Wildcard Cookbook

for Microsoft Word

Jack Lyon

What a Deal!

If you downloaded this book, you are eligible for a $30 discount on Editor's ToolKit PLUS 2014, the Editorium's premier Microsoft Word add-in. You can learn more (and download the program) here:

http://www.editorium.com/ETKPlus2014.htm

To get the discount, use the following redemption code when you place your order:

ETKP30

You can place your order here:

https://www.swreg.org/com/storefront/47578/product/47578-26

Thanks!

Wildcard Cookbook for Microsoft Word

Wildcard Cookbook for Microsoft Word

Jack Lyon

The Editorium

Please read this: The reader (that's you) assumes the entire risk as to the use of this book and the information therein. I don't anticipate any problems, but all computers and Microsoft Word installations are different, and I can't be held responsible for what might happen with yours. To avoid problems, be sure to *back up your files* before trying anything in this book; then you'll have something to go back to if anything goes wrong. As the lawyers say, this book is provided for informational purposes only and without a warranty of any kind, either express or implied, including but not limited to implied warranties of merchantability, fitness for a particular purpose, and freedom from infringement.

ISBN 978-1-4341-0398-7

Printed in the United States of America

The Editorium
West Jordan, UT 84081-6132
www.editorium.com
editor@editorium.com

Contents

Preface

Okay, so this isn't really a cookbook. Well, it's *kind* of a cookbook, because it will help you search a Word document with a combination of wildcards[1] and other codes that will give you delicious results. Microsoft Word's advanced search features are extremely powerful, but they're also virtually undocumented, and most explanations of their use have been limited to a simple table of various wildcards. I wrote and published a series of articles to remedy that situation, then packaged the articles as a downloadable paper called "Advanced Find and Replace in Microsoft Word." This book incorporates the material I've previously written, updates and reorganizes that material, and adds a wealth of new material, all of which I hope you'll find extremely useful. Although I write and sell Microsoft Word macros for a living, the tools I depend on most are the advanced features of Word's find and replace. Learning to use these tools takes time and effort, but the payoff is huge.

Want to learn? Then don't just *read* this book; rather, work *through* the book, experimenting with the techniques it explains on some junk documents that you no longer need. That will help you understand how all of this works, and it will also help you internalize the techniques so that you no longer need the book (except for reference).

I hope this book will help you understand how powerful Word's advanced search features can be and how much time they can save you. Using these features, you can quickly fix repetitive problems that would take hours to correct by hand.

Enjoy!

---

1. Many other programs use wildcards, but they're usually called "regular expressions" (also known as "regex") and are actually pretty standard from program to program. Microsoft Word uses a modified form of these tools.

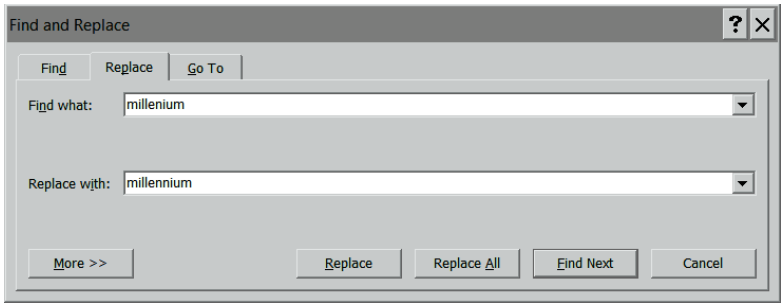
Chapter 1

Basic Find and Replace

Replacing Basic Text

You probably already know how to find and replace in Microsoft Word, but if not, here are the basic steps:

1. Press CTRL + H (or click Home > Editing > Replace on Word's ribbon interface). This will open the "Find and Replace" dialog.
2. In the "Find what" box, enter a word you want to search for. (We'll use the misspelled "millenium" for an example.)
3. In the "Replace with" box, enter a word you want to replace with. (We'll use the correctly spelled "millennium" for an example.)
4. Click the Replace All button.

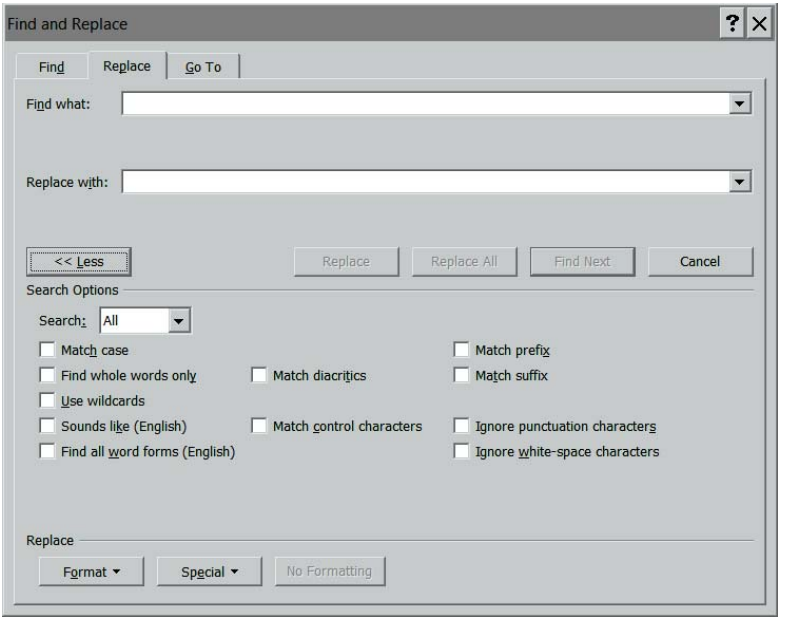


That's it. Every occurrence of "millenium" will be replaced with "millennium." Simple and quick.

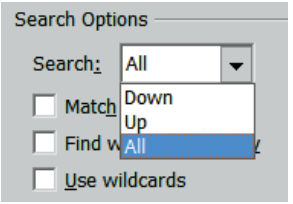
Refining Your Search with More Options

But wait—there's "more"! Microsoft Word provides many ways to *refine* your search. See that button at the bottom of the Replace dialog?

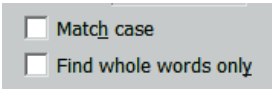
Click it. Here's what you'll get:



Under "Search Options," you can specify whether to search up, down, or through all your text:



You can also match case and find whole words only:



There are actually lots of options, all worth exploring:

Match case

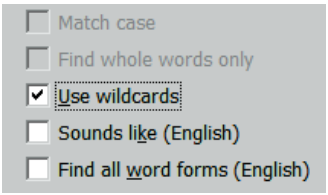
Obviously, this option finds only text that matches the case (capitalized or lowercased) of the text in the "Find what" box. If you enter "Hello," Word finds "Hello" but not "hello." If you enter "hello," Word finds "hello" but not "Hello."

Find whole words only

This option finds whole words only. For example, if you search for "sing," Word finds "sing" but not "Singapore." If this option is *not* checked, Word finds both "sing" *and* "Singapore."

Use wildcards

This option tells Word that you want to search using wildcards:



Very important. We'll discuss it in great detail later in the book.

Sounds like (English)

This option finds words that sound like the word in the "Find what" box. For example, if you search for "cot," Word also finds "caught." If you search for "horse," Word also finds "hoarse." This could be useful if you're working on a document in which certain words have been confused or mistyped. Basically, this feature works on words that are homophones; it doesn't seem to work on words that sound *almost* alike, such as "horse" and "whores." On

the other hand, while searching for "horse," it also finds "horsey" but not "horses," so who knows?

Find all word forms (English)

This option finds what Microsoft calls "all" forms of the word in the "Find what" box. For example, if you search for "sit," Word also finds "sat" and "sitting." The word "all" is a little misleading, however. The feature relies on an underlying database of word forms that is pretty good but has some omissions. For example, if you search for "eat," Word finds "eat, "ate," "eaten," and "eating" but not "eater." Similary, if you search for "horse," Word finds "horse," "horses," and "horsing" but not "horseless." It's a useful feature, mostly for finding verb forms; just don't expect it to actually find *all* forms of a word.

Match prefix

This option matches words beginning with the search string. For example, if you put "pre" in the "Find what" box, Word finds "prepare," "present," and so on. This isn't a "smart" feature; it searches for characters only, not word roots. For example, searching for "pre" also finds "prestidigitation" and "pressure," even though "pre" isn't really a prefix in those words.

Match suffix

This option matches words ending with the search string. For example, if you put "ing" in the "Find what" box, Word finds "singing," "typing," and so on. This isn't a "smart" feature; it searches for characters only, not word roots. For example, searching for "ing" alsos find "boing," "spring," and "thing," even though "ing" isn't really a suffix in those words.

Ignore punctuation characters

Ignores punctuation characters between words. For example, "trees plants and flowers" finds "trees, plants, and flowers" as well as "trees plants and flowers." This might be useful for fixing problems with serial commas.

Ignore white-space characters

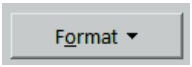
Ignores all white space (spaces, tabs, and so on) between words. For example, "webpage" finds "web page" as well as "webpage." This is the inverse of "Find whole words only" and could be useful for fixing words that are sometimes spelled open and sometimes closed.

Other options

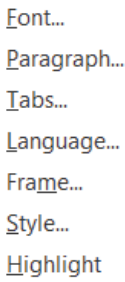
If you're working in a language other than English, other options may be available, including Match Kashida, Match Diacritics, Match Alef Hamza, and Match Control. I know almost nothing about these options, so I can't comment on them with any degree of expertise.

Format

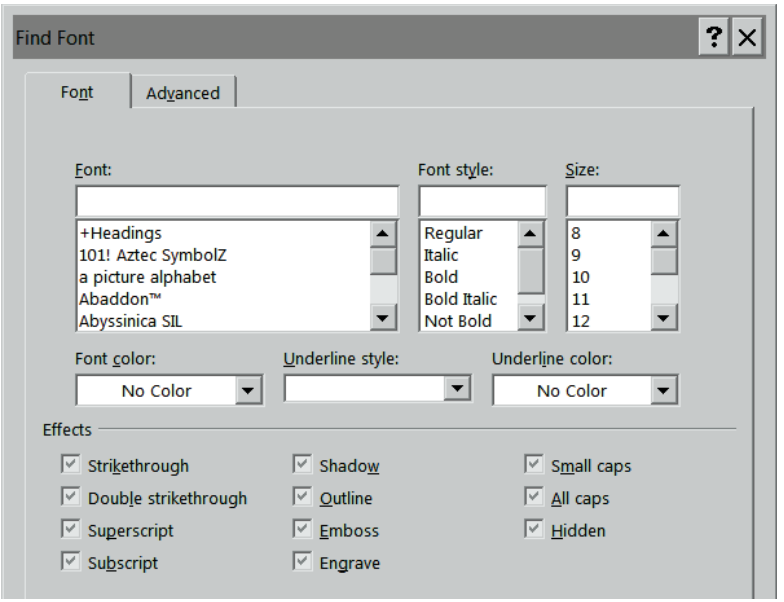
One of the most important tools in Microsoft Word's find and replace toolbox is the ability to search for formatting—all kinds of formatting. To do so, click the Format button:



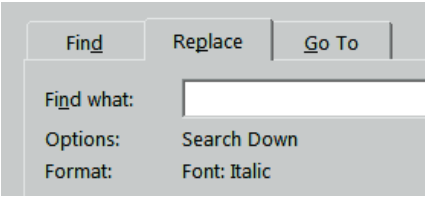
Here's what you'll get:



Each option (such as "Font") opens the usual dialog for that feature:



I won't go into all of the options in these dialogs as they're basically the same ones you'd get while formatting any text in Word. "Font" displays font options, "Styles" displays styles, and so on. You can select any of those options and use them as something to find or replace. For example, if your cursor is in the "Find what" box and you select "Italic" in the "Find Font" dialog, here's what you'll get:



Now Word will find text in italics but not in roman. If you also enter a word, you'll find *that word* in italic but not in roman. If you *don't* enter a word, you'll find *anything* formatted as italic.

But what about the "Replace with" box? What happens if you use formatting there?

If the "Replace with" box includes some text, whatever is found will be replaced by that text in the format you specified.

If the "Replace with" box *doesn't* include text, whatever is found will be replaced with *itself* in the format you specified. For example, if you search for the word "apples" to be replaced by "pears" in bold, that's exactly what you'll get—"pears" in bold. If you search for the word "apples" to be replaced by bold *alone* (with no text), you'll get "apples" in bold.

If, on the other hand, you search for "apples" but don't specify text *or* formatting in the "Replace with" box, "apples" will be replaced with nothing. In other words, it will be deleted.

Many variations are possible. Here's a basic summary:

Find	*Replace*	*Result*
apples	pears	pears
apples	pears [bold]	pears [bold]
apples	[bold]	apples [bold]
apples	[nothing]	[apples deleted]
[bold]	[nothing]	[bold text deleted]
[bold]	pears	[bold text becomes "pears" in bold]
[bold]	pears [italic]	[bold text becomes "pears" in bold italic]
[bold]	[italic]	[bold text becomes bold italic]

Note that you can also specify *not* a certain kind of formatting, such as "not bold" or "not italic" in either find or replace. You can also use combinations of formatting (and "not" formatting). For example, you can search for bold but replace with italic and *not* bold, which will turn any bold text into italic (but not bold italic) text.

Chapter 2

Finding and Replacing with Word's Built-in Codes

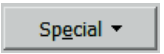
This book is mainly about using wildcards in Microsoft Word. But before we discuss that fascinating subject, you should know that Microsoft Word includes lots of built-in find and replace codes that are *not* wildcards (although lots of people call them that).

You can use Word's built-in codes to search for things like paragraph breaks, tabs, section breaks, column breaks, dashes, footnotes, endnotes, graphics, and many other things that aren't actual text, and codes are a whole lot easier to use than wildcards. In fact, codes should be your default tool; you should use wildcards only when built-in codes won't do what you need (which is actually fairly often, unfortunately).

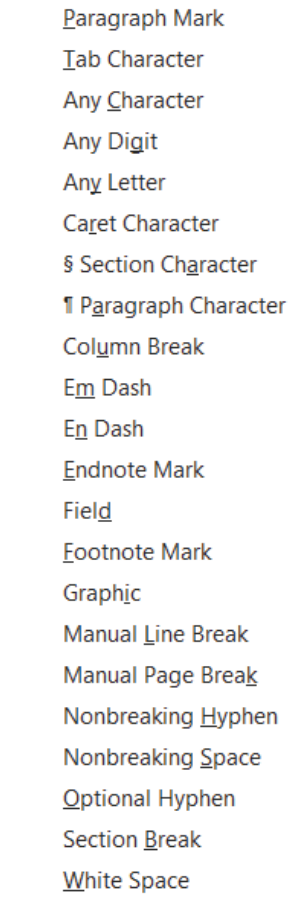
Some of Word's built-in codes can be used only in the "Find what" box; others can be used only in the "Replace with" box. Some of the codes can be used in both boxes.

"Find What" Codes

To see the codes that can be used in the "Find what" box, put your cursor in the box. Now see that "Special" button at the bottom of the "Find and Replace" dialog?



Go ahead, click it. You'll get a list like this:



Now click one of those items—let's say "Paragraph Mark." You'll get the following code in the "Find what" box (since that's where your cursor was located):

```
^p
```

That tells Word to find a paragraph break—that is, the end of a paragraph.

Each item on the list will insert a different code. For example, here's the code for an em dash:

```
^+
```

And here's the code for an en dash:

^=

"Replace With" Codes

Now put your cursor in the "Replace with" box and click the "Special" button again. This time, you'll get a different list:

Paragraph Mark
Tab Character
Caret Character
§ Section Character
¶ Paragraph Character
Clipboard Contents
Column Break
Em Dash
En Dash
Find What Text
Manual Line Break
Manual Page Break
Nonbreaking Hyphen
Nonbreaking Space
Optional Hyphen

Again, clicking one of the list items will insert a code into the "Replace with" box. For example, if you click "Clipboard Contents" you'll get this:

^c

That's an extremely useful code, because ordinarily the "Replace with" box can hold no more than 255 characters. But using the ^c code, you can replace with anything that is currently copied to the Clipboard, which can hold many pages of text, graphics, or anything else.

After you've worked with built-in codes for a while, you'll find it easy to just type them in by hand. In the meantime, you can use the "Special" lists to insert them.

You can also use combinations of codes. For example, you could search for tabs followed by paragraph breaks (^t^p) and replace them with paragraph breaks alone (^p).

Using Built-in Codes

Codes make it possible to find and replace things you ordinarily couldn't, such as graphics, dashes, and symbols. This can be a big help in cleaning up all kinds of editorial and typographical problems that you'd otherwise have to fix by hand. Here's a summary of the built-in codes and where they can be used:

Character or object	*Find what*	*Replace with*
Annotation Mark (comment)	^a	
Any character	^?	
Any digit	^#	
Any letter	^$	
Caret character	^^	^^
Clipboard contents		^c
Column break	^n	^n
"Find what text" (whatever was found during your search)		^&
Em dash	^+	^+
En dash	^=	^=
Endnote mark	^e	
Field	^d	
Footnote mark	^f	
Graphic	^g	
Line break	^l	^l
Manual page break	^m	^m
Nonbreaking hyphen	^~	^~
Nonbreaking space	^s	^s
Optional hyphen	^-	^-
Paragraph mark	^p	^p
Section break	^b	
Tab character	^t	^t
White space	^w	

Replacing with "Find What Text"

One of the most important of Word's built-in codes is what Microsoft calls the "Find What Text" code. The code looks like this—

^&

—and it stands for whatever was *found* during your search. That sounds rather confusing, but let's try an example for clarification.

Let's say you need to add the HTML italic tags <I> and </I> around anything formatted with italic. (If you don't understand HTML, don't worry. You'll soon see the point of this example.) You might think you'd need a macro to add the tags, but you don't. You can easily do it like this:

1. With your cursor in the "Find what" box, turn on italic formatting (CTRL+I) so that the word "Italic" is displayed below the box. Make sure the box itself is empty.
2. In the "Replace with" box, enter "<I>^&</I>" (if you want, you can also set this box to "Not Italic" by pressing CTRL+I a couple of times).
3. Click the "Replace All" button. Any italicized text will be surrounded by the HTML italic tags.

The ^& code in the "Replace with" box represents the text you specified in the "Find what" box. In this case, that's any text with italic formatting. What you're saying is, "Find any text in italic and replace it with *itself* surrounded by HTML italic codes." As a specific example, let's take the following line,

This is a test to *see* what will happen.

When you use the find and replace procedure above, you'll get the following result:

This is a test to <I>see</I> what will happen.

You can use the same principle to manipulate text in a variety of ways:

- Put quotation marks around the titles of magazine articles that an author has italicized.
- Insert a bullet in front of every paragraph formatted with Heading 3 style.
- Insert "Chapter" in front of every number formatted with Heading 1 style.

And so on. Any time you need to add something to unspecified text that's formatted in a specific way, try using "Find What Text."

Here's another example—using the "Find What Text" code to change the format of note numbers. I'm going to use footnotes as an example, but you can do the same thing with endnotes.

When you create footnotes in Microsoft Word, the footnote numbers are formatted in superscript, like this:

[1] This is the text of note 1.
[2] This is the text of note 2.

And so on. But sometimes you might want your footnote numbers to have regular formatting and be followed by a period, like this:

1. This is the text of note 1.
2. This is the text of note 2.

Microsoft Word has no numbering option that will do this. Nevertheless, there *is* a way to do it, using "Find What Text":

1. Open a document containing footnotes (be sure to keep a backup copy of the document, just in case).
2. Make sure the document is in Draft view (View > Draft).
3. Open the footnote pane (References > Show Notes).
4. Make sure your cursor is at the top of the footnote pane.
5. Open the "Find and Replace" dialog (Home > Replace).
6. In the "Find what" box, enter "^f" (don't include the quotation marks). ^f is the code that represents a footnote number.

7. In the "Replace with" box, enter "^&." (don't include the quotation marks). Be sure to include the period after the ampersand. The ^& code itself represents any text that was found, or in other words, the "Find What Text."
8. With your cursor in the "Replace with" box, click the "Format" button. (You may need to click the "More" button first.)
9. Click "Font."
10. In the "Find Font" dialog, clear the "Superscript" checkbox so that the replacement text won't be formatted in superscript.
11. Click the "OK" button to close the dialog.
12. In the "Find and Replace" dialog, click the "Replace All" button.

Your footnotes will now be formatted like this:

1. This is the text of note 1.
2. This is the text of note 2.

Pretty neat! Remember, however, that if you now add another footnote, its number will be formatted in the superscript default, and you'll have to fix it by hand. To do so:

1. Select the number.
2. Press CTRL+SPACE to remove the superscript format.
3. Type a period after the number.

WARNING: Be careful not to delete a note number or type a note number by hand. Microsoft Word uses a special code to represent a note number, and if you fool around with that code, you risk corrupting your file. You can, however, delete or move a note reference number that appears in the body of your document, like this,[3]and Microsoft Word will automatically renumber your notes, leaving their new formatting intact.

I ordinarily advise people not to mess around with automatic note numbers, because it's fairly easy to corrupt a document by doing so. If you know what you're doing, however, you can at least change the formatting of the note numbers if you really need to. Now you know how!

Chapter 3

Finding and Replacing with Character Codes

In addition to using Word's built-in codes, you can find and replace using numeric character codes. But before getting into that, we need to look at some history of the character sets available on computers.

In the beginning was ASCII (American Standard Code for Information Interchange), and ASCII was limited to 128 characters (numbered 0 to 127). To quote Wikipedia, "The first edition of the standard was published during 1963. . . . The characters encoded are numbers 0 to 9, lowercase letters a to z, uppercase letters A to Z, basic punctuation symbols, control codes that originated with Teletype machines, and a space."

As you can imagine, 128 characters wasn't nearly enough; ASCII didn't even include diacritics. So using character codes from the American National Standards Institute (ANSI), Microsoft extended the number of characters available in Word to 256 (numbered 0 to 255). That made it possible to access what they called "foreign-language" and other special characters by using "code pages" with different fonts. If you've clicked Insert > Symbol > More Symbols and then changed the font on the drop-down list in the Symbol dialog, you've seen how this works: the same character "position" (or number) often displays a different character in different fonts.

But what if you want to use special characters—*any* special characters—in the *same* font as your regular text? That's what Unicode is all about. As the Unicode website explains, "Unicode provides a unique number for every character, no matter what

the platform, no matter what the program, no matter what the language." How many characters? The latest version has a repertoire of more than 120,000 characters covering 129 modern and historic scripts, as well as multiple symbol sets, including Greek and Gothic, Klingon(!) and Korean—pretty much anything you might need.

In Microsoft Word, you can find and replace using any of these three kinds of numeric character codes:

- ASCII
- ANSI
- Unicode

ASCII

I'm not going to say much about using ASCII codes, as they're the same as the first 128 of the ANSI codes, just not preceded by a zero. For example, here's the ANSI code for a carriage return:

^013

And here's the *ASCII* code for a carriage return:

^13

It's easier to type the ASCII code (three keystrokes rather than four), but in my opinion it's better to get into the habit of using ANSI codes (all of which begin with a zero) just so you don't have to *think* about whether to use the zero or not. It's up to you, of course; there's nothing to keep you from using both.

ANSI

To use ANSI codes for finding or replacing special characters, simply type them (preceded by a caret and a zero) into the "Find what" or "Replace with" boxes. For example, if you wanted to find a u with an umlaut, you'd enter the following code in the "Find what" box on a PC:

^0252

On a Macintosh, you'd enter this:

^0159

That's right—the codes higher than 127 don't always represent the same character from platform to platform.

Here's a list of the ANSI codes available in Microsoft Word on both Macintosh and PC (also included in the reference section at the end of the book). The list doesn't include the codes for such ordinary characters as letters of the alphabet, since you can search for those by using the characters themselves; no code is needed. Also, be aware that some *fonts* assign different characters to the numeric codes. The list below should be accurate for Times New Roman on a PC and Times on a Macintosh.

Character	*Name*	*Macintosh*	*PC*
	footnote reference	2	2
	tab	9	9
	line break	11	11
	page/section break	12	12
	paragraph break	13	13
	column break	14	14
-	nonbreaking hyphen	30	30
–	optional hyphen	31	31
	space	32	32
,	comma	226	130
ƒ	folio	196	131
„	double comma	227	132
…	ellipses	201	133
†	dagger	160	134
‡	double dagger	224	135
ˆ	caret	246	136
‰	per thousand	228	137
Š	capital S hacek		138
‹	open angle bracket	220	139
Œ	capital oe diphthong	206	140
‘	open single quote	212	145
‘	close single quote	213	146
“	open double quote	210	147

Character	*Name*	*Macintosh*	*PC*
“	close double quote	211	148
•	bullet	165	149
–	en dash	208	150
—	em dash	209	151
˜	tilde	247	152
™	trademark	170	153
š	lowercase s hacek		154
›	close angle bracket	221	155
œ	lowercase oe diphthong	207	156
Ÿ	capital Y umlaut	217	159
	non-breaking space	160	160
¡	inverted exclamation	193	161
¢	cent	162	162
£	pound	163	163
	cell (in a table)	219	164
¥	yen	180	165
¦	pipe	124	166
§	section	164	167
¨	umlaut	172	168
©	copyright	169	169
ª	ordinal, feminine	187	170
«	left chevrons	199	171
¬	not	194	172
-	soft hyphen	248	173
®	registered	168	174
¯	macron	248	175
°	degree	161	176
±	plus or minus	177	177
²	superscript 2	50	178
³	superscript 3	51	179
´	acute accent	171	180
µ	micro	181	181
¶	pilcrow	166	182
·	middle dot	225	183
¸	cedilla	252	184
¹	superscript 1	49	185
º	ordinal, masculine	188	186
»	right chevrons	200	187
¼	one-fourth		188

Character	*Name*	*Macintosh*	*PC*
½	one-half	189	189
¾	three-fourths	190	190
¿	inverted question	192	191
À	capital A, grave	203	192
Á	capital A, acute	231	193
Â	capital A, circumflex	229	194
Ã	capital A, tilde	204	195
Ä	capital A, umlaut	128	196
Å	capital A, angstrom	129	197
Æ	capital AE, diphthong	174	198
Ç	capital C, cedilla	130	199
È	capital E, grave	233	200
É	capital E, acute	131	201
Ê	capital E, circumflex	230	202
Ë	capital E, umlaut	232	203
Ì	capital I, grave	237	204
Í	capital I, acute	234	205
Î	capital I, circumflex	235	206
Ï	capital I, umlaut	236	207
Ð	capital eth		208
Ñ	capital N, tilde	132	209
Ò	capital O, grave	241	210
Ó	capital O, acute	238	211
Ô	capital O, circumflex	239	212
Õ	capital O, tilde	205	213
Ö	capital O, umlaut	133	214
×	multiply	120	215
Ø	capital O, slash	175	216
Ù	capital U, grave	244	217
Ú	capital U, acute	242	218
Û	capital U, circumflex	243	219
Ü	capital U, umlaut	134	220
Ý	capital Y, acute	89	221
Þ	capital thorn		222
ß	sharp s	167	223
à	lowercase a, grave	136	224
á	lowercase a, acute	135	225

Character	*Name*	*Macintosh*	*PC*
â	lowercase a, circumflex	137	226
ã	lowercase a, tilde	139	227
ä	lowercase a, umlaut	138	228
å	lowercase a, angstrom	140	229
æ	lowercase ae, diphthong	190	230
ç	lowercase c, cedilla	141	231
è	lowercase e, grave	143	232
é	lowercase e, acute	142	233
ê	lowercase e, circumflex	144	234
ë	lowercase e, umlaut	145	235
ì	lowercase i, grave	147	236
í	lowercase i, acute	146	237
î	lowercase i, circumflex	148	238
ï	lowercase i, umlaut	149	239
ð	lowercase eth		240
ñ	lowercase n, tilde	150	241
ò	lowercase o, grave	152	242
ó	lowercase o, acute	151	243
ô	lowercase o, circumflex	153	244
õ	lowercase o, tilde	155	245
ö	lowercase o, umlaut	154	246
÷	divide	214	247
ø	lowercase o, slash	191	248
ù	lowercase u, grave	157	249
ú	lowercase u, acute	156	250
û	lowercase u, circumflex	158	251
ü	lowercase u, umlaut	159	252
ý	lowercase y, acute	121	253
þ	lowercase thorn		254
ÿ	lowercase y, umlaut	216	255

Unicode

Unicode characters include all kinds of things—fractions, Greek, Hebrew, and much, much more. How to search for these isn't readily apparent, but there are actually two different methods that will work.

Method 1: Unicode number

Searching with Unicode numbers is similar to using ANSI codes, but you use a "u" instead of a "0" in front of the number, and of course you need to know the Unicode decimal (not HEX) number for the character. There are far too many Unicode characters to list here, but you can look them up online:

https://en.wikipedia.org/wiki/List_of_Unicode_characters

For example, to find a small Greek alpha in Microsoft Word, you'd search for this:

^u945

Method 2: Copy and paste

If you can see an example of the Unicode character in your document (or insert one), you can actually *copy* the character and then paste it into the "Find what" box. Then just search as usual.

Replacing with Unicode characters

Replacing text with Unicode characters can be a little trickier than finding them, as Word won't let you use a numeric code (like ^u945) in the "Replace with" box. I've usually had success, however, in *pasting* the character into the box. But if that doesn't work, you may be able to follow this procedure instead:

1. Find an example of the character in your document.
2. Copy the character.
3. In the "Find what" box, enter the text you want to find.
4. In the "Replace with" box, enter ^c to tell Word you want to replace with the contents of the Clipboard—in other words, with the Unicode character you copied.
5. Click the "Replace All" button.

What's That Character?

But what if you're trying to search for some obscure character in an unusual font and don't know its number in ANSI or Unicode? Here's the scenario: You open a giant document from a client and start looking through it. But what's this? The same odd character appears at the beginning of every paragraph. Must be some kind of file translation error. Odder still, Microsoft Word won't let you paste the character into its "Find and Replace" dialog, so how are you going to get rid of them all? By hand? Horrors!

If you knew the character's numeric code, you could search for it. But this character isn't on the usual list. How can you find out its numeric code? With a simple macro:

```
Sub NextCharacter()
MsgBox CStr(AscW(Selection))
End Sub
```

Here's how to put this macro into Microsoft Word so it will be available when you need it:

1. Type the text of the macro (with complete accuracy) into a document.
2. Copy the text of the macro, starting with the first "Sub" and ending with the last "Sub."
3. Click the "View" tab on Microsoft Word's ribbon.
4. Click the "Macros" button.
5. Type the name of the macro (NextCharacter) into the "Macro name" box.
6. Click the "Create" button.
7. Delete the "Sub [macro name]" and "End Sub" lines that Word created in the macro window. The macro window should now be completely empty (unless you already have other macros in there).
8. Paste the macro text at the current insertion point.
9. Click "File," then "Close and Return to Microsoft Word."

To actually use the macro:

1. Place your cursor directly in front of the character you want to identify.
2. Click the "View" tab on Microsoft Word's ribbon.
3. Click the "Macros" button.
4. Click "NextCharacter."
5. Click the "Run" button.

A message box will appear with the numeric code for the character. To dismiss the message box, click OK.

Please note that a numeric code from 0 to 255 is the same for both ANSI and Unicode. Anything else is a Unicode number.

You may want to assign a keyboard shortcut to the NextCharacter macro so you can run it with the press of a key. Here's how:

1. Click "File > Options." Click the "Customize Ribbon" button (on the left).
2. Under "Choose commands from," select "Macros."
3. Select the NextCharacter macro.
4. At the bottom of the dialog, you'll see "Keyboard shortcuts: Customize."
5. Click the "Customize" button.
6. Put your cursor in the box labeled "Press new shortcut key" and, well, press a new shortcut key (CTRL+SHIFT+N might work well for this macro).
7. Click the "Assign" button (on the lower left).
8. Click the "Close" button (on the lower right).

Now whenever you press the shortcut key you've assigned (CTRL+SHIFT+N), the macro will run, and you'll get a message box that gives you the numeric code for the character after your cursor.

Chapter 4

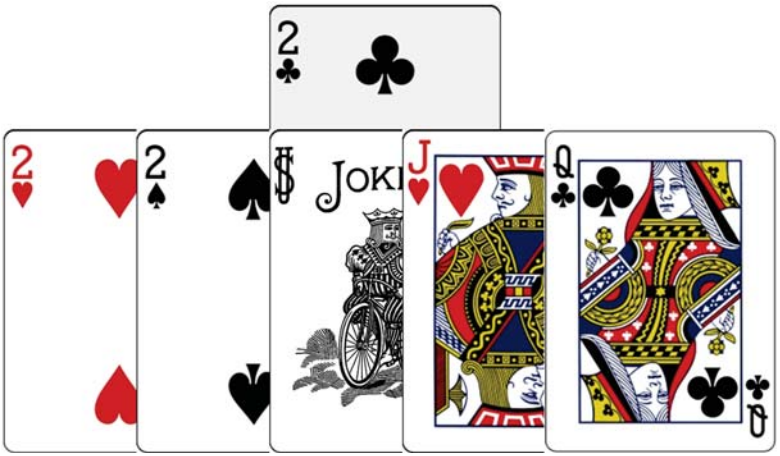
Finding and Replacing with Wildcards

The Basics

When I was in the fifth grade in wintry Idaho, rather than venturing out into the cold, some fellow students and I often spent recess playing poker. (Did our teacher know about this? I can't remember.) Being *extremely* sophisticated players, we often designated jokers *and* one-eyed jacks as wildcards—that is, they could represent any card in the deck. With the help of these wildcards, we had plenty of royal flushes, hands with five aces, and so on. Now that was poker! Here's an example:

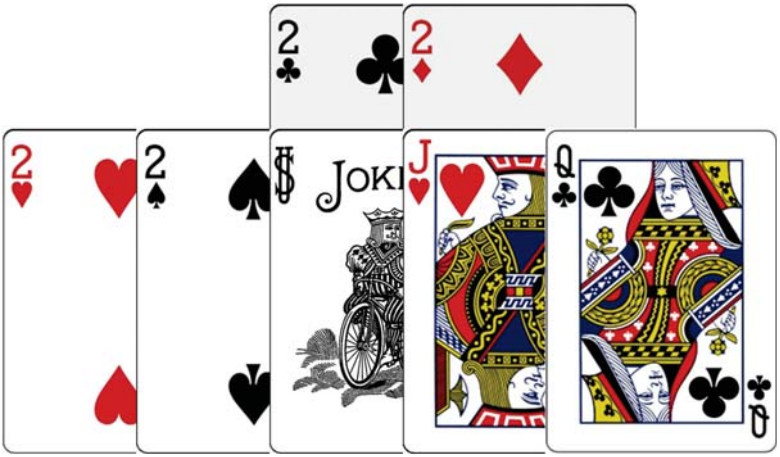
A pair of twos—not a very good hand.

But now let's suppose the joker is wild—it can represent something else: let's say another two.



That means we have three of a kind!

Now suppose the one-eyed jack is also wild, so we'll have it represent a two as well:



That gives us four of a kind, which could very well be a winning hand! (Yes, we're talking about money, because time *is* money, and using wildcards can save you a *lot* of time.)

The point is, a wildcard *represents* something else. For example, the simplest wildcard in Word is the question mark (?), which represents any single character. If you want to see how it works, try this:

1. In the "Find what" box, enter a question mark (?).
2. Put a checkmark in the "Use wildcards" box. That tells Microsoft Word that you're going to search with a wildcard. If you didn't check the box, Word would assume you were trying to find a question mark.

Now click the "Find" button. Microsoft Word will find the first character after your cursor position. Click the "Find" button again. Microsoft Word will find the next character. And so on.

That doesn't seem very useful, but let's suppose you're editing a document that was scanned from a magazine article and is riddled with typos. You notice that the word "but" shows up in various ways, including "bat" and "bet." Let's say that this is a technical article with no references to baseball, winged mammals, or games of chance, so you decide to use the ? wildcard to find "bat" and "bet" and replace them in a single pass. Here's the procedure:

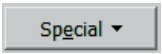
1. Enter "b?t" in the "Find what" box.
2. Enter "but" in the "Replace with" box.
3. Put a checkmark in the "Use wildcards" box.
4. Click the "Replace All" button.

Both "bat" and "bet" will be replaced with "but." The problem is, so will "bit." And, unfortunately, since you can't specify "Find Whole Words Only" when the "Use wildcards" box is checked, Microsoft Word will replace "better" with "butter," "combat" with "combut," and who knows what else. So, instead of clicking the "Replace All" button, you should click the "Replace" button for each individual item as needed.

Now you begin to see the power—and the danger—of using wildcards. Like cut-throat poker, they are not for the faint of heart. But if you know what you're doing, they can be very useful. Unfortunately, they won't help much in the "Replace with" box. In fact (with one exception that we'll discuss later), you can't use

them there at all. Why? Because Word has no way of knowing what you want them to represent.

Let's say you want to find "but" and replace it with either "bet" or "bat," so you put "b?t" in the "Replace with" box and click the "Replace All" button. Word doesn't know whether you want to replace "but" with "bet" or "bat," so it just replaces it with the actual text "b?t." So, basically, the only thing you can use in the "Replace with" box is actual text or certain built-in codes. You can find out what wildcards are available in the "Find what" box by clicking our old friend, the "Special" button, when the "Use wildcards" option is selected:



Here's what you'll get:

Any Character	?
Character in Range	[-]
Beginning of Word	<
End of Word	>
Expression	( )
Not	[!]
Num Occurrences	{ , }
Previous 1 or More	@
0 or More Characters	*
Tab Character	
Caret Character	
Column Break	
Em Dash	
En Dash	
Graphic	
Manual Line Break	
Page / Section Break	
Nonbreaking Hyphen	
Nonbreaking Space	
Optional Hyphen	

Items like "Em Dash" and "En Dash" are self-evident and will give you one of Word's built-in codes (discussed earlier). But the first nine items on that list are wildcards, pure and simple. Here's what they find:

?	Finds any single character: "c?t" finds "cat," "cut," and "cot."
*	Finds any string of characters: "b*d" finds "bewitched," "bothered," and "bewildered."
[ ]	Finds one of the specified characters: "b[ai]t" finds "bat" and "bit" but not "bet."
[-]	Finds any single character in the specified range (which must be in ascending order): "[l-r]ight" finds "light," "might," "night," and "right" (and "oight," "pight," and "qight," if they exist).
[!]	Finds any single character except those specified: "m[!u]st" finds "mist" and "most" but not "must." "t[!ou]ck" finds "tack" and "tick" but not "tock" or "tuck."
[!x-z]	Finds any single character except those in the specified range: "t[!a-m]ck" finds "tock" and "tuck" but not "tack" or "tick."
{n}	Finds exactly n occurrences of the previous character or expression: "re{2}d" finds "reed" but not "red."
{n,}	Finds at least n occurrences of the previous character or expression: "re{1,}d" finds "red" and "reed."
{n,m}	Finds from n to m occurrences of the previous character or expression: "10{1,3}" finds "10," "100," and "1000."
@	Finds one or more of the previous character or expression *before* something else: "me@t" finds both "met" and "meet"; "me@" (without the "t") finds only "me" because nothing comes after it.
<	Finds the beginning of a word: "<inter" finds "interest" and "interrupt" but not "splinter."
>	Finds the end of a word: "in>" finds "in" and "main" but not "inspiring."

Don't just skip over that list. Please study it carefully and try to understand what it's saying. You'll need that knowledge as we look at some examples of wildcards in use, which should also help clarify things.

Searching with Wildcards

Earlier we used the "?" wildcard to find every three-letter combination starting with b and ending with t—"bet," "but," "bit," "bat," and so on—by searching for "b?t" with "Use wildcards" turned on.

Now let's say we wanted to find the same characters but add others as well. For example, we might want to find every three-letter combination starting with b and ending with d—"bed," "bud," "bid," "bad," and so on—in addition to the combinations ending in t. Can we really do that? Sure! We'll start by entering the letter *b* into the "Find what" box, telling Microsoft Word to find that letter.

Next, we'll enter the ? wildcard, which tells Microsoft Word to find any single character.

Finally, we'll enter a new wildcard:

```
[td]
```

Microsoft Word will find any one of the characters specified in the brackets (in this case, either *t* or *d*). Characters specified in this way are called a "range."

Altogether, the string of characters looks like this:

```
b?[td]
```

That particular combination tells Microsoft Word to find the letter *b* followed by any other single character followed by *t* or *d*.

How can something like this help you? Suppose you're editing a manuscript in which the author has misspelled a name in nearly every way possible. You could comb through the manuscript over and over, hoping to catch all the variations. Or, you could be sure to catch them all by searching with wildcards. For example, let's say your manuscript is a book about India and the name in question is Gandhi. Your author has misspelled it as

"Ghandi," "Gahndi," and "Ganhdi." (Not possible? Hah!) You can find every last one of them with the following string:

```
G[andh][andh][andh][andh]i
```

Then, if you've put the correct spelling, "Gandhi," in the "Replace with" box, you can find and replace each wrong spelling with the right one in a single pass, which is much more efficient than finding and replacing each variation separately.

You may be wondering why you couldn't just use the * wildcard to represent the whole string of letters, like this:

```
G*i
```

You could. But the * wildcard represents any string of characters—including spaces and carriage returns. It's not limited to characters within a word (and neither are other wildcards). That means, in addition to finding the misspelled names, it will find the first 14 characters of the following phrase: "Go to the officer's hall." So be careful, especially if you're planning to use "Replace All" rather than finding and replacing one item at a time.

There is a way to simplify the wildcard combination, however. Consider this string:

```
G[andh]{4}i
```

It's functionally the same as this:

```
G[andh][andh][andh][andh]i.
```

The {4} tells Word to find exactly four occurrences of the previous "expression," which is [andh].

But now a complication: Suppose that our slapdash author has also spelled Gandhi's name as "Gandi." Uh-oh. Our original string won't catch that, because this new misspelling is one character shorter than our string specifies. But consider this:

```
G[andh]{3,4}i
```

The {3,4} tells Word to find from 3 to 4 occurrences of the previous expression, so this string will catch all of our misspelled variations so far.

What if we want to allow for more or fewer characters, being particularly unsure of our author? We can use this string:

```
G[andh]@i
```

The @ wildcard tells Microsoft Word to find one or more occurrences of the previous expression (if there are any) until it reaches an *i*. That ought to cover nearly anything our author throws at us. If we want to get a little more specific, we can use {3,}, which tells Word to look for at least three occurrences of the previous expression.

Here's a tip: What would happen if we put a lowercase g rather than a capital G at the beginning of our string? Word wouldn't find the misspelled names. Why? Because with "Use wildcards" turned on, Word automatically matches case—a useful thing to know.

Wildcard Ranges

The example above introduced wildcard ranges, which are actually fairly simple. You just use the [-] wildcard to tell Microsoft Word what to find. Let's continue with our example:

```
b?[td]
```

As you probably recall, this tells Word to find the letter *b* followed by any single character followed by either *t* or *d*. In other words, it will find "bet," "but," "bit," "bat," "bed," "bud," "bid," "bad," and so on.

But what if we wanted to find "bat," "bad," "bet," and "bed" but *not* "bit," "bid," "bud," and "but"? We could use this wildcard combination in the "Find what" box:

```
b[a-e][td]
```

This tells Word to find the letter *b* followed by any letter from *a* to *e* (in other words, *a*, *b*, *c*, *d*, or *e*) followed by *t* or *d*. (The range

must be in ascending order—in other words, from a "lower" letter [such as a] to a "higher" letter [such as *z*].)

Here's another way to approach this:

```
b[!f-z][td]
```

Notice the exclamation mark at the front of the "range" wildcard. The exclamation mark tells Word to find every character *except* those specified—in this case, the letters *f* through *z*. This wildcard combination, too, will find "bat," "bad," "bet," and "bed" but not "bit," "bid," "bud," and "but."

Here's a range that I use all the time:

```
[0-9]
```

That little beauty finds any occurrence of a digit. What's that good for? Let's say you're editing a document with lots of numbered lists, like this:

1. Lorem ipsum dolor sit amet.
2 Ut wisi enim ad minim veniam.
3. Duis autem vel eum iriure dolor.

Did you notice that the number *2* has no period? Good! You must have "the eye." But if you have several long lists, you might want to let Word find these problem numbers for you. To do so, try this wildcard string:

```
^013[0-9]@[!.]
```

Pretty cryptic. But if you've been following along, you can probably figure this out:

^013 is the numeric code for a carriage return.

[0–9] represents any digit (yes, it's a range—of numbers, in this case).

@ tells Word to find one or more occurrences of the previous expression, if there are any (in this case, any digit). This is necessary in case you have lists with two-digit (or longer) numbers.

[!.] tells Word to find any character *except* a period (also a range).

Piece of cake! Here are three other wildcard ranges you might find useful:

[a-z] represents any occurrence of a lowercase letter.
[A-Z] represents any occurrence of an uppercase letter.
[A-z] represents any occurrence of any letter.

Remember, too, that you can use the [ ] wildcard (without a hyphen) to specify a whole group of arbitrary characters. For example, this wildcard will find various kinds of punctuation:

```
[.,;:\?\!]
```

Don't be afraid to try all of these wildcard combinations and ranges for yourself (on some junk text, of course). As you experiment, you'll better understand what works and what doesn't. Then, when the need to use wildcards arises (which it will), you'll be ready.

Using a range to find Unicode characters

If you need to find *any* (unspecified) Unicode character in a document, you can use this *not* range:

```
[!^000-^255]
```

127 is the upper limit on ASCII characters. 255 is the upper limit on ANSI characters. This string excludes them both, so anything else it finds must be Unicode.

"ESCAPING" WILDCARDS

You may be wondering about the backslash (\) in front of the question and exclamation marks in our previous example:

```
[.,;:\?\!]
```

The backslash tells Word to treat the following character *as a character* and not as a wildcard. (Remember, ? is the wildcard for a single character, and ! is the wildcard for "except.") Using a backslash in this way is called "escaping" the character it precedes.

Here's a specific example. Let's say we want to find any question that appears in our document:

Where have all the flowers gone?

You might think the following wildcard string would do the job, but it doesn't:

[A-z ?]{1,}

Why? Because Word sees ? as a wildcard representing "any character." To make Word see it *as a character* instead (an actual question mark), you have to escape it with a backslash, like this:

\?

So this string *does* work to find any question:

[A-z \?]{1,}

There are actually quite a few characters that have to be escaped if you want to use them as characters rather than wildcards. Here they are, along with their meaning as wildcards:

? any character
* zero or more characters
[begins a range
] ends a range
{ begins a specified number
} ends a specified number
(begins an expression
) ends an expression
< begins a word
> ends a word
^ introduces a numeric character code
\ the escape character!

WILDCARD GROUPING

Wildcard grouping is simply a way of telling Word that you want certain wildcards to be used together as a unit. Continuing with our example from above, let's say that you're editing a document with lots of numbered lists, like this:

1. Lorem ipsum dolor sit amet.
2. Ut wisi enim ad minim veniam.
3. Duis autem vel eum iriure dolor.

Now let's say that you want to replace the space after each number and period with a tab. You could enter the following string of characters into the "Find what" box:

^013[0-9]@.^032

As you probably recall, that tells Microsoft Word to do the following:

Find a paragraph mark: ^013
followed by a number: [0-9]
followed by one or more numbers, if there are any: @
followed by a period
followed by a space

But that still won't let us replace that space with a tab. Why? Because there's no way to replace the space independently of the rest of the string—whatever the string finds *includes* the space.

So let's try this:

(^013[0-9]@.)(^032)

Notice that we've grouped the wildcards and other characters together with parentheses. (That's our uncooperative space between the last two parentheses.) Such groups, for reasons known only to the mathematically minded, are called "expressions," and in this case there are two of them:

1. (^013[0-9]@.)
2. (^032)

Grouping things together like this makes it possible to refer to each group independently in the "Replace with" box—a wonderful thing! So in the "Replace with" box, we'll enter this string:

```
\1^t
```

That "\1" is an example of the little-known "Find What Expression" wildcard, which lives deep in the wilds of Redmond, Washington, and only comes out at night. It's a backslash followed by the number one, and it tells Word to replace whatever is found by the first expression—

```
(^013[0-9]@.)
```

—with whatever the first expression finds. (Yes, you read that correctly.) In other words, Word replaces whatever the first expression finds with *itself*. That seems strange, but it means we can treat the second expression—

```
(^032)
```

—as an independent unit, which is exactly what we need to do. The ^t, of course, is the code for a tab.

You'll notice that we haven't included a "\2" code, which would replace something with whatever is found by our *second* expression, the space in the parentheses. Since we haven't included that code, the space will be replaced by nothing—in other words, it will be *deleted* during the find and replace. So the relationship between the wildcards in the "Find what" string and the "Replace with" string is something like this:

Find what:	*Replace with:*
(^013[0-9]@.)	\1 (followed by a tab: ^t)
(^032)	[nothing]

Now let's try using them:

1. In the "Find what" box, enter this: (^013[0-9]@.)(^032)
2. In the "Replace with" box, enter this: \1^t
3. Click the "Replace All" button.

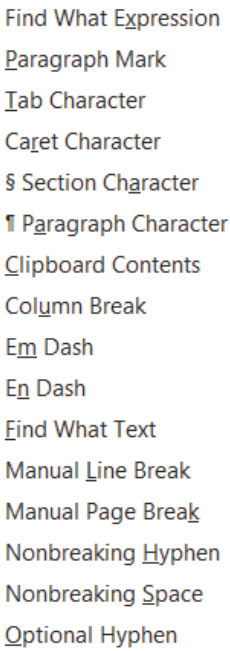
Presto! The spaces after your numbers will be replaced with tabs:

1.<tab>Lorem ipsum dolor sit amet.
2.<tab>Ut wisi enim ad minim veniam.
3.<tab>Duis autem vel eum iriure dolor.

To me, this is like magic, and it comes in handy more often than you might think. I hope you'll find it useful.

Using the "Find What Expression" Wildcard

In the previous example, I introduced the "Find What Expression" wildcard (\n), which is the only wildcard that can be used in the "Replace with" box. In fact, if you click the "Special" button while your cursor is in the "Replace with" box, you'll get this:



The first item on the list is the "Find What Expression" wildcard. Everything else will give you one of Word's built-in codes.

The "Find What Expression" wildcard is really important because it's so powerful. It's especially useful for moving things around. Let's say you've got a list of authors, like this:

Emily Dickinson
Ezra Pound
Willa Cather
Ernest Hemingway

And let's say you need to put last names first, like this:

Dickinson, Emily
Pound, Ezra
Cather, Willa
Hemingway, Ernest

You can use the "Find What Expression" wildcard to do this in a snap.

In the "Find what" box, enter this:

^013([A-z]@) ([A-z]@)^013

By now, you'll probably understand these codes and wildcards:

^013 represents a paragraph mark.

[A-z] represents any single alphabetic character, from uppercase A to lowercase z.

@ represents any additional occurrences of the previous character—in this case, any single alphabetic character, from uppercase A to lowercase z.

() groups [A-z]@ together as an "expression" representing an author's first name. (This grouping is the key to using the "Find What Expression" wildcard in the "Replace with" box.)

The space after the first ([A-z]@) expression represents the space between first name and last name.

The next ([A-z]@) group represents the author's last name.

The final ^013 represents the paragraph mark after the name.

Now, in the "Replace with" box, enter this:

^p\2, \1^p

The ^p codes represent paragraph marks. "Wait a minute," you say. "You just used ^013 for a paragraph mark. Why the change?"

Excellent question. The answer has two parts:

1. If we could use ^p in the "Find what" box, we would. But since Word won't let us do that when using wildcards (it displays an error message), we have to resort to the ANSI code, ^013, instead.

2. If we use ^p in the "Replace with" box, Word retains the formatting stored in the paragraph mark (a good thing). If we use ^013, Word loses the formatting for the paragraph (a bad thing). In a list of author names, this probably doesn't matter, but you'll need to know this when finding and replacing with codes in more complicated settings.

Continuing with our example:

^p\2, \1^p

\2 is the "Find What Expression" wildcard for our *second* expression (hence the 2) in the "Find what" box—in other words, it represents the last name of an author in our list.

The comma follows this wildcard because we want a comma to follow the author's last name.

A space follows the comma because we don't want the last and first names mashed together, like this: "Pound,Ezra."

\1 is the "Find What Expression" wildcard for our *first* expression (hence the 1) in the "Find what" box—in other words, it represents the first name of an author in our list.

Now click the "Replace All" button. The authors' names will be transposed:

Dickinson, Emily
Pound, Ezra
Cather, Willa
Hemingway, Ernest

You've always wondered how to do that, right? But now you're wondering about middle initials. And middle names. And Ph.D.s. All of those make things more complicated. But here, in a nutshell, are the find and replace strings you'll need for some common name patterns (first last, first middle last, first initial last, and so on). First comes the name pattern, then the Find string, and finally the Replace string, like this:

Name Pattern
Find What
Replace With

William Shakespeare
^013([A-z]@) ([A-z]@)^013
^p\2, \1^p

Alfred North Whitehead
^013([A-z]@) ([A-z]@) ([A-z]@)^013
^p\3, \1 \2^p

Philip K. Dick
^013([A-z]@) ([A-Z].) ([A-z]@)^013
^p\3, \1 \2^p

L. Frank Baum
^013([A-Z].) ([A-z]@) ([A-z]@)^013
^p\3, \1 \2^p

G. B. Harrison, Ph.D.
^013([A-Z].) ([A-Z].) ([A-z]@,) (*)^013
^p\3 \1 \2, \4^p

J.R.R. Tolkien
^013([A-Z].)([A-Z].)([A-Z].) ([A-z]@)^013
^p\4, \1\2\3^p

That list doesn't show every pattern you'll encounter, but it should provide enough examples so you'll understand how to create new patterns on your own—which is the whole point. Once you've created all of the patterns you need, you could record all of that finding and replacing in a single macro that you could run whenever you need to transpose names in a list. (Please see my book *Macro Cookbook for Microsoft Word.*)

Using Wildcards with ANSI Codes

Wildcards are powerful, but they do have a problem. If you're doing a wildcard search (you've put a check in the "Use wildcards" box), some of Word's built-in codes won't work in the "Find what" box. That means you'll have to use ANSI codes instead. For example, here's Word's built-in code to search for footnotes:

^f

If you try to use that code while searching with wildcards, you'll get an error message that ^f is not valid while using wildcards. So instead of using ^f, you'll need to use the ANSI equivalent (which finds endnotes as well as footnotes):

^02

Here's a list of the built-in codes that do and don't work with wildcards, along with some wildcard and ANSI equivalents for those that don't:

Character	*Works with wildcards*	*Doesn't work with wildcards*	*Wildcard or ANSI equivalent*
Annotation Mark (comment)	^a		
Any character		^?	?
Any digit		^#	[0-9]
Any letter		^$	[A-z]
Caret character	^^		
Column break	^n		
Em dash	^+		
En dash	^=		
Endnote mark		^e	^02
Field		^d	^019
Footnote mark		^f	^02
Graphic		^g	^047
Line break	^l		
Manual page break	^m		
Nonbreaking hyphen	^~		
Nonbreaking space	^s		
Optional hyphen	^-		
Paragraph mark		^p	^013
Section break		^b	^012
Tab character	^t		
White space		^w	[^s^t^032]

Let's look at another example: Let's say that (for some reason) you're searching for "wh" followed by any other character (the wildcard for which is "?"), followed by a carriage return. In the Find dialog's "Find what" box, you enter this:

```
wh?^p
```

You put a check in the box labeled "Use wildcards" and click the "Find" button. And there's that doggone error message:

> ^p is not a valid special character for the Find What box or is not supported when the Use Wildcards check box is selected.

"Well then, how," you politely ask your computer, "am I supposed to find what I'm looking for?" As usual, it doesn't reply, but here's the answer anyway. In the "Find what" box, you enter this:

```
wh?^013
```

Ordinarily, you should use built-in codes like ^p and ^b. But when they don't work, you've got an alternative.

Finding carriage returns on a Macintosh

At least, on a PC you've got an alternative. On a Macintosh, even numeric codes don't always work with wildcards. In particular, the numeric code for a paragraph break won't work:

```
^013
```

But there is a solution: Use the ^013 but "escape" it with a backslash and treat it as a range with square brackets:

```
[\^013]
```

If you need to specify *not* a carriage return, use this:

```
[!\^013]
```

On a Mac, you may need to do the same thing with other ANSI codes in addition to ^013.

Chapter 5

Wildcards in the Real World

What's Your Handle?

When faced with a situation requiring a complex find and replace in Microsoft Word, many people have no idea even where to begin. If you're one of those people, here's the secret: Find the handle.

What do I mean by "handle"? Something your find and replace routine can grab onto to do what it needs to do. For example, I remember one particular 500-page manuscript that had no style formatting for its different text levels—something I'm sure your authors would *never* give you. Basically, the text looked like this (but there was a lot more of it, of course):

> This Is a Heading
>
> This is some text. And several paragraphs more.
>
> JML
>
> This Is a Heading
>
> This is some text. And several paragraphs more.
>
> ED
>
> This Is a Heading
>
> This is some text. And several paragraphs more.
>
> CBD

So there I am, badly needing styles to be applied and yet not wanting to do it by hand. The first thing I looked for was a handle—some regularly occurring pattern that I could find and then

replace with itself but now with a style applied. Since this author, like many authors, was utterly ignorant of the proper way to put line spacing in front of a heading (by modifying "space before" in the heading style), he'd inserted two extra carriage returns in front of every main heading—and nowhere else. There was my handle!

So I typed this into the "Find what" box:

```
^013^013^013(*)^013
```

And I typed the Find What Expression code, surrounded by carriage returns, into the "Replace with" box:

```
^p\1^p
```

After typing in my find and replace strings, I clicked the More button to display the other find and replace options. I clicked the Format button, then "Styles," and then "Heading 1" so the replaced text would be formatted with that style. I put a check in the "Use wildcards" checkbox. Then I clicked the "Replace All" button.

Ta-da! All of my main headings (and author attributions) were now formatted with the Heading 1 style.

So, how about those author attributions? There sure were a lot of them—each on its own line at the end of each short article. And each one was simply the author's initials—JML, ED, CBD, and the like. There was my handle—two or more capital letters preceded and followed by a carriage return.

In the "Find what" box I typed this:

```
^013([A-Z]{2,})^013
```

And in the "Replace with" box I typed this:

```
^p\1^p
```

Again, I clicked the Format button, then "Styles," and this time "Heading 2" so the replaced text would be formatted with that style. I made sure the check was still in the "Use wildcards" checkbox. Then I clicked the "Replace All" button, which formatted all of those authors' initials with the Heading 2 style.

The final thing I needed to style was the paragraphs between each occurrence of Heading 1 text and Heading 2 text. There were no obvious handles associated with that text, but it did have those styled headings above and below it. Could I use those for my handles? Yes, but first I'd need to mark them with some arbitrary codes. Why? Because there's no way to find Heading 1 *and* some text *and* Heading 2, all in one pass. So here are the searches (this time with "Use wildcards" turned *off*) that I used to mark those headings:

Find what:

Heading 1 formatting

Replace with:

^&<H1>

Find what:

Heading 2 formatting

Replace with:

<H2>^&

That left me with an <H1> code at the end of each Heading 1 (really, at the beginning of the paragraph following it) and an <H2> code at the beginning of each Heading 2. Excellent handles indeed!

My final step was to search for those codes and the text between them, removing the codes and styling the text as Body Text. Piece of cake:

Find what (with "Use wildcards" turned on):

\<H1\>(*)\<H2\>

Replace with (formatted with the Body Text style):

\1

And that did the job. I still had some cleanup to do (like eliminating double carriage returns), but by looking for the handles in the text I was editing, I was able to style a 500-page document in less than five minutes.

The next time you're faced with a similar chore, don't just slog through the document doing everything by hand. Instead, see if there are some handles that will let you automate the whole process. You won't always find them, but you'll find them often enough to make the effort well worth your while.

If you spend much time doing the kind of thing this article describes, you really should try my RazzmaTag program, which will automate a whole raft of complex find-and-replace operations (including formatting) over a whole raft of documents. You can learn more here:

http://www.editorium.com/razzmatag.htm

For something a little simpler to use, you might be interested in my MegaReplacer program:

http://www.editorium.com/14843.htm

More Real-Life Examples

You might be interested in seeing some of the wildcard combinations I've used in a few other editing projects. Maybe you'll find them useful too.

Example 1

One manuscript I worked on had lots of parenthetical references like this:

(Thoreau, *Walden*, p 10.)

You'll notice that there's no period after the p. To fix these references, I used the following string in Microsoft Word's "Find what" box:

p ([0-9]@.\))

That's an odd-looking thing with its double parentheses, but its meaning becomes clear when you consider that the first closing parenthesis represents the closing parenthesis of the reference. The backslash in front of it tells Word to treat it as a character rather than the end of a group "expression." So the whole string says this:

1. Find a *p* followed by a space.
2. Find, as a group, one or more digits followed by a period followed by a closing parenthesis.

I put this in the "Replace with" box:

```
p. \1
```

And that string says this:

1. Replace the *p* followed by a space with *p* followed by a period and a space.
2. Replace the rest of the "Find what" string (the group in parentheses) with itself.

When I was finished finding and replacing, the references looked like this:

(Thoreau, *Walden*, p. 10.)

Example 2

Here's another example:

(Genesis 8:26)

You'll notice that there's no period before the closing parenthesis. Wanting to fix these, I put this string in the "Find what" box:

```
([0-9]@:[0-9]@)\)
```

It says:

1. Find, as a group, any number of digits followed by a colon followed by any number of digits.

2. Find a closing parenthesis character.

I put this in the "Replace with" box:

\1.)

And that string says:

1. Replace the group with itself.
2. Replace the closing parenthesis with a period and a closing parenthesis.

When I was finished finding and replacing, the references looked like this:

(Genesis 8:26.)

"Why," you may be wondering, "did you have to use wildcards? Why didn't you just find a closing parenthesis and replace it with a closing parenthesis and a period, like this:

Find what:

)

Replace with:

.)

I couldn't do that because the manuscript had other parenthetical items (like this one) that didn't need a period. Using wildcards makes it possible to find exactly the items you want and ignore those you don't.

Example 3

The manuscript also had Bible references that looked like this:

II Corinthians
II John
II Kings

I wanted them to look like this:

2 Corinthians
2 John
2 Kings

I put this in the "Find what" box:

II ([A-Z])

That says:

1. Find "I" followed by "I" followed by a space.
2. Find any capital letter.

And I put this in the "Replace with" box: 2 \1 – which says:

1. Replace the "II" with a "2".
2. Replace the capital letter with itself.

Worked like a charm.

"Why," you ask, "didn't you just replace "II" with "2" throughout the manuscript rather than use wildcards?" Well, I could have. But I was also thinking about other entries like these:

I Corinthians
I John
I Kings

Obviously, I couldn't just replace "I" with "1" throughout the manuscript, so I used this in the "Find what" box:

I ([A-Z])

and this in the "Replace with" box:

1 \1

and that took care of the problem.

I hope you're beginning to see how powerful wildcards can be and how much time they can save while you're editing a

manuscript. Using wildcards, you can quickly fix repetitive problems that would take hours to correct by hand. I highly encourage you to try them, but I also urge you to back up your documents and experiment on some junk text before using wildcards in the "real world." Also, try finding and replacing items individually before replacing all of them globally. Then you'll know that the wildcards you're using actually do what you need to have done.

Adding Periods to Lists

A book I recently edited had lots of lists—with no terminal punctuation. The lists looked something like this:

1. Text of the first item
2. Text of the second item
3. Text of the third item

As I worked, I found myself jumping to the end of each line and typing in a period, like this:

1. Text of the first item.
2. Text of the second item
3. Text of the third item

Then this:

1. Text of the first item.
2. Text of the second item.
3. Text of the third item

And finally this:

1. Text of the first item.
2. Text of the second item.
3. Text of the third item.

After two or three lists, I realized how silly this was. The solution is elementary:

1. Select all the items in your list, including the paragraph mark on the final item.

2. Bring up the "Find and Replace" dialog.
3. In the "Find what" box, enter this (the code for a paragraph mark): ^p
4. In the "Replace with" box enter this (a period followed by the code for a paragraph mark): .^p
5. Click "Replace All."
6. If Word asks if you want to search the rest of your document, click No; all you want to search is the list you selected.

That should do the trick. And, of course, if you want to remove periods rather than add them, follow the same procedure but swap the contents of the "Find what" and "Replace with" boxes.

Two-Step Searching

While editing in Microsoft Word, I often need to find something that's *partially* formatted and replace it with something else. For example, let's say a manuscript has a bunch of superscript note numbers preceded by a space that's *not* in superscript. Here's an example:

> Lorem ipsum dolor sit amet. [1]

I'd like to have Word find all such spaces and replace them with nothing (in other words, delete them), so that the result looks like this:

> Lorem ipsum dolor sit amet.[1]

Unfortunately, that doesn't seem possible. I can open set the "Find what" box to superscript, but the *space* isn't superscript, and the manuscript has thousands of spaces that *don't* precede a superscript number. It also has numbers that aren't superscript (like "2015"), so I can't just find spaces preceding numbers. What's an editor to do?

Find and replace the spaces in two steps rather than one:

1. Mark the superscript with codes.
2. Delete the spaces and codes.

Step 1

To mark the superscript with codes, do this:

1. Put your cursor in the (otherwise empty) "Find what" box.
2. Click the "Format" button.
3. Click "Font."
4. Put a checkmark in the "Superscript" box.
5. Click the "OK" button. The "Find what" box should now be set to superscript.
6. Put your cursor in the "Replace with" box.
7. Type the following string in the "Replace with" box:

   <S>^&

Now click "Replace All." Superscript numbers will be replaced with themselves, preceded by <S> (which code I just made up to indicate superscript). In other words, your sentences will now look like this:

> Lorem ipsum dolor sit amet. <S>1

Feel free to make up your own codes for whatever you need (italic, bold, paragraph styles, and so on). I frequently use %%% and ~~~ as code delimiters because they don't need to be "escaped" when finding them with wildcards, as angle brackets do.

The other code in the "Replace with" box, ^&, is Microsoft Word's "Find What Text" code (discussed earlier), which represents the text that was found (the superscript numbers).

Step 2

To delete the spaces and codes, do this:

1. Type "^032<S>" in the "Find what" box.
2. Click the "No Formatting" button so you're no longer finding superscript, which is now represented by the <S> code.
3. Put your cursor in the "Replace with" box and make sure the box is empty.

4. Click "Replace All."

All of the spaces in front of the codes (and thus in front of the superscript numbers) will be deleted, as will the codes themselves, leaving your sentences looking like this:

Lorem ipsum dolor sit amet.[1]

You can use this little two-step trick any time you need to find and replace partially formatted text. Now that you know how, that will probably be quite often.

Having said that, however, I can see a possible problem with this approach to fixing note references. What if your manuscript includes *other* superscript items in addition to note references? If it does, you may not want to remove the spaces in front of those items. The solution is is simple: rather than searching for superscript (in a two-step approach), just search for a space in front of a "grouped" note reference number, like this:

^032(^02)

Replace whatever was found with this:

\1

But if you want to do a wildcard search with footnotes (^f) or endnotes (^e) *only*, you're out of luck; Word won't let you. But again, here's where two-step searching can come in handy. For example, to remove spaces in front of footnotes alone, you could search for this:

^032^f

And replace with this (using the "Find What Text" code), which will put a percent sign in front of the space:

%^&

Then search for this:

%^032

And replace it with nothing (thus deleting the percent sign *and* space.)

Finding "Whole Words Only" with Wildcards

Microsoft Word won't let you specify "Find whole words only" when the "Use wildcards" option is checked. This is more than an annoyance; sometimes you really *need* to be able to find whole words only while searching with wildcards.

One solution is to include a space before and after the words you're looking for. Of course, not every word begins or ends with a space; words are often preceded or followed by quotation marks, dashes, and other characters, which would require multiple searching and replacing.

That suggests another solution: Use a wildcard "group" that includes every possible character that might precede or follow a word. For example, if we were searching for the word "bet," we could use a group like this before the word in the "Find what" box:

```
[ "-_/]
```

That character range (preceding "bet") would find the following text:

bet [preceded by a space]
"bet
-bet
_bet
/bet

We'd need a similar group after the word:

```
[ .,;:\!"-_/]
```

That character range (following "bet") would find the following text:

bet [followed by a space]
bet.
bet,

bet;
bet:
bet!
bet”
bet-
bet_
bet/

So our entire “Find what” string would look something like this:

["-_/]bet[.,;:\!"-_/]

So far so good, but there ought to be an easier way. How about using a group to specify what *not* to find before and after the word we’re looking for—like this:

[!A-z]bet[!A-z]

That string tells Word to find the word “bet” preceded and followed by any nonalphabetic character, which would certainly omit “bet” as part of another word. If we wanted to find “bet” both capped and lowercased, we could use this string:

[!A-z][b,B]et[!A-z]

These approaches are clever, and they will certainly work. In some situations, they (or variations of them) may be the best way to go, which is why I’ve included them here. However, we also need to remember that Microsoft Word includes a wildcard code for “beginning of word” (<) and "end of word (>).”

So, if we needed to find the whole word “bet” in a wildcard search, we could put this in the “Find what” box:

<bet>

That string would find “bet” but not “better” or “sorbet”—in other words, it would find “bet” as a whole word only!

Using < and > is probably the most elegant (and the easiest) way to find whole words only while searching with wildcards.

Wildcard Searching with Tracked Changes

Have you ever put together a clever wildcard find and replace routine that you *know* should work, but when you run the routine, you end up with something unexpected? You do it all the time? So do I, but that's not quite what I meant. I'm thinking specifically about routines that use the "Find What Text" code or the "Find What Expression" code, discussed earlier.

Let's say you've got a document that has revision tracking turned on (Tools > Track Changes), and in that document is a numbered list, like this:

1. First
2. Second
3. Third

Let's also say you want to use a wildcard find and replace to change the list to this:

(1) First
(2) Second
(3) Third

You should be able to do it like this:

1. In the "Find what" box, enter this:

```
([0-9]@)(. )
```

2. In the "Replace with" box, enter this (with a space after it):

```
(\1)
```

3. Click the button labeled "Replace All."

But it won't work. What you'll get is a list that looks like this:

1() First
2() Second
3() Third

How frustrating!

The problem is a bug in Word's wildcard find and replace engine. The easy way around the problem is to turn off revision tracking *before* doing the find and replace.

If you *need* the changes to be tracked, however, you're in trouble. I know of one possible solution:

1. Keep a backup copy of your original document.
2. Do your find and replace with revision tracking turned off.
3. Use Review > Compare > Compare to mark the differences between the changed document and your backup copy.

Expert word whacker Hilary Powers offers the following advice:

> The Wildcard and Revision Tracking features do bad things to each other. Some simple replaces will work with tracking on, but it's hard to predict which ones are safe and which ones will scramble the new info. Before running any wildcard replace operation, it's best to save the file and then turn the tracking off. Run the replace, check to see if it worked, then *turn the tracking back on*.

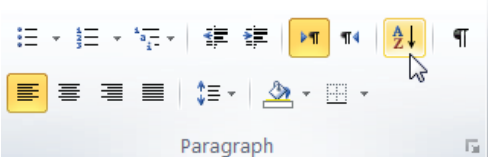
Deleting Duplicate Paragraphs

You can use wildcards to delete duplicate paragraphs, which comes in handy most often when working with lists, like this one:

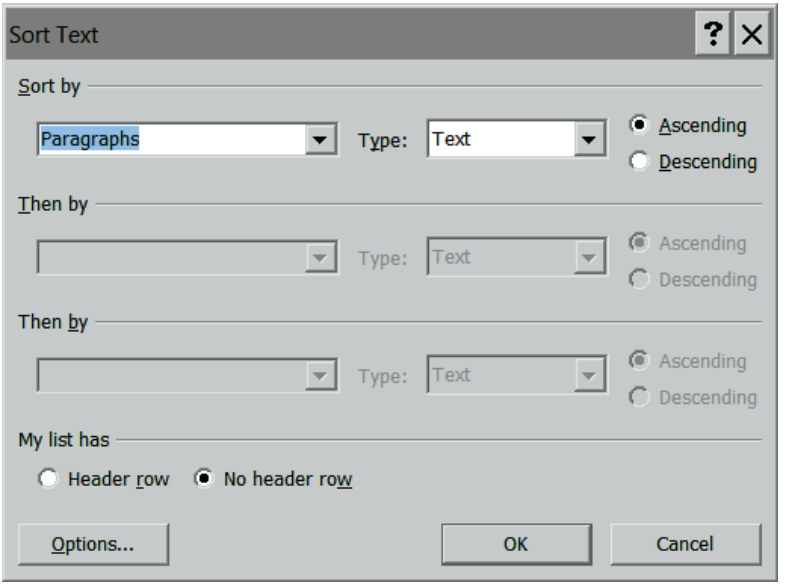
Jane Austen
Charles Dickens
Louisa May Alcott
Victor Hugo
Edgar Allen Poe
Jane Austen
Herman Melville
G. K. Chesterton
George Orwell
Ezra Pound
Jane Austen
Charles Dickens
T. S. Eliot
Victor Hugo

If that list were long, it would be difficult to spot any duplicated names, but a good start would be this:

1. Select all the names in the list.
2. On Word's ribbon interface, click Home > Paragraph > Sort:



Set the options to sort by paragraphs composed of ascending text with no header row. Then click the OK button:



The names will be sorted alphabetically, like this:

Charles Dickens
Charles Dickens
Edgar Allen Poe
Ezra Pound
G. K. Chesterton

George Orwell
Herman Melville
Jane Austen
Jane Austen
Jane Austen
Louisa May Alcott
T. S. Eliot
Victor Hugo
Victor Hugo

You could then scroll through the list and manually delete the duplicates, but a faster and easier way would be to use wildcards:

Find what:

^013([!^013]@)^013\1

Replace with:

^p\1

Here's what those mean:

Find a paragraph break: ^013
followed by any character that's not a paragraph break: [!^013]
as many times as necessary: @
with the previous two items grouped in parentheses: ([!^013]@)
followed by a carriage return: ^013
followed by whatever was found by the grouped expression: \1

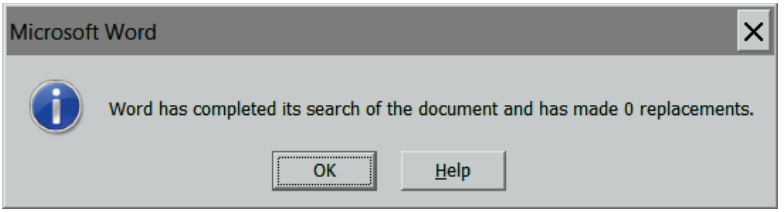
Then we replace whatever was found with this:

A carriage return: ^p
followed by whatever was found by the grouped expression: \1

In other words, we find a name, followed by itself, and replace both of those with the name.

You may need to run that find and replace several times to get all the duplicates. For example, in our list above, "Jane Austen"

is repeated three times. Running the find and replace one time will delete the first duplicate, but you'll need to run it again to get the second one. To make sure you get all the duplicates, run the find and replace over and over again until Word tells you it has "made 0 replacements":



At that point, the list will look like this, with all of the duplicates removed:

Charles Dickens
Edgar Allen Poe
Ezra Pound
G. K. Chesterton
George Orwell
Herman Melville
Jane Austen
Louisa May Alcott
T. S. Eliot
Victor Hugo

Considering Context

When working with wildcards, you always have to consider the context—in other words, *all* of the text you're working with, not just the text you're trying to find. If you don't, you may inadvertently find and replace something you don't *want* to find and replace.

Let's say we're editing a document with lots of numbered lists:

1. Lorem ipsum dolor sit amet.
2. Ut wisi enim ad minim veniam.
3. Duis autem vel eum iriure dolor.

But we want to replace the period and space after each number with a tab, so the lists look like this:

1 Lorem ipsum dolor sit amet.
2 Ut wisi enim ad minim veniam.
3 Duis autem vel eum iriure dolor.

What shall we search for? Let's use this:

^013[0-9]@.

Why the ^013 at the beginning? Because we *don't* want to find the number in something like this:

Columbus sailed in 1492. He sailed a long time.

The ^013 at the beginning of our find string ensures that only the numbers on lists will be found.

^013[0-9]@.

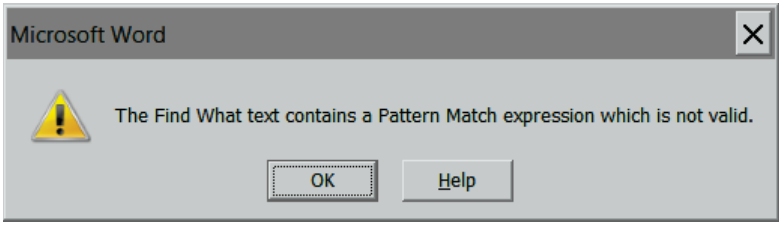
Again, you always have to consider the context in which you are working. What else is in the text that your wildcards might find but that you don't *want* them to find? Figure that out and then modify your wildcards to accommodate the problem. One of the best ways to do that is to repeatedly click the "Find" button and see what turns up. If you find something unexpected, it's time to fix your wildcard string so it *doesn't* find something unexpected. Once you're confident that your wildcards are working properly, it's probably safe to click the "Replace All" button. But what if you later find that your find and replace has royally messed something up? Now what? Well, that's why you should *always* back up your document before running a complex find and replace. Always, always, always.

Dealing with Errors

And then there are outright errors while using wildcards. Consider the following Find string:

^013[(0-9]@.)

If you try to use it, you'll get an error message:



Why is that? Can you see the problem?

That's right—the first square bracket and parenthesis are transposed. The string *should* look like this:

```
^013([0-9]@.)
```

So watch those brackets and parentheses! And if you start running into problems, don't panic. Instead, divide and conquer; break your string down into its component parts and test each part separately. For example, let's say we have a complex string like this one:

```
<[A-Z ]{1}[a-z]@.^032\([0-9]{2}:[0-9]{4}.\])
```

When we run it, we get that doggone error message, which means there's something wrong with our wildcard string. So first let's test this:

```
<[A-Z ]{1}
```

That works okay. Now let's add this and test again:

```
[a-z]@.^032
```

That also works okay. So we'll add this and test again:

```
\([0-9]{2}:
```

Still okay. We add the final bit and try again.

```
[0-9]{4}\])
```

Oops! This time we get the error message, so that's not okay. But at least we know that the problem is (probably) in the final part of the string, so we check that part carefully. Aha! See that closing parenthesis at the end of the string?

```
<[A-Z ]{1}[a-z]@.^032\([0-9]{2}:[0-9]{4}.\])
```

It shouldn't be there. We probably added it because there's an opening parenthesis in the *middle* of the string, which led us to think we we were creating a group. But we had previously "escaped" the opening parenthesis because we were looking for an actual parenthesis rather than using a parenthesis as a wildcard. Our string should actually have no group at all! We remove the closing parenthesis, and now our wildcard string works as it should:

```
<[A-Z ]{1}[a-z]@.^032\([0-9]{2}:[0-9]{4}.\]
```

When you use wildcards a lot, you'll run into stuff like this all the time. The solution is to slow down and test, test, test until you find the part of the string that's causing the problem.

Working with wildcards isn't really that difficult; it's just a matter of thinking logically and taking one step at a time. First get your ducks in a row:

Then knock 'em down, one after another (after another, after another) until everything is working exactly as it should.

Numbers by Chicago

Years ago I worked on a manuscript with lots of source citations that had page numbers formatted like this:

122-123

I prefer the shorter style recommended in the *Chicago Manual of Style* (8.69):

122-23

And besides, the manuscript was inconsistent, sometimes using one style, sometimes the other. Not wanting to fix all of these by hand, I decided to put the old wildcard search to work.

The first thing I needed to do was simplify things. Consider the style for even hundreds:

100-109
100-119
100-201

In all such cases, the numbers were already in the correct style, so I decided to just get them out of the way, like this:

Find what:

00-

Replace with:

~~-

(Those tildes are just arbitrary placeholders to be turned back to zeroes later.)

With that taken care of, I originally thought I could change all the other numbers like this:

Find what:

([0-9]{3}-)[0-9]([0-9]{2})

Replace with:

```
\1\2
```

That "Find what" string finds any set of three {3} numbers [0-9] followed by a hyphen, followed by a single number [0-9], followed by any set of two {2} numbers [0-9]. The items in parentheses are treated as as a group.

The "Replace with" string replaces the first parenthetical group with itself and the second parenthetical group with itself, leaving out any number [0-9] that was not grouped in parentheses.

That will definitely change 122-123 to 122-23, but it will also change 308-309 to 308-09, so we'll need to get a little fancier. How about this?

Find what:

```
([0-9]{3}-)[0-9]([1-9]{2})
```

Replace with:

```
\1\2
```

Notice that I've changed that last number range to [1-9] rather than [0-9]. That means numbers like 308-309 will not be found but numbers like 308-319 will. (Come to think of it, that single number in the middle could probably be [1-9] as well, since there shouldn't be any page numbers like 308-019. Of course, you never know.) Now, does that solve the problem?

Well, no. We still need to deal with numbers like this:

```
398-415
```

We certainly don't want that changing to 398-15. And what about this?

```
247-517
```

Unlikely, I'll admit, but still possible.

And that means we can't do our find and replace all in one shot. Instead, we'll have to do 18 specific searches:

(1[0-9]{2}-)1([1-9][0-9])
(2[0-9]{2}-)2([1-9][0-9])
(3[0-9]{2}-)3([1-9][0-9])
(4[0-9]{2}-)4([1-9][0-9])
(5[0-9]{2}-)5([1-9][0-9])
(6[0-9]{2}-)6([1-9][0-9])
(7[0-9]{2}-)7([1-9][0-9])
(8[0-9]{2}-)8([1-9][0-9])
(9[0-9]{2}-)9([1-9][0-9])
(10[1-9]-)10([1-9])
(20[1-9]-)20([1-9])
(30[1-9]-)30([1-9])
(40[1-9]-)40([1-9])
(50[1-9]-)50([1-9])
(60[1-9]-)60([1-9])
(70[1-9]-)70([1-9])
(80[1-9]-)80([1-9])
(90[1-9]-)90([1-9])

At least that's how it originally looked to me (but more below).

You can do the searches by hand if you like. You've got 20 chapters, all in separate files? Let's see—20 x 18 = 360 separate searches. Ouch! Of course, you could use my MegaReplacer program to do them all at once, freeing up your time for something more interesting:

http://www.editorium.com/14843.htm

Don't forget, we still need to turn those tildes back into zeroes:

Find what:

~~

Replace with:

00

Now all of those page numbers should be in Chicago style. How beautiful!

"What about four-digit numbers?" you ask. I leave it as an exercise for you to work out.

If you'd like this whole thing ready to run in MegaReplacer, here it is:

```
00-|~~-
(1[0-9]{2}-)1([1-9][0-9])|12+m
(2[0-9]{2}-)2([1-9][0-9])|12+m
(3[0-9]{2}-)3([1-9][0-9])|12+m
(4[0-9]{2}-)4([1-9][0-9])|12+m
(5[0-9]{2}-)5([1-9][0-9])|12+m
(6[0-9]{2}-)6([1-9][0-9])|12+m
(7[0-9]{2}-)7([1-9][0-9])|12+m
(8[0-9]{2}-)8([1-9][0-9])|12+m
(9[0-9]{2}-)9([1-9][0-9])|12+m
(10[1-9]-)10([1-9])|12+m
(20[1-9]-)20([1-9])|12+m
(30[1-9]-)30([1-9])|12+m
(40[1-9]-)40([1-9])|12+m
(50[1-9]-)50([1-9])|12+m
(60[1-9]-)60([1-9])|12+m
(70[1-9]-)70([1-9])|12+m
(80[1-9]-)80([1-9])|12+m
(90[1-9]-)90([1-9])|12+m
~~|00
```

Numbers by Chicago, Simplified

After perusing that lengthy find and replace routine to make sure inclusive (elided) numbers follow the style outlined in the *Chicago Manual*, astute reader Andrew Lockton responded with a really slick technique I hadn't previously known about. He suggested taking the "Find What Expression" wildcard, which takes the form \1, \2, and so on, and putting it *not* in the "Replace with" box, where it is ordinarily used, but in the "Find what" box—something I did not know was possible. Hats off to you, Andrew!

Andrew's discovery opens up all kinds of possibilities for various problems I had previously been unable to solve, but let's look specifically at getting numbers by Chicago. The previous method required 18 separate searches. Andrew's brilliant methodology requires only three. Here's the explanation:

1. Numbers that take the form 104-105 need to be converted to 104-5:

Find what:

```
([1-9])0([1-9])-\10([1-9])
```

Replace with:

```
\10\2-\3
```

What's going on there is that the first number grouping, ([1-9]), is being referred to by the \1 that follows the hyphen—in the "Find what" string. See it? Just before the 0 there? That tells Word to find (again) whatever was found by the first number grouping. For example, when the search hits something like "203-205," it says, "Hey, my first number group finds 2 [the first number in 203]. Let's see, is there also a 2 after the hyphen? Yes, there is!"

2. Numbers that take the form 104-110 need to be converted to 104-10:

Find what:

```
([1-9])0([1-9])-\1([1-9])([0-9])
```

Replace with:

```
\10\2-\3\4
```

3. Numbers that take the form 111-112 or 119-120 need to be converted to 111-12 or 119-20:

Find what:

```
([1-9])([1-9])([0-9])-\1([1-9])([0-9])
```

Replace with:

```
\1\2\3-\4\5
```

At first I thought it might be possible to combine 2 and 3:

```
([1-9])([0-9])([0-9])-\1([1-9])([0-9])
```

But that would also find even hundreds (100, 200), which need to be ignored (100-114 rather than 100-14).

Reader Jeanne Pinault suggested a different approach altogether:

> What I do with elided numbers is just replace all the hyphens with en dashes and then fix whatever comes up wrong when I edit the notes. That's because every set of endnotes I see is wrong in a slightly different way from every other set of endnotes I ever saw, so I have to read every character anyway. I can see that your marvelous find and replace would be a godsend with consistently formatted and voluminous endnotes produced on a regular basis, though. *Are en dashes in there someplace?*

Good question! I responded:

> In the Find string, use ^0150 (the en-dash code) instead of the hyphen.

On a Macintosh, you'd use ^0208.

Fixing Citations: We Can Do This the Easy Way, or We Can Do This the Hard Way

American Editor Rich Adin called me recently with a puzzle. He was editing a list of citations that looked like this:

> Lyon J, Adin R, Poole L, Brenner E, et al: blah blah blah.

But his client wanted the citations to look like this:

> Lyon J, Adin R, Poole L, et al: blah blah blah.

In other words, many of the citations included one author name too many; the client wanted a limit of three rather than four. And there were hundreds of citations. Rich really didn't want to remove the superfluous names by hand; it would have taken hours to do, and hours are money. And so, Rich queried, "Is there a way to remove the fourth name automatically?"

There's nearly always a way. Rich had already tried using a wildcard search, but without success. Microsoft Word kept giving him an error message:

> The Find What pattern contains a Pattern Match expression which is too complex.

The Too-Complex Find What

I'm not sure what wildcard search Rich tried to use, but it might have looked like this:

Find what:

```
([A-Z][a-z]@ [A-Z], )([A-Z][a-z]@ [A-Z], )([A-Z][a-z]@ [A-Z], )([A-Z][a-z]@ [A-Z], )(et al:)
```

Replace with:

```
\1\2\3\5
```

That's definitely too complex. Here's what it means:

> Find a capital letter: [A-Z]
> followed by a lowercase letter: [a-z]
> repeated any number of times: @
> followed by a space
> followed by a capital letter: [A-Z]
> followed by a comma
> followed by a space
> with all of that in parentheses to form a "group."

All of that is repeated three more times, then followed by "et al:" in parentheses to form a group.

The "Replace with" string tells Word to replace what it finds with the contents of groups 1, 2, 3, and 5—in other words, with the first *three* names followed by "et al:".

What's the Handle?

But Word is telling us that the search is too complex, which means we'll need to simplify. So we ask ourselves, "What, besides letters, do all of the names have in common?" In other words, "What's the handle? What can we grab onto?" Well, that's easy—each name is followed by a comma and a space. That's our handle!

The Find That Works

The handle means we can simplify our wildcard search string to something like this:

Find what:

([!^013]@, [!^013]@, [!^013]@,)[!^013]@, (et al:)

Replace with:

\1\2

Here's what that means:

Find any characters except a carriage return: [!^013]
repeated any number of times: @
followed by a comma
followed by a space
with all of that repeated three times
and enclosed in parentheses to form a "group."
Then it's repeated one more time, *ungrouped*
and followed by "et al:" in parentheses to form a group.

The "Replace with" string tells Word to replace what it finds with the contents of groups 1 and 2—in other words, with the first *three* names (group 1) followed by "et al:" (group 2). The fourth name is simply ignored.

To Group or Not to Group

Rich ran the new find and replace, then replied, "Thanks, Jack, that works like a charm. Why isn't the second 'group' grouped, that is, in parentheses? I thought that was necessary."

I replied, "No, it's not necessary. You group only the items that you want to reference (by \1, \2, etc.) in the 'Replace with' box. You *could* group the other item, in which case you would use '\1\3' in the 'Replace with' box. But there's no *need* to do so."

Note that this method of finding the names offers another advantage. Not only will it find names that look like this:

Lyon J,

it will also find names that look like this:

Lyon JM,

or even this:

Lyon JMQ

It will even find names like this:

Thaler-Carter Ruth,

Or like this:

Harrison G.B.H.,

In fact, it will find *anything* (except a carriage return) followed by a comma and a space.

Why the Carriage Return?

"Why," you may be wondering, "specify anything but a carriage return? Why not specify letters instead?" Well, we could have done that, using something like this:

Find what:

([A-z]@, [A-z]@, [A-z]@,)[A-z]@, (et al:)

Replace with:

\1\2

That means:

Find any capital or lowercase letter or space: [A-z]
repeated any number of times: @
followed by a comma
followed by a space

And so on.
Such a wildcard string would find names like this:

Lyon J,

but not this:

Thaler-Carter R,

Yes, we could add a hyphen to our string, but then we start to wonder about other characters we might need to include, and then things get complicated again. And besides, it's true that we *don't* want to include carriage returns in our search, so it makes sense to exclude them. If we tried to simplify too far, we might use this:

Find what:

(*, *, *,)*, (et al:)

Replace with:

\1\2

The problem with using the asterisk wildcard (*) is that it finds *any character any number of times,* including tabs, spaces, carriage returns, and everything else you can think of. Sometimes that's useful, but more often it just leads to confusion. We want to keep things simple but not *too* simple.

Why Wildcards?

To return to our original problem: Rich could have removed all those extra names one at a time, by hand, which is doing it the hard way and eats into the profit line—remember that time is

money. Microsoft Word includes powerful tools for doing things the easy way, so why not learn them and use them? If you've read this far, you're doing that, so congratulations.

Fixing Citations: The Easy Way, Not So Easy

After I solved Rich's citation problem, he wrote me again:

> As written, your wildcard find and replace reduces four names to three if "et al:" is the ending characteristic. How do you write it so that it can handle any number of names, say up to seven?

Good question, and a nice challenge for a wildcard search. Let's say we have citations with strings of names like this:

> Lyon J, Adin R, Carter TO, Jackson TT, Doe J, Smith K, Winger W, et al: blah blah blah

That's seven names, but let's see if we can make a wildcard string that will find any number of names and cut them down to three. My first impression is that this might be difficult or even impossible. But let's try the following wildcard string:

```
([!^013]@, ){3}([!^013]@, ){1,}(et al:)
```

Here's what that means:

```
Find any character except a carriage return: [!^013]
repeated any number of times: @
followed by a comma
followed by a space
and enclosed in parentheses to form a "group."
Do that three times in a row: {3}
Find using the same group: ([!^013]@, )
if it occurs once or more (as indicated by the comma): {1,}
followed by "et al:" in parentheses to form a group.
```

There's just one problem: It doesn't work. And that's how it often is with wildcards—sometimes you have to fiddle around to get the result you want; trial and error are key. So let's see if we can find just three instances of text using our group:

```
([!^013]@, ){3}
```

That doesn't work either. Let's try using the group three times in a row:

```
([!^013]@, )([!^013]@, )([!^013]@, )
```

That *does* work. So why not this?

```
([!^013]@, ){3}
```

Specifying a number (like {3}) has always worked in other situations, so why not here? Let's try using the same wildcard string to find the first three names if the names are all identical (yes, weird, but we're testing here):

```
Lyon J, Lyon J, Lyon J, Lyon J, Lyon J, Lyon J, Lyon J, et al:
blah blah blah
```

Sure enough, that works! What in the world is going on?

Aha, aha! I've got it! It's the @ that's causing this weird behavior. We haven't provided a *specific* string of characters for {3} to find; instead, the @ grabs the characters in front of the comma, and *they become* the specific string of characters that {3} tries to find (three times in succession). That specific string of characters doesn't *exist* three times in succession in our original citation, so naturally Word can't find it three times in succession. Whew!

Well, okay, then. For our current purposes, we'll stop using numbers (such as {3}) to specify how many times our pattern should be repeated. Let's try this instead:

```
([!^013]@, [!^013]@, [!^013]@, )[!^013]@(et al:)
```

Here's what that means:

```
Find any characters except a carriage return: [!^013]
repeated any number of times: @
followed by a comma
followed by a space
repeated three times
and enclosed in parentheses to form a "group."
```

Then find any character except a carriage return: [!^013]
repeated any number of times: @
followed by "et al:" in parentheses to form a group.

Well, son of a gun; that actually works. So now we can use the following in the "Replace with" box:

\1\2

Here's what that means:

Replace everything that was found
with the text represented by group 1: \1
followed by the text represented by group 2: \2

Group 1, you'll remember, was this:

([!^013]@, [!^013]@, [!^013]@,)

It finds the first three names in our citations. And group 2 was this:

(et al:)

It finds the end of our citations.

And so, finally, we've succeeded in fulfilling Rich's original request:

As written, your wildcard find and replace reduces four names to three if "et al:" is the ending characteristic. How do you write it so that it can handle any number of names, say up to seven?

Sometimes the easy way isn't so easy. Nevertheless, it's almost always worth pursuing. In Rich's case, it reduced his editing time from hours (removing extraneous names by hand) to minutes (removing the names with a wildcard find and replace). It also gave Rich a wildcard search that he can save in his fabulous EditTools software for use with future projects. And it provided a deeper and clearer understanding of how to use wildcard searches.

After all these years of editing, wildcard searching is the tool I rely on the most. I encourage you to invest the time needed to learn to use this tool, which will repay your efforts many times over.

Fixing Citations: From Easy to Impossible—Three Variations on a Theme

After that, Rich Adin sent one more request, which is okay because I appreciate a good challenge:

> Okay, Jack, you solved the problem of reducing the number of authors from more than three down to three. But there is a caveat: the list of names needs to end with "et al:". So let me pose three more variations.

Three?! Oh, all right. Here we go:

Variation 1

> How do I handle instances where the ending is punctuation other than "et al:"? For example, it could be a different punctuation mark than the colon or it could end with an author name and not "et al" (e.g., "Lyon J, Adin R, Carter TO, Jackson TT, Doe J, Smith K, Winger W:" or "Lyon J, Adin R, Carter TO, Jackson TT, Doe J, Smith K, Winger W, Hoffnagle TTP.")

How do we handle instances where the ending is punctuation other than "et al:"? Here are Rich's examples, all laid out for our inspection:

> Lyon J, Adin R, Carter TO, Jackson TT, Doe J, Smith K, Winger W:
> Lyon J, Adin R, Carter TO, Jackson TT, Doe J, Smith K, Winger W, Hoffnagle TTP.

As discussed earlier, the key is to find the "handle," the unique elements we can grab to carry out our search. In Rich's examples, the "handles" would have to be the colon that ends the first entry and the period that ends the second. Let's try modifying the wildcard string that we used earllier:

```
([!^013]@, [!^013]@, [!^013]@, )[!^013]@([:.])
```

Here's what that means:

```
Find any characters except a carriage return: [!^013]
repeated any number of times: @
followed by a comma
followed by a space
repeated three times
and enclosed in parentheses to form a "group."
Then find any character except a carriage return: [!^013]
repeated any number of times: @
followed by [:.] (specifying a colon or a period) in parentheses
to form a group.
```

And we can use the following in the "Replace with" box:

```
\1\2
```

Here's what that means:

```
Replace everything that was found
with the text represented by group 1: \1
followed by the text represented by group 2: \2
```

But does that actually work? Well, sort of, Here's what we get:

```
Lyon J, Adin R, Carter TO, :
Lyon J, Adin R, Carter TO, .
```

Maybe that's close enough, as it would now be an easy matter to search for comma space colon and replace it with a colon, and to search for comma space period and replace it with a period. But if we want to refine our search string even further, we could use this:

```
([!^013]@, [!^013]@, [!^013]@), [!^013]@([:.])
```

Here, we've placed the comma and space following the third name outside the parenthetical group, so they're not included

when the group is replaced by /1. That actually solves the problem, if you want to get precise, giving us a result like this:

> Lyon J, Adin R, Carter TO:
> Lyon J, Adin R, Carter TO.

Variation 2

Rich wrote:

> How can I revise the string to work even if there is no consistency in punctuation of names? For example, suppose the names are: "Lyon, J, Adin R, Carter T.O., Jackson TT, Doe, J.; Smith K; Winger, W; Hoffnagle TTP."

As given, this can't be done. Why? Because we've lost the uniqueness of the comma "handles" that separate the names. For example, instead of this—

> Lyon J,

—we have this:

> Lyon, J,

And instead of this—

> Smith K,

—we have this:

> Smith K;

So again, as given, we can't fulfill Rich's request. But can we change the "as given"? Why, yes, we can! (And that makes this another example of two-step find and replace. Well, three, really. If circumstances won't let you do what you need, you can change those circumstances. This is often true in life as well as Microsoft Word.)

So let's search for a lowercase letter followed by a comma (after a last name) and replace it with just the lowercase letter:

Find what:

```
([a-z]),
```

Replace with:

```
\1
```

Then we can search for a semicolon (which sometimes follows initials) and replace it with a comma:

Find what:

```
;
```

Replace with:

```
,
```

Finally, we can use the same wildcard string we used earlier to fulfill Rich's request:

Find what:

```
([!^013]@, [!^013]@, [!^013]@), [!^013]@([:.])
```

Replace with:

```
\1\2
```

You may be wondering if this will affect article titles and journal names and not just author names. The answer is, it depends. I'm assuming that article titles and journal names don't include commas (just for purposes of illustration). But if they do, you may have to get creative. Here's an example:

> Levy, D, Ehret G, Rice K, Verwoert G, Launer L, Dehghan A, Glazer N, Morrison A, Johnson A, Aspelund T, Ganesh S, Chasman D: Genome-wide association study of blood pressure, stress, and hypertension. *Nature* 2009, 41(6): 677-687.

See that comma after "Levy"? Above, we got rid of it with the following strings:

Find what:

```
([a-z]),
```

Replace with:

```
\1
```

But notice that this will also remove the commas after "pressure" and "stress" in the article title, which we don't want to do. The solution, again, comes down to handles. What do we have that sets off the article title and journal name? In this example, they're preceded by the colon after the author names ("Chasman D:") and followed by a carriage return (at the end of the citation). So here's a rather sneaky solution: Search for a colon followed by anything that isn't a carriage return until you come to a carriage return. Then replace whatever was found with itself (^&) formatted as Hidden:

Find what:

```
:[!^013]@^013
```

Replace with (using Hidden formatting):

```
^&
```

If you don't know how to replace using formatting, here's the secret:

1. Put your cursor in the "Replace with" box.
2. Click the "More" button if it's showing.
3. Click the "Format" button on the bottom left.
4. Click "Font."
5. Put a check in box labeled "Hidden."
6. Click the "OK" button.

As explained earlier, you can replace with all kinds of formatting: styles, paragraph alignment, and so on. You can also use formatting in the "Find what" box! This is really powerful stuff, and if you didn't know about it before, now you can add it to your bag of tricks.

At any rate, with the article titles and journal names formatted as Hidden, you can make sure they actually *are* hidden by clicking the "Show/Hide" button (with the pilcrow icon: ¶) on Word's "Home" tab. Then run your find and replace to remove commas from last names:

Find what:

([a-z]),

Replace with:

\1

Finally, unhide the article titles and journal names (after using "Show/Hide" to display them):

Find what:

(Hidden formatting)

Replace with:

(Not Hidden formatting)

At that point, the commas will be gone from the authors' last names but preserved in the article titles and journal names.

Variation 3

Rich wrote:

> How can I adapt the wildcard string to delete those in excess of a certain number? I have a client who wants up to ten author names listed and "et al" used only for names eleven and following. I'd like to specify how many names I want retained and replace the excess with "et al." For example, if there are fifteen names, delete the last five if ten are okay and replace them with "et al."

Theoretically, we could do that as long as there's a "handle" that marks the end of the names. Let's take this example:

> Levy D, Ehret G, Rice K, Verwoert G, Launer L, Dehghan A, Glazer N, Morrison A, Johnson A, Aspelund T, Ganesh S, Chasman D: Genome-wide association study of blood pressure and hypertension. *Nature* 2009, 41(6): 677-687.

There are actually twelve names there, so we want to keep the first ten and replace the last two with "et al." What's our handle? The colon after the last name ("Chasman D:") and before the article's title. So let's try an expansion of the wildcard search string we used earlier. Instead of grouping three comma-separated names, we'll group ten:

Find what:

([!^013]@, [!^013]@, [!^013]@, [!^013]@, [!^013]@, [!^013]@, [!^013]@, [!^013]@, [!^013]@,)[!^013]@(:)

Replace with:

\1 et al.\2

That would work if Word could handle it. But if you try it, Word will complain:

> The Find What text contains a Pattern Match expression which is too complex.

So now what? Honestly, I'm not sure. I tried several other possibilities, none of which were successful. So if you, Gentle Reader, have any ideas about how to accomplish this seemingly impossible feat, I'd love to hear them.

Wildcard searching can't do everything, but it can do an awful lot. As I've said before, after all these years of editing, wildcard searching is the tool I rely on the most. I encourage you to invest the time needed to learn to use this tool, which will repay your efforts many times over.

Wildcard Dictionary

Even though I use wildcard searches all the time, I don't do a very good job of saving my wildcard entries so I can use them again. But really, I should save them all in a "wildcard dictionary" that would include entries in the following format:

- A description of what the find and replace wildcard strings do.
- The wildcard find and replace strings themselves.
- Some key words I can search for if I'm looking for wildcard strings for a certain purpose.
- Before-and-after examples of what the wildcard strings do.
- Other comments.

Here's an example, with a wildcard string you may be able to use:

> *Description:* Find parenthetical publishing information in source citations and replace it with nothing to help in changing citations to short form.
> *Find What:* ^032\([A-z ,.]@:[!)]@[0-9]{4}\)
> *Replace With:* [nothing]
> *Key Words:* publishing information, source, citation, footnotes, endnotes, books, parentheses, long, short, delete
> *Before:* Jack M. Lyon, *Total Word Domination* (Edina, Minn.: PocketPCPress, 2001), p. 237.
> *After:* Jack M. Lyon, *Total Word Domination*, p. 237.
> *Comments:* Won't find or delete other parenthetical text.

When I suggested that idea in my newsletter, I asked readers to share their favorite wildcard strings. I've since heard from Rosalie Wells, Hilary Powers, Eric Fletcher, Allene Goforth, Michael Coleman, Mary L. Tod, and Steve Hudson, who sent some great wildcard strings and commentary on their use. Many thanks to all of them!

And now, the wildcard dictionary entries!

Rosalie Wells wrote:

I use this one all the time in my translations into Spanish to change the decimal separator "period" to a "comma" separator as required for many Spanish-speaking countries:

Find what: ([0-9]).([0-9])
Replace with: 1,2

Hilary Powers wrote:

I tend to design strings from scratch when needed, but here are a couple that I use often enough to more or less remember them:

.([A-Z])|. 1

opens up initials on reference lists; requires fixing things like U.S. and N.Y. after

([0-9]). |^t1.^t

indents hand-typed list numbers

[!.]^013 — review one by one and add periods by hand where needed. There's a way of scanning for more end-sentence punctuation and doing the change automatically, but I'm usually too lazy to look it up and this is what I remember. A complete punctuation scan would be quite welcome. . . .

Eric Fletcher wrote:

I have my favourites in various Word files I seem to never get around to consolidating. But here are a few I found without having to look very hard:

Description: Finding a telephone number formatted as 123-4567.
Find What: ([0-9]{3})(-)([0-9]{4})
Replace With:
Key Words: Telephone number
Before:
After:

Comments: This is handy for doing a quick review of phone numbers. In Word 2002, you can choose to select all occurrences so you can see them easily in context.

Description: Changing telephone numbers formatted as (123) 456-7890 or (123)456-7890 to 123-456-7890.
Find What: ([(])([0-9]{3})([)])(*)([0-9]{3})(-)([0-9]{4})
Replace With: 2-567
Key Words: Telephone number
Before: Telephone numbers formatted as (123) 456-7890 or (123)456-7890.
After: Telephone numbers formatted as 123-456-7890 or 123-456-7890.
Comments: Note that the (*) looks after catching situations where there may or may not be a space after the area code portion.

Description: Find formatted text and change it to use HTML codes.
Find What: Font=Italic
Replace With: ^&
Key Words: Italic, HTML, formatting
Before: Change the italicized words to use HTML codes.
After: Change the *italicized* words to use HTML codes.
Comments: If you include Font=Not italic in the Replace with, the italics will be removed as well. Use variations of this for any formatting and other HTML codes.

Description: Find text coded with HTML and change it to Word formatting.
Find What: *()(*)()*
Replace With: 2 Font=Italic
Key Words: HTML, italic, formatting
Before: Change the *italicized* words to regular Word formatting.
After: Change the italicized words to regular Word formatting.
Comments: Use variations of this for any formatting and other HTML codes.

When I recently referred someone to your tips about using wildcards in find and replace, it reminded me that I had intended to send you some tips about techniques I use frequently.

Multi-step F&R to Set Styles: I often need to rationalize formatting in jobs I do: setting styles, sometimes reducing the number of styles, frequently setting up styles for translation to html. I use the

wildcard features to simplify the process but sometimes need to do it in several steps. One "routine" saved an enormous amount of work in removing the typed numbers in ordered lists and setting the style from Normal to List number.

1. The first F&R used "(^013[0-9]@.)( )" in Find; "\1þ" in Replace and used wildcards. The result is a unique code in each of the ordered list paragraphs. (The pattern was from one of your tips. I choose to use þ (ALT + 0254) as a unique character for later steps.)

2. The second F&R used "þ" in both find and replace without wildcards and had Style=List Number set in the Replace with. This caused the lines that had been flagged as starting with the number-period pattern to be set in the desired style.

3. The final F&R used "[0-9]@.þ" in the Find and nothing in the Replace with no styles but with wildcards. This caused the numbers and my unique flag to be stripped, leaving the paragraphs formatted with the List Number style as intended. I use a similar technique for bullet lists. When the document is converted to html, having the appropriate styles set saves a huge amount of effort.

Use Character Styles to Flag Items: I find character styles very useful. When phrases or words need to be flagged for in-context review on-screen, I often use F&R to tag them with a special character style. Then, during review, I set the flag style definition to something that really stands out—a bold colour—so I can see it in a zoomed view of many pages at once. The items stand out and I can just click and scroll-zoom in to examine them. To finalize the document, I replace the style with the default character style to remove any remaining flagging. (The other advantage of this method is that I can find the flag style and rapidly skip from one to the next instance.)

Highlight Styles Temporarily: When I have assembled a set of documents for conversion to html, I use F&R to set the styles (above) whenever possible. Then, to be able to see the potential problems, I redefine the style colours so I can see them in the zoomed-out view. For example, by setting the List Number style to green type, I can easily spot all instances of ordered lists—and elements like bullet paragraphs within an ordered list stand out clearly. I used to set resolved styles in hidden text to make the document shrink as I dealt with it but it introduces too many problems with tabular material. The nice thing about this technique is that the style changes can be temporary: just reset the template to update the styles to their normal definition.

Allene Goforth wrote:

Here are five of my wildcard routines. I use more than those, but some are specific to various publishers, and others are of the half-baked variety.

Description: In APA-style references lists, find volume numbers in roman and change them to italic. Retain the issue numbers in roman.
Find What: , [0-9]{1,}
Replace With: [nothing]; change font to italic
Key Words: APA, references, volume numbers
Before: Developmental Neurobiology, 13(2)
After: Developmental Neurobiology, 13(2)
Comments: Find string includes a space between the first comma and the bracket.

Description: In APA-style references lists, find issue numbers in italics and change to roman.
Find What: ([0-9]@)
Replace With: [nothing]; change format to roman
Key Words: APA, references, issue numbers
Before: Developmental Neurobiology, 13(2)
After: Developmental Neurobiology, 13(2)

Description: In APA-style references lists, find initials in names that need a space inserted after the period.
Find What: ([A-Z].[!A-Z])
Replace With: 1
Key Words: APA, references, initials
Before: Brown, A.C.
After: Brown, A. C.
Comments: A space is needed at the beginning of the Replace string.

Description: In APA-style references lists, find name strings containing "&" that need commas inserted before the "&."
Find What: ([&])
Replace With: ,1
Key Words: APA, references, &, comma
Before: Smith, A. B. & Gordon, D. J.
After: Smith, A. B., & Gordon, D. J.

Comments: In the Find string, there is a space between the opening parenthesis and the bracket.

Description: Find and close up space between journal volume number and issue number in APA-style references lists.
Find What: (([0-9]@))
Replace With: 1
Key Words: APA, references, space, volume, issue
Before: 45 (3)
After: 45(3)
Comments: In the Find string there should be a space before the opening parenthesis. There should not be a space before the first character in the Replace string.

Michael Coleman wrote:

Right now I'm working on an index. There's not a lot of work to be done, but it was exported from Quark to Word, so all the formatting was stripped. (If there's a way to avoid that, I'd love to learn about it.) So I set styles for the four levels. Simple enough. The only other trick is to get back all of the italics. There are a few titles that need to be italicized, and fortunately I know that they all have names with at least three words, so I searched for a string

[A-Z]([a-z]@) [A-Z]([a-z]@) [A-Z]

I didn't make any automatic changes because several titles fit the string but don't get italicized.

We used to have a lot of tables, figures, and exhibits in our books, but now they're all called figures. In the index, the appropriate first letter—t, f, or e—appeared in italics after the page number, such as 11-11e. (We use chapter-page pagination.) So I searched for

([0-9])[e,f,t]

I set the replace string to italic and replaced with

\1f

Then I searched for ([0-9]) formatted as italic and changed it back to roman using \1.

Our old style was to use en dashes to show a range of pages, but that was hard to read because of the hyphens in the chapter-page pagination format. So we changed it to "to." I therefore searched for

^=([0-9])

And I replaced it with

to \1

Mary L. Tod wrote:

Here's a funky list of references in my current work. I've edited the first three to show you the style desired. I will attack these with a combination of wildcard searches and whittle it down.

Unedited references:

1. Dor V, Saab M, Coste P, Kornaszewska M, and Montiglio F, Left ventricular aneurysm: A new surgical approach. Thoracic and Cardiovascular Surgeon, 1989. 37: p. 11–19.
2. Burkhoff D and Wechsler AS, Surgical ventricular remodeling: a balancing act on systolic and diastolic properties. J Thorac Cardiovasc Surg, 2006. 132(3): p. 459–63.
3. Jones RH, Velazquez EJ, Michler RE, Sopko G, Oh JK, O'Connor CM, et al., Coronary Bypass Surgery with or without Surgical Ventricular Reconstruction. N Engl J Med, 2009: p. NEJMoa0900559.
4. Fung YC, Biomechanics: Motion, flow, stress, and growth. 1990, New York: Springer Verlag.

Edited references to show style desired:

1. Dor V, Saab M, Coste P, Kornaszewska M, Montiglio F. Left ventricular aneurysm: a new surgical approach. Thorac Cardiovasc Surg 1989;37:11–9.
2. Burkhoff D, Wechsler AS. Surgical ventricular remodeling: a balancing act on systolic and diastolic properties. J Thorac Cardiovasc Surg 2006;132:459–63.
3. Jones RH, Velazquez EJ, Michler RE, et al. Coronary bypass surgery with or without surgical ventricular reconstruction. N Engl J Med 2009;360:1705–17.
4. Fung YC. Biomechanics: motion, flow, stress, and growth. New York: Springer Verlag; 1990.

Here's the string I tried:

([!\^013]@, [!\^013]@, [!\^013]@,)[!\^013]@(et al.)

And that did the trick!

Steve Hudson wrote:

Remove Time stamping from most logs:
Find What: [[]*[]]
Replace With: nothing

Kill excessive blank paragraphs
Find What: ^p^p^p
Replace With: ^p^p

Locate some passive voice instances
Find What: be <*ed>

Convert a list of Firstname Lastname to Initial. Lastname
Find What: <(?)(*)> <(*)>
Replace With: \1. \3

Find manually formatted numbering (hand tweak)
Find What: [0-9]@.^t
Replace With: Pass 1 List style, pass 2, nil.

Turn straight quotes to curly quotes
Turn off AutoCorrect. then:
Find What: "
Replace With: "
Find What: '
Replace With: '

Change all numeric dates from DD/MM/YY(YY) to MM/DD/YY(YY) and back again
Find What: <([0-9]@[/.-])([0-9]@[/.-])([0-9]@>)
Replace With: \2\1\3

Sharon Key wrote asking why, after selecting smart quotes, her find and replace of quotes with themselves didn't change straight to curly. It's because the find and replace is not triggering the smart quote function. To do that, it needs something before or after the quote to help the system dope it out. It's actually triggered by an "end of word" condition. So use these:

Find What: "(<*>)
Replace With: "\1

Find What: ([!])"
Replace With: '1"

Both formulae use the () to force capture of that segment to be referred to in the replace section as \1 (or whatever left -> right position it holds if there are multiple bracketed entries).

The first finds quotes followed by a word (a < is a start of a word, * is anything, a > is the end of a word), and replaces the quote with the word (now referred to as \1 from being bracketed) after it. You can't use that same trick for the second, as it selects the whole string of words afore it, and the smart quote feature is confused as the last typed character was in a range. So, we find any non-blank character followed by a quote, and replace the single character and the quote. This will take care of most of your problems.

The easiest way to specify a special character (without having to "escape" it) is to use the ASCII code instead. For example, for an opening parenthesis, use ASCII code 40. (ASCII codes are specified in *any* sort of search with the caret ^.) Here are the ASCII codes you can use rather than remembering to escape other characters:

\ = ^92
(= ^40
) = ^41
? = ^63
{ = ^123
} = ^125
[= ^91
] = ^93
@ = ^64
< = ^60
> = ^62
* = ^42
^ = ^94

Here are some other handy wildcard searches:

10{1,3} finds "10", "100", and "1000".
[10]@ finds any binary number
<[a-zA-Z]{1,3}> finds words of three letters or less.
<[A-Z][a-z]@> finds any title-cased word.
<[0-9]@> finds any whole number, <[0-9]{1,3}> from 0-999

Reference

Built-in Codes

Character	*Find What*	*Replace With*
Annotation Mark (comment)	^a	
Any character	^?	
Any digit	^#	
Any letter	^$	
Caret character	^^	^^
Clipboard contents		^c
Column break	^n	^n
Contents of the Find What box		^&
Em dash	^+	^+
En dash	^=	^=
Endnote mark	^e	
Field	^d	
Footnote mark	^f	
Graphic	^g	
Line break	^l	^l
Manual page break	^m	^m
Nonbreaking hyphen	^~	^~
Nonbreaking space	^s	^s
Optional hyphen	^-	^-
Paragraph mark	^p	^p
Section break	^b	
Tab character	^t	^t
White space	^w	

ANSI Character Codes

Character	*Name*	*Macintosh*	*PC*
	footnote reference	2	2
	tab	9	9
	line break	11	11
	page/section break	12	12
	paragraph break	13	13
	column break	14	14
-	nonbreaking hyphen	30	30
–	optional hyphen	31	31
	space	32	32
,	comma	226	130
ƒ	folio	196	131
„	double comma	227	132
. . .	ellipses	201	133
†	dagger	160	134
‡	double dagger	224	135
ˆ	caret	246	136
‰	per thousand	228	137
Š	capital S hacek		138
‹	open angle bracket	220	139
Œ	capital oe diphthong	206	140
‘	open single quote	212	145
‘	close single quote	213	146
“	open double quote	210	147
“	close double quote	211	148
•	bullet	165	149
–	en dash	208	150
—	em dash	209	151
˜	tilde	247	152
™	trademark	170	153
š	lowercase s hacek		154
›	close angle bracket	221	155
œ	lowercase oe diphthong	207	156
Ÿ	capital Y umlaut	217	159
	non-breaking space	160	160
¡	inverted exclamation	193	161
¢	cent	162	162

Character	*Name*	*Macintosh*	*PC*
£	pound	163	163
	cell (in a table)	219	164
¥	yen	180	165
¦	pipe	124	166
§	section	164	167
¨	umlaut	172	168
©	copyright	169	169
ª	ordinal, feminine	187	170
«	left chevrons	199	171
¬	not	194	172
-	soft hyphen	248	173
®	registered	168	174
¯	macron	248	175
°	degree	161	176
±	plus or minus	177	177
²	superscript 2	50	178
³	superscript 3	51	179
´	acute accent	171	180
µ	micro	181	181
¶	pilcrow	166	182
·	middle dot	225	183
¸	cedilla	252	184
¹	superscript 1	49	185
º	ordinal, masculine	188	186
»	right chevrons	200	187
¼	one-fourth		188
½	one-half	189	189
¾	three-fourths	190	190
¿	inverted question	192	191
À	capital A, grave	203	192
Á	capital A, acute	231	193
Â	capital A, circumflex	229	194
Ã	capital A, tilde	204	195
Ä	capital A, umlaut	128	196
Å	capital A, angstrom	129	197
Æ	capital AE, diphthong	174	198
Ç	capital C, cedilla	130	199
È	capital E, grave	233	200

Character	*Name*	*Macintosh*	*PC*
É	capital E, acute	131	201
Ê	capital E, circumflex	230	202
Ë	capital E, umlaut	232	203
Ì	capital I, grave	237	204
Í	capital I, acute	234	205
Î	capital I, circumflex	235	206
Ï	capital I, umlaut	236	207
Ð	capital eth		208
Ñ	capital N, tilde	132	209
Ò	capital O, grave	241	210
Ó	capital O, acute	238	211
Ô	capital O, circumflex	239	212
Õ	capital O, tilde	205	213
Ö	capital O, umlaut	133	214
×	multiply	120	215
Ø	capital O, slash	175	216
Ù	capital U, grave	244	217
Ú	capital U, acute	242	218
Û	capital U, circumflex	243	219
Ü	capital U, umlaut	134	220
Ý	capital Y, acute	89	221
Þ	capital thorn		222
ß	sharp s	167	223
à	lowercase a, grave	136	224
á	lowercase a, acute	135	225
â	lowercase a, circumflex	137	226
ã	lowercase a, tilde	139	227
ä	lowercase a, umlaut	138	228
å	lowercase a, angstrom	140	229
æ	lowercase ae, diphthong	190	230
ç	lowercase c, cedilla	141	231
è	lowercase e, grave	143	232
é	lowercase e, acute	142	233
ê	lowercase e, circumflex	144	234
ë	lowercase e, umlaut	145	235
ì	lowercase i, grave	147	236
í	lowercase i, acute	146	237

Character	*Name*	*Macintosh*	*PC*
î	lowercase i, circumflex	148	238
ï	lowercase i, umlaut	149	239
ð	lowercase eth		240
ñ	lowercase n, tilde	150	241
ò	lowercase o, grave	152	242
ó	lowercase o, acute	151	243
ô	lowercase o, circumflex	153	244
õ	lowercase o, tilde	155	245
ö	lowercase o, umlaut	154	246
÷	divide	214	247
ø	lowercase o, slash	191	248
ù	lowercase u, grave	157	249
ú	lowercase u, acute	156	250
û	lowercase u, circumflex	158	251
ü	lowercase u, umlaut	159	252
ý	lowercase y, acute	121	253
þ	lowercase thorn		254
ÿ	lowercase y, umlaut	216	255

WILDCARDS

?	Finds any single character: “c?t” finds “cat,” “cut,” and “cot.”
*	Finds any string of characters: “b*d” finds “bad,” “bread,” and “bewildered.”
[ ]	Finds one of the specified characters: “b[ai]t” finds “bat” and “bit” but not “bet.”
[-]	Finds any single character in the specified range (which must be in ascending order): “[l-r]ight” finds “light,” “might,” “night,” and “right” (and “oight,” “pight,” and “qight,” if they exist).
[!]	Finds any single character except those specified: “m[!u]st” finds “mist” and “most” but not “must.” “t[!ou]ck” finds “tack” and “tick” but not “tock” or “tuck.”
{n}	Finds exactly n occurrences of the previous character or expression: “re{2}d” finds “reed” but not “red.”
{n,}	Finds at least n occurrences of the previous character or expression: “re{1,}d” finds “red” and “reed.”

{n,m}	Finds from n to m occurrences of the previous character or expression e.g. 10{1,3} finds "10," "100," and "1000."
@	Finds one or more of the previous character or expression *before* something else: "me@t" finds both "met" and "meet"; "me@" (without the "t") finds only "me" because nothing comes after it.
<	Finds the beginning of a word: "<inter" finds "interest" and "interrupt" but not "splinter."
>	Finds the end of a word: "in>" finds "in" and "main" but not "inspiring."

Ranges

[a-e]	Finds any of a, b, c, d or e
[0-9]	Finds any digit
[a-z]	Finds any occurrence of a lowercase letter.
[A-Z]	Finds any occurrence of an uppercase letter.
[!x-z]	Finds any single character except those in the specified range: "t[!a-m]ck" finds "tock" and "tuck" but not "tack" or "tick."

Groups

( )	Creates a wildcard group
\1	Inserts the contents of the first wildcard group in the "Replace With" text.
\2	Inserts the contents of the second wildcard group in the "Replace With" text; etc.

Characters to "Escape"

There are actually quite a few characters that have to be escaped if you want to use them as characters rather than wildcards. Here they are, along with their meaning as wildcards:

? any character
* zero or more characters
[begins a range
] ends a range

{ begins a specified number
} ends a specified number
(begins an expression
) ends an expression
< begins a word
> ends a word
^ introduces a numeric character code
\ the escape character!

Built-in Codes with Wildcards

Character	*Works with wildcards*	*Doesn't work with wildcards*	*Wildcard or ANSI equivalent*
Annotation Mark (comment)	^a		
Any character		^?	?
Any digit		^#	[0-9]
Any letter		^$	[A-z]
Caret character	^^		
Column break	^n		
Em dash	^+		
En dash	^=		
Endnote mark		^e	^02
Field		^d	^019
Footnote mark		^f	^02
Graphic		^g	^047
Line break	^l		
Manual page break	^m		
Nonbreaking hyphen	^~		
Nonbreaking space	^s		
Optional hyphen	^-		
Paragraph mark		^p	^013
Section break		^b	^012
Tab character	^t		
White space		^w	[^s^t^032]

Other Resources

There's an excellent explanation of how to find and replace with wildcards at the Microsoft Word MVP site:

http://www.mvps.org/word/FAQs/General/UsingWildcards.htm

Allen Wyatt provides all kinds of information about finding and replacing on his WordTips site:

http://wordribbon.tips.net/C0919_Find_and_Replace.html

Graham Mayor provides a beautifully detailed and illustrated wildcard tutorial:

http://www.gmayor.com/replace_using_wildcards.htm

You can download a free find and replace reference card from the Editorium website:

http://www.editorium.com/wildcard_reference.pdf

I recommend printing it on 8.5-by-11-inch cardstock, both front and back (each side will be different), and then cutting the cardstock in half lengthwise (at 4.25 inches). That will give you a handy reference card to keep by your computer and another card to give to a friend.

Acknowledgments

I would like to thank the following readers of *Editorium Update* for their questions, suggestions, and inspiration:

Richard Adin
Pamela Angulo
Karen L. Bojda
Michael Coleman
Eric Fletcher
Allene M. Goforth
Steve Hudson
Sharon Key
Andrew Lockton
Jeanne Pinault
Hilary Powers
Mary L. Tod
Rosalie Wells

I would also like to thank my daughter Rachel Lyon for her excellent index. Rachel writes indexes for some of the best publishers in the world, and she would love to write an index for you. You can learn more here:

http://www.lyonpublishingservices.com

Index

Wildcard Cookbook
for Microsoft Word

Microsoft Word's advanced search features are extremely powerful, but they're also virtually undocumented; most explanations of their use have been limited to a simple table of wildcards. This book explains in detail how you can use these powerful tools to blaze through repetitive problems that would take hours to correct by hand. It covers:

- Using Word's find and replace options.
- Finding and replacing with Word's built-in codes.
- Finding and replacing with numeric character codes (ASCII, ANSI, and Unicode).
- Finding and replacing with wildcards, including wildcard ranges, wildcard groups, and the powerful "Find What Expression" wildcard.

It also includes numerous examples of using wildcards with real-world documents, wildcard tips and techniques from readers of *Editorium Update,* and a thorough reference section.

Jack Lyon writes, "Although I write and sell Microsoft Word macros for a living, the tools I depend on most are the advanced features of Word's find and replace. Learning to use these tools takes time and effort, but the payoff is huge. I hope this book will help you understand how powerful Word's advanced search features can be."

The Editorium

ISBN 978-1-4341-0398-7

90000

9 781434 103987