 2009 09:50:14 +0100
Subject: RE: [SfEPLine] Re: TextStat help, please

Problem solved. The program on http://www.softpedia.com/progDownload/TextStat-Download-44558.html <http://www.softpedia.com/progDownload/TextStat-Download-44558.html%20is%20TextStat%203.0> is TextStat 3.0, developed by Lionel Allorge, whereas that on http://neon.niederlandistik.fu-berlin.de/textstat/ is TextSTAT, developed by Matthias Huening. They are different programs. The first is not simply a driver for Vista; both work on Vista, but I had problems getting TextSTAT to export a list in other than an Excel format. Has anyone actually used TextSTAT successfully with Paul's macro?

Best wishes

Jacqueline

==========

==========

Trim(MyText) takes spaces off the ends of the string.
Ltrim and Rtrim take spaces of one or other end.

```
Sub AllLowerCase()
'
' Macro2 Macro
' Macro recorded 2/2/2008 by Paul Beverley
'
    Selection.Range.Case = wdLowerCase
End Sub


Sub NewComment()
'
' Macro1 Macro
' Macro recorded 2/7/2008 by Paul Beverley
'
    WordBasic.InsertNewComment
End Sub


Sub PrepToFind()
'
' Macro recorded 2/9/2008 by Paul Beverley
'
    Dim MyData As DataObject
    Dim strClip As String

    Set MyData = New DataObject
    MyData.GetFromClipboard
    strClip = MyData.GetText
    Selection.Find.ClearFormatting
    Selection.Find.Replacement.ClearFormatting
    With Selection.Find
        .Text = strClip
        .Replacement.Text = strClip
        .Forward = True
        .Wrap = wdFindAsk
        .Format = False
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With
    ShowVisualBasicEditor = True
    CommandBars("Stop Recording").Visible = False
```

```
End Sub
Sub AdditInfo()
'
' Macro2 Macro
' Macro recorded 2/12/2008 by Paul Beverley
'
    Selection.TypeText Text:="az1"
    Selection.MoveRight Unit:=wdCell
    Selection.MoveRight Unit:=wdCell
    Selection.TypeText Text:="az2"
    Selection.MoveRight Unit:=wdCell
    Selection.MoveRight Unit:=wdCell
    Selection.TypeText Text:="az3"
    Selection.MoveRight Unit:=wdCell
    Selection.MoveRight Unit:=wdCell
    Selection.TypeText Text:="az4"
    Selection.MoveRight Unit:=wdCell
    Selection.MoveRight Unit:=wdCell
    Selection.TypeText Text:="az5"
    Selection.MoveRight Unit:=wdCell
    Selection.MoveRight Unit:=wdCell
    Selection.TypeText Text:="az6"
    Selection.MoveRight Unit:=wdCell
    Selection.MoveRight Unit:=wdCell
    Selection.TypeText Text:="az7"
    Selection.MoveRight Unit:=wdCell
    Selection.MoveRight Unit:=wdCell
    Selection.TypeText Text:="az8"
    Selection.MoveRight Unit:=wdCell
    Selection.MoveRight Unit:=wdCell
    Selection.TypeText Text:="az9"
    Selection.MoveRight Unit:=wdCell
    Selection.MoveRight Unit:=wdCell
    Selection.TypeText Text:="az10"
End Sub
Sub NumTabTwo()
'
' Macro2 Macro
' Macro recorded 2/12/2008 by Paul Beverley
'
    Selection.HomeKey Unit:=wdLine
    Selection.MoveRight Unit:=wdCharacter, Count:=4
    Selection.Delete Unit:=wdCharacter, Count:=1
    Selection.TypeText Text:=vbTab
    Selection.MoveRight Unit:=wdCharacter, Count:=1, Extend:=wdExtend
    Selection.Range.Case = wdLowerCase
End Sub
Sub NumTabOne()
'
' Macro2 Macro
' Macro recorded 2/12/2008 by Paul Beverley
'
    Selection.HomeKey Unit:=wdLine
    Selection.MoveRight Unit:=wdCharacter, Count:=3
    Selection.Delete Unit:=wdCharacter, Count:=1
    Selection.TypeText Text:=vbTab
    Selection.MoveRight Unit:=wdCharacter, Count:=1, Extend:=wdExtend
```

```
    Selection.Range.Case = wdLowerCase
End Sub
Sub TabOneWord()
'
' Macro3 Macro
' Macro recorded 2/12/2008 by Paul Beverley
'
    Selection.HomeKey Unit:=wdLine
    Selection.MoveRight Unit:=wdCharacter, Count:=1, Extend:=wdExtend
    Selection.Range.Case = wdLowerCase
    Selection.HomeKey Unit:=wdLine
    Selection.TypeText Text:=vbTab
    Selection.MoveRight Unit:=wdWord, Count:=2
    Selection.MoveRight Unit:=wdCharacter, Count:=1, Extend:=wdExtend
    Selection.Range.Case = wdLowerCase
End Sub
Sub TabTwoWords()
'
' Macro4 Macro
' Macro recorded 2/12/2008 by Paul Beverley
'
    Selection.HomeKey Unit:=wdLine
    Selection.MoveRight Unit:=wdCharacter, Count:=1, Extend:=wdExtend
    Selection.Range.Case = wdLowerCase
    Selection.HomeKey Unit:=wdLine
    Selection.TypeText Text:=vbTab
    Selection.MoveRight Unit:=wdWord, Count:=3
    Selection.MoveRight Unit:=wdCharacter, Count:=1, Extend:=wdExtend
    Selection.Range.Case = wdLowerCase
End Sub

Sub TabThreeWords()
'
' Macro4 Macro
' Macro recorded 2/12/2008 by Paul Beverley
'
    Selection.HomeKey Unit:=wdLine
    Selection.MoveRight Unit:=wdCharacter, Count:=1, Extend:=wdExtend
    Selection.Range.Case = wdLowerCase
    Selection.HomeKey Unit:=wdLine
    Selection.TypeText Text:=vbTab
    Selection.MoveRight Unit:=wdWord, Count:=4
    Selection.MoveRight Unit:=wdCharacter, Count:=1, Extend:=wdExtend
    Selection.Range.Case = wdLowerCase
End Sub

Sub KnowHowTo()
'
' KnowHowTo Macro
' Macro recorded 2/13/2008 by Paul Beverley
'
    Selection.HomeKey Unit:=wdLine
    Selection.MoveRight Unit:=wdCharacter, Count:=3
    Selection.TypeText Text:="know how to "
    Selection.MoveRight Unit:=wdCharacter, Count:=1, Extend:=wdExtend
    Selection.Range.Case = wdLowerCase
End Sub
```

```
Sub ColonPullUp()
'
' ColonPullUp Macro
' Macro recorded 2/13/2008 by Paul Beverley
'
    Selection.EndKey Unit:=wdLine
    Selection.TypeText Text=": "
    Selection.Delete Unit:=wdCharacter, Count:=1
    Selection.MoveRight Unit:=wdCharacter, Count:=1, Extend:=wdExtend
    Selection.Range.Case = wdLowerCase
End Sub

Sub SearchReplaceCall()
    Selection.MoveRight Unit:=wdCharacter, Count:=1
    SendKeys ("^h")
    SendKeys ("^v")
    SendKeys ("{TAB}")
    SendKeys ("^v")
End Sub


Sub FindHighlight()

' FindHighlight Macro
' Macro recorded 14/01/2009 by Christine Vaughan
' This macro finds any text highlighted in red and changes
' its text colour into red, while removing the highlighting.

    Selection.HomeKey Unit:=wdStory
    Selection.Find.ClearFormatting
    Selection.Find.Highlight = True
    Selection.Find.Execute
    While Selection.Find.Found = True
    If Selection.Range.HighlightColorIndex = wdRed Then
    Selection.Range.Font.Color = wdColorRed
    Selection.Range .HighlightColorIndex = False
    Selection.MoveLeft Unit:=wdCharacter, Count:=1
    End If
    Selection.Find.ClearFormatting
    Selection.Find.Highlight = True
    Selection.Find.Execute
    Wend
    Selection.HomeKey Unit:=wdStory
End Sub



Sub WhatChar()
' This doesn't do anything useful yet but it shows me how to check what the next character is.
' Could be very useful!!

  If Asc(Selection) = 145 Then Selection.TypeText Text:="•"
End Sub


Sub TabPara()
'
```

```
' Macro1 Macro
' Macro recorded 10/29/2008 by P Beverley
' Done for someone else: find all paras with a particular indent?

    Selection.Find.ClearFormatting
    With Selection.Find.ParagraphFormat
        .SpaceBeforeAuto = False
        .SpaceAfterAuto = False
        .FirstLineIndent = CentimetersToPoints(1.27)
        .CharacterUnitFirstLineIndent = 0
    End With
    Selection.Find.Replacement.ClearFormatting
    With Selection.Find.Replacement.ParagraphFormat
        .SpaceBeforeAuto = False
        .SpaceAfterAuto = False
        .FirstLineIndent = CentimetersToPoints(0)
        .CharacterUnitFirstLineIndent = 0
    End With
    With Selection.Find
        .Text = ""
        .Replacement.Text = "^t^&"
        .Forward = True
        .Wrap = wdFindContinue
        .Format = True
        .MatchCase = False
        .MatchWholeWord = False
        .MatchWildcards = False
        .MatchSoundsLike = False
        .MatchAllWordForms = False
    End With
    Selection.Find.Execute Replace:=wdReplaceAll
End Sub


==========================================
Sub WikiToStyles()

  Selection.HomeKey Unit:=wdStory

  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .MatchWildcards = True
    .Text = "===(*)===^13"
    .Replacement.Text = "\1^p"
    .Replacement.Style = "Heading 3"
    .Wrap = wdFindContinue
    .Format = True
  End With
  Selection.Find.Execute Replace:=wdReplaceAll

  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .MatchWildcards = True
    .Text = "==(*)==^13"
    .Replacement.Text = "\1^p"
```

```
  .Replacement.Style = "Heading 1"
  .Wrap = wdFindContinue
  .Format = True
End With
Selection.Find.Execute Replace:=wdReplaceAll

With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .MatchWildcards = True
  .Text = """(*)"""
  .Replacement.Text = "\1"
  .Replacement.Font.Bold = True
  .Wrap = wdFindContinue
  .Format = True
End With
Selection.Find.Execute Replace:=wdReplaceAll


With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .MatchWildcards = True
  .Text = ""(*)""
  .Replacement.Text = "\1"
  .Replacement.Font.Italic = True
  .Wrap = wdFindContinue
  .Format = True
End With
Selection.Find.Execute Replace:=wdReplaceAll


Selection.HomeKey Unit:=wdStory

With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .MatchWildcards = False
  .Text = "<pre>"
End With


Do While Selection.Find.Execute
  Do
    Selection.MoveDown Unit:=wdParagraph, Count:=1, Extend:=wdExtend
  Loop Until InStr(Selection, "</pre>")
  Selection.Style = "HTML Sample"
  Selection.Collapse wdCollapseEnd
Loop

With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .MatchWildcards = False
  .Text = "<pre>^p"
  .Replacement.Text = ""
  .Wrap = wdFindContinue
```

```
    .Format = True
  End With
  Selection.Find.Execute Replace:=wdReplaceAll

  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .MatchWildcards = False
    .Text = "^p</pre>"
    .Replacement.Text = ""
    .Wrap = wdFindContinue
    .Format = True
  End With
  Selection.Find.Execute Replace:=wdReplaceAll


End Sub


==========================================
Sub ListAllWords()
    caseSensitive = False
 ' Minimum length of word to be recorded
    minWordLength = 1
 ' (1 = all words, 2 = two-letter words and longer, etc)
    Dim rng As Range, myWord As String
    Set rng = ActiveDocument.Range

 ' Hide the hyphens
    With rng.Find
      .ClearFormatting
      .MatchCase = False
      .MatchWildcards = False
      .Text = "-"
      .Replacement.Text = "zczc"
      .Execute Replace:=wdReplaceAll
    End With

 ' Hide the close single curly quotes
    Set rng = ActiveDocument.Range
    With rng.Find
      .ClearFormatting
      .Text = "’"
      .Replacement.Text = " ]{[}"
      .Execute Replace:=wdReplaceAll
    End With

    Selection.EndKey Unit:=wdStory
    ' Make sure list is in Normal style
    Selection.TypeParagraph
    Selection.Range.Style = ActiveDocument.Styles(wdStyleNormal)
    Selection.TypeText Text:="==========" & vbCrLf

    ' Highlight just the text
    Selection.WholeStory
    Selection.End = Selection.End - 12
    Selection.Range.HighlightColorIndex = wdYellow
```

```
  Selection.EndKey Unit:=wdStory

  gogo = True
  Do While gogo = True
   ' Find any highlighted word
     Set rng = ActiveDocument.Range
     With rng.Find
        .ClearFormatting
        .MatchWildcards = True
        .Highlight = True
        .Replacement.Highlight = False
        .Text = "<*>"
        .Execute Replace:=wdReplaceOne
     End With

   ' Did you find a highlighted word?
     gogo = rng.Find.Found
   ' If so, unhighlight all occurrences of that word
     If gogo = True Then
        myWord = rng
        Set rng = ActiveDocument.Range
        With rng.Find
           .ClearFormatting
           .Replacement.Highlight = False
           .MatchWildcards = False
           .MatchWholeWord = True
           If caseSensitive = False Then
              .MatchCase = False
           Else
              .MatchCase = True
           End If
           .Text = myWord
           .Execute Replace:=wdReplaceAll
        End With
      ' force word to lower case
        If caseSensitive = False Then
           myWord = StrConv(myWord, vbLowerCase)
        End If
      ' Add the word to the list
        If Len(myWord) > minWordLength - 1 Then
           Selection.TypeText Text:=myWord & vbCrLf
        End If
     End If
  Loop

  Selection.WholeStory
  Selection.Range.HighlightColorIndex = wdNoHighlight

' Restore the hyphens
  Set rng = ActiveDocument.Range
  With rng.Find
     .ClearFormatting
     .MatchCase = False
     .MatchWildcards = False
     .Text = "zczc"
     .Replacement.Text = "-"
     .Execute Replace:=wdReplaceAll
```

```
  End With

' Restore the close single curly quotes
  Set rng = ActiveDocument.Range
  With rng.Find
    .ClearFormatting
    .Text = " ]{[}"
    .Replacement.Text = "’"
    .Execute Replace:=wdReplaceAll
  End With

' Find beginning of word list
  Selection.HomeKey Unit:=wdStory
  With Selection.Find
    .ClearFormatting
    .Text = "=====^p"
    .MatchWildcards = False
    .Execute
  End With

' Select the word list and sort it alphabetically
  listStart = Selection.End
  Selection.EndKey Unit:=wdStory
  Selection.Start = listStart
  Selection.Sort
  Selection.End = Selection.Start

End Sub

Sub WordFrequency()
  Call ListAllWords
  Call PhraseCount
End Sub

Sub PhraseCount()
  caseSensitive = False
' Hide the hyphens
  Dim rng As Range
  Set rng = ActiveDocument.Range
  With rng.Find
    .ClearFormatting
    .MatchCase = False
    .MatchWildcards = False
    .Text = "-"
    .Replacement.Text = "zczc"
    .Execute Replace:=wdReplaceAll
  End With

' Find beginning of phrase list
  Selection.HomeKey Unit:=wdStory
  With Selection.Find
     .ClearFormatting
     .Text = "=====^p"
     .MatchWildcards = False
     .Execute
  End With
  Selection.Collapse wdCollapseEnd
```

```
  listStart = Selection.End

  totaltotal = 0
  diffWords = 0
  myPhrase = "something"
  Do While Asc(myPhrase) > 13
   ' Select the next phrase
    With Selection.Find
      .ClearFormatting
      .MatchWildcards = True
      .Text = "*^13"
      .Execute
    End With

    Selection.End = Selection.End - 1
    myPhrase = Selection

    If Asc(myPhrase) > 13 Then
     ' Prepare to find the phrase
       Set rng = ActiveDocument.Range
       With rng.Find
          .ClearFormatting
          If caseSensitive = False Then
             .MatchCase = False
          Else
             .MatchCase = True
          End If
          .MatchWholeWord = True
          .MatchWildcards = False
          .Text = myPhrase
          .Execute
       End With

    ' Count the no. of occurrences
       WordsTotal = 0
       Do
          WordsTotal = WordsTotal + 1
          rng.Find.Execute
       Loop Until rng.End > listStart
    ' Type frequency figure next to the phrase
       Selection.Collapse wdCollapseEnd
       Selection.TypeText Text:=vbTab & Str(WordsTotal)
       Selection.MoveRight Unit:=wdCharacter, Count:=1
    ' Update totals
       totaltotal = totaltotal + WordsTotal
       diffWords = diffWords + 1
     End If

  Loop

' Restore the hyphens
  Set rng = ActiveDocument.Range
  With rng.Find
    .ClearFormatting
    .MatchCase = False
    .MatchWildcards = False
    .Text = "zczc"
```

```
      .Replacement.Text = "-"
      .Execute Replace:=wdReplaceAll
    End With

    MsgBox ("Words counted = " & Str(totaltotal) & vbCrLf & _
    "Different words =" & Str(diffWords))

End Sub
```

==========

==========

==========

Proper case-sensitive sorting

```
  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = " @([A-Z])"
    .Replacement.Text = "!!!\1"
    .Forward = True
    .Wrap = wdFindContinue
    .Format = False
    .MatchWildcards = True
  End With
  Selection.Find.Execute Replace:=wdReplaceAll

  Selection.WholeStory
  Selection.Sort

  Selection.HomeKey Unit:=wdStory

  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = "!!!"
    .Replacement.Text = ""
    .Forward = True
    .Wrap = wdFindContinue
    .Format = False
    .MatchWildcards = False
  End With
  Selection.Find.Execute Replace:=wdReplaceAll

  With Selection.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = " @([a-z^0145^45])"
```

```
    .Replacement.Text = "\1"
    .Forward = True
    .Wrap = wdFindContinue
    .Format = False
    .MatchWildcards = True
  End With
  Selection.Find.Execute Replace:=wdReplaceAll
```

==========

An example using on error
error handling

```
Sub Whatever()

On Error GoTo ReportIt

Dim MyText As String
MyText = Selection

Selection.MoveLeft Unit:=wdCharacter, Count:=1

With Selection.Find
  .ClearFormatting
  .Replacement.ClearFormatting
  .Text = Trim(MyText)
  .Replacement.Text = Trim(MyText)
  .Forward = True
  .Wrap = wdFindContinue
  .MatchWildcards = False
  .MatchCase = True
  .Format = True
  .MatchAllWordForms = False
End With
Dialogs(wdDialogEditReplace).Show

FinishHere:
Exit Sub

ReportIt:
MsgBox Err.Description
Resume FinishHere
End Sub
```

==========

Playing with documents docs

```
Sub UsefulBits()
'
' Macro1 Macro
' Macro recorded 6/3/2009 by P Beverley
'

For Each aDOC In Documents
```

```
    aname = aname & aDOC.Name & vbCr
Next aDOC
MsgBox aname



Exit Sub

For Each doc In Documents
    If doc.Name = "Report.doc" Then Found = True
Next doc
If Found <> True Then
    Documents.Open fileName:="C:\Documents\Report.doc"
Else
    Documents("Report.doc").Activate
End If


' Not useful??
  Documents(1).Activate
  Set rng = ActiveDocument.Content
  lengthOne = rng.End

  Documents(2).Activate
  Set rng = ActiveDocument.Content
  lengthTwo = rng.End

  If lengthTwo < lengthOne Then
    Set theList = ActiveDocument
    Documents(1).Activate
    Set theText = ActiveDocument
  Else
    Set theText = ActiveDocument
    Documents(1).Activate
    Set theList = ActiveDocument
  End If


End Sub


==========================================

Spelling errors spell errors

Dim sWord As String
Dim oSource As Document
Dim oTarget As Document
Set oSource = ActiveDocument
Set oTarget = Documents.Add
For i = 1 To oSource.Words.Count
    sWord = oSource.Words(i)
    If CheckSpelling(sWord, IgnoreUppercase:=False) = False Then
        oTarget.Range.InsertAfter sWord & vbCr
    End If
Next i
With oTarget
```

```
    .Range.End = .Range.End - 1
    .Range.Sort
    .Paragraphs(1).Range.Delete
    .Activate
End With
```

==========

An example showing how to do dialog boxes
dialogue boxes

```
Sub ReplaceWithSelect()
' <alt-ctrl-H>

Dim MyText As String
MyText = Selection
On Error GoTo ReportIt

Selection.MoveLeft Unit:=wdCharacter, Count:=1

With Dialogs(wdDialogEditReplace)
  .Find = Trim(MyText)
  .Replace = Trim(MyText)
  .MatchCase = True
  .Show
End With
FinishHere:
Exit Sub

ReportIt:
 MsgBox Err.Description
 Resume FinishHere
End Sub
```

==========

The following example cuts the last paragraph of the first document in the Documents collection and pastes it at the beginning of the second document.

```
With Documents(1)
    .Paragraphs.Last.Range.Select
    .ActiveWindow.Selection.Cut
End With

With Documents(2).ActiveWindow.Selection
    .StartOf Unit:=wdStory, Extend:=wdMove
    .Paste
End With
```

==========

Functions

```
Sub test()
' <ctrl-alt-shift-T>
Dim aname$

aname$ = "hello"
MsgBox (Reverse(aname$))

End Sub


Function Reverse(nm As String)

Temp$ = ""
For i = 1 To Len(nm)
Temp$ = Mid$(nm, i, 1) + Temp$
Next
Reverse = Temp$

End Function
```

==========

==========

==========

==========

==========

==========

==========

==========

Smart quotes on and off

```
With Options
   .AutoFormatAsYouTypeReplaceQuotes = True
End With
```

==========

file search file find

```
With Application.FileSearch
   .FileName = "99*.*"
   .LookIn = "C:\My Documents"
   .Execute
   For I = 1 to .FoundFiles.Count
      MsgBox .FoundFiles(I)
   Next I
End With
```

==========

http://www.jojo-zawawi.com/code-samples-pages/code-samples.htm
Zawawi
Zawawi
Zawawi
Word-VBA
Code Samples
by JoJo Zawawi

There are lots of different ways to code something, depending on what you're trying to accomplish. Some ways may be efficient for one use but inefficient for another. The purpose of this listing is to show you various ways of coding the simple stuff that you can play with on your own and learn with. One of these days, I'm going to add some Sample Programs. Sorry — at the moment, I'm not able to answer questions. However, I highly recommend Allen Wyatt's Daily Word Tips e-mail list. You can contact Allen by way of his web site and request that you be added to the list.
Search Now:

In Association with Amazon.com

vba code samples

For more stuff, visit the Home page a b c d
Tax Help & Bookkeeping e f g h
Word Processing & Transcription i j k l
About Us m n o p
q r s t
u v w x
y z top

Download the NUMBERING TEMPLATE.
(Open the template for instructions on its use.)

==========

Remove all empty paragraphs from a document

Article contributed by Dave Rado

You can remove most empty paragraphs from a document by doing a wildcard Find & Replace.

Replace: ^13{2,} with ^p, which (in theory – see below) replaces all occurrences of two or more consecutive paragraph marks with one paragraph mark. Or you can run the following macro, which does the same thing:

```
With Selection.Find
    .Text = "^13{2,}"
    .Replacement.Text = "^p"
    .Forward = True
    .Wrap = wdFindContinue
    .Format = False
    .MatchCase = False
    .MatchWholeWord = False
    .MatchAllWordForms = False
    .MatchSoundsLike = False
    .MatchWildcards = True
    .Execute Replace:=wdReplaceAll
End With
```

(Note that using Find and Replace is dramatically faster than cycling through the Paragraphs collection).

However, you can't use Find & Replace to delete the first or last paragraph in the document, if they are empty. To delete them you would need to add the following code to the above macro:

```
Dim MyRange As Range
Set MyRange = ActiveDocument.Paragraphs(1).Range
If MyRange.Text = vbCr Then MyRange.Delete

Set MyRange = ActiveDocument.Paragraphs.Last.Range
If MyRange.Text = vbCr Then MyRange.Delete
```

In addition, you can't use Find & Replace to delete the paragraph immediately preceding or following any tables, if these are empty. You would need to add the following code to the macro if you want them deleted – but be careful; if two tables are separated only by an empty paragraph, the following code will merge them into one table, which may or may not be the result you wanted:1

```
Dim oTable As Table, MyRange As Range

For Each oTable In ActiveDocument.Tables
    #If VBA6 Then
        'The following is only compiled and run if Word 2000 or 2002 is in use
        'It speeds up the table and your code
        oTable.AllowAutoFit = False
    #End If

    'Set a range to the para following the current table
     Set MyRange = oTable.Range
```

```
    MyRange.Collapse wdCollapseEnd
    'if para after table empty, delete it
    If MyRange.Paragraphs(1).Range.Text = vbCr Then
        MyRange.Paragraphs(1).Range.Delete
    End If

     'Set a range to the para preceding the current table
    Set MyRange = oTable.Range
    MyRange.Collapse wdCollapseStart
    MyRange.Move wdParagraph, -1
     'if para before table empty, delete it
    If MyRange.Paragraphs(1).Range.Text = vbCr Then
        MyRange.Paragraphs(1).Range.Delete
    End If

Next oTable
```

You also can't use Find & Replace to delete the first or last paragraph in a table cell, if empty. If the user inserted an empty paragraph at the start or end of a table cell (in order to simulate “space before paragraph” or “space after paragraph”), you have to use something like the following to remove those empty paragraphs:

```
Dim oTable As Table, oCell As Cell, MyRange As Range
For Each oTable In ActiveDocument.Tables
    'Using oCell.Next to cycle through table cells is much quicker
    ' in long tables than using For Each oCell
    Set oCell = oTable.Range.Cells(1)
    For Counter = 1 To oTable.Range.Cells.Count

        If Len(oCell.Range.Text) > 2 And _
            oCell.Range.Characters(1).Text = vbCr Then
            'if cell is NOT blank, but it starts with a blank paragraph, delete the blank para
            'Note that a blank cell contains 2 characters;
            'a paragraph mark and an end of cell marker
            oCell.Range.Characters(1).Delete
        End If

        If Len(oCell.Range.Text) > 2 And _
            Asc(Right$(oCell.Range.Text, 3)) = 13 Then
            'if cell is NOT blank, but it ends with a blank paragraph, delete the blank para
            Set MyRange = oCell.Range
            MyRange.MoveEnd Unit:=wdCharacter, Count:=-1
            MyRange.Characters.Last.Delete
        End If

        Set oCell = oCell.Next
    Next Counter

Next oTable
```

So the complete macro would look like this:

```
Sub DeleteEmptyParas()

Dim MyRange As Range, oTable As Table, oCell As Cell

With Selection.Find
    .Text = "^13{2,}"
```

```
    .Replacement.Text = "^p"
    .Forward = True
    .Wrap = wdFindContinue
    .Format = False
    .MatchCase = False
    .MatchWholeWord = False
    .MatchAllWordForms = False
    .MatchSoundsLike = False
    .MatchWildcards = True
    .Execute Replace:=wdReplaceAll
End With

Set MyRange = ActiveDocument.Paragraphs(1).Range
If MyRange.Text = vbCr Then MyRange.Delete

Set MyRange = ActiveDocument.Paragraphs.Last.Range
If MyRange.Text = vbCr Then MyRange.Delete

For Each oTable In ActiveDocument.Tables
    #If VBA6 Then
        'The following is only compiled and run if Word 2000 or 2002 is in use
         'It speeds up the table and your code
         oTable.AllowAutoFit = False
    #End If

    'Set a range to the para following the current table
     Set MyRange = oTable.Range
    MyRange.Collapse wdCollapseEnd
    'if para after table empty, delete it
    If MyRange.Paragraphs(1).Range.Text = vbCr Then
        MyRange.Paragraphs(1).Range.Delete
    End If

     'Set a range to the para preceding the current table
    Set MyRange = oTable.Range
    MyRange.Collapse wdCollapseStart
    MyRange.Move wdParagraph, -1
     'if para before table empty, delete it
    If MyRange.Paragraphs(1).Range.Text = vbCr Then
        MyRange.Paragraphs(1).Range.Delete
    End If

Next oTable

End Sub
```

1.

You could modify the macro to cater for that; for example, if my formatting macro finds a blank paragraph separating two tables, it applies the Heading 1 style to that paragraph and inserts the text: “Heading text needs to go here” at that point; and at the end of the macro, a message box is displayed (when appropriate) warning the user that they need to type meaningful heading text at those places, and explaining how to find them. However, the code to do that is beyond the scope of this article.

Click to view Terms of Use page

Click to view Disclaimer page

==========

##########