Typing Arabic in Vim

## Introduction

LTR display in Vim

LTR display in Vim

I often write documents, such as exams and lecture notes, that contain both Latin and Arabic script, often on the same line of text. This can be challenging due to the complications of mixing of LTR (left-to-right) and RTL (right-to-left) scripts. These seems like an easy problem to solve for software developers, and it is, only not in software with graphical WYSIWYG interfaces, such as Word of OpenOffice. (I’m sure everyone who has tried writing mixed direction text in such software share my frustration with them, and I will therefore refrain from rants.) Since my shift to exclusively producing and editing text in plain text formats (.txt, .mkd, .csv, etc.) with the editor Vim, writing texts with mixed directionlity has become a lot easier. In this post I will attempt explain how.

When typing and editing on the computer, switching from Latin to Arabic script at the most basic level involves changing two things: a) the direction in which characters appear, and b) the keyboard layout that is used to insert characters in the text. In Vim, each of these two changes is done idependently with the options rightleft for directionality and keymap for the keyboard layout. In I will discuss them in turn.

# Directionality and display

It is important to understand that when text is stored electronically is is represented by series of characters, including spaces and line breaks, order one after another in a long list. Think of this as the order they are typed. Some of these characters come with information that they are to be displayed LTR and others RTL, but it doesn’t effect the order in which they are stored in file. Thus if i type hello the letters are stored in that particular order, and it can be displayed as LTR (hello) or as RTL (olleh) with the characters in the same order, only the direction the are displayed in has changed. In English, of course, the former is appropriate. If i type the letters م``ر``ح``ب``ا, they can be displayed as RTL (ابحرم) or as LTR (مرحبا) with the characters in same order but with different direction of display. For Arabic, the latter is appropriate.

Now, what WYSIWYG word processors do is that when they come across characters in the file that are part of an alphabet that is written from right to left, it rearranges them so that they are displayed RTL on the screen. This is of course nice in that it shows the text as it is intended to be read by humans. However, try to highlight a portion of a text with mixed directionlity (or even with only LTR) with the mouse, add a at a direction switch, enclose a single LTR word in parenthesis, and you are likely to run into trouble. Editing a text displayed like this is a pain.

Enter Vim. Vim lacks the capability of visually reordering specific letters according to the reading direction of that particular alphabet. All characters are displayed in a logical order, either LTR or RTL, as in the image in the image below introduction to this post. In the left hand with RTL display the text in English is be displayed correctly and Arabic text reversed. In the right hand image, the text in English is be displayed reversed and Arabic text is displayed correctly.

RTL display LTR display in Vim

You switch between these two ways of displaying a text with :set rightleft for RTL :set norightleft for LTR. I have this mapped to \a to quickly switch back and forth. (This mapping also switches to Arabic keyboard layout, as explained below.) Thus, writing documents with mixed Latin/Arabic script becomes a process of quickly reorienting the entire text back and forth between the two modes exemplified above whenever you want to switch between editing or typing a word or section of the text in Arabic or Latin script. The idea is that all characters, regerdless of how they are to appear in the final document, are displayed in their logical, rather than their proper visual order. The reordering of the characters of as appropriate is postponed to when the final document is compiled or converted to, be it in a pdf using LaTeX, on a .doc file using Pandoc, or the rendering if a HTML file in browser (as in the text you are now reading).

The drawback of this method is that it is a bit disorienting. It is more difficult to navigate the text you are writing using visual cues, like headings and list, since these change shape with every switch in directionlity. The gains, as compared to WYSIWYG word processors, are however significant. Editing, marking, copying, and moving things around becomes completely streamlined, intuitive, and friction free. You can edit and change stuff as easily as you would piece of text with containing only Latin script.

# Keymap

Vim also comes with its keyboard layout for Arabic (see :h mbyte-keymap). You switch to it with :set keymap=arabic. Using Vim’s internal settings for keyboard layout is preferable to using the system for keyboard layouts in the OS. Of you use the OS keyboard switch, normal mode commands no longer work since the keys that are input to Vim are now different, meaning that all of Vim’s core functionality is lost. Vim’s internal keymap switching only applies to insert mode and search mode while normal mode is unaffected, meaning that everything works intuitively.

The Arabic keyboard layout and the directionlity are thus set independently of one another. If you change the keymap option without changing the rightleft option, you insert Arabic characters but they are displayed LTR, as in the image below. I have found that if you only type a few isolated Arabic words here and there in an otherwise English text, this is the most convenient way to do it. The Arabic words will look strange on the screen with letters ordered from left to right and connected according to this order, but they are still more or less readable, and for a few isolated words this is not too big of a deal. It is well worth it to avoid the hassle of editing text with mixed directionality.

Arabic words in a LTR text.

Arabic words in a LTR text.

Another advantage with Vim’s keyboard layouts is that they are very easy to modify. Each keyboard layout is specified in a file that consists of two columns, the first containing the input key and the second the output character. The example below is an excerpt from the Arabic keyboard layout with the Arabic output characters specified as Unicode characters.

loadkeymap  
q <char-0x0636> " (1590) - DAD  
w <char-0x0635> " (1589) - SAD  
e <char-0x062b> " (1579) - THEH  
r <char-0x0642> " (1602) - QAF  
[...]

I am used to having the letter *dhaal* (ذ) on the §-key on the top right of the keyboard, a setup common on PCs, but in Vim’s Arabic layout it is on the `-key next to the left shift. To change this I copied the original Arabic keymap file (I use MacVim so this file is locate in /Applications/MacVim.app/Contents/Resources/vim/runtime/keymap/arabic\_utf-8.vim) to ~/.vim/keymap/ and changed its name to arabic-pc\_utf-8.vim. The file name is arbitrary, accept that it must end with \_utf-8.vim. In the new file change the input key that produces the letter *dhāl* from \`` to§(line&nbsp;54). I can now call this new modified keyboard layout with:set keymap=arabic-pc` (the firs part of the file name of the new layout). You could also change the original file, but this method is better since you can store it separately and make it more easy accessible for different Vim installations. My slightly modified Arabic keymap file can for instance be found [here](https://github.com/andreasmhallberg/dotfiles).

## Language switching command

If you are often switching back and forth between English (or some other language that uses Latin script) and Arabic, it is a good idea to map function that changes to RTL directionality and the Arabic keyboard layout, and then back. I have this set to \a for Arabic and \e for English using the code below (adapted from [my vimrc](https://github.com/andreasmhallberg/dotfiles/blob/master/.vimrc)).

" Switch to English - mapping  
nmap <Leader>e :<C-U>call EngType()<CR>  
" Switch to Arabic - mapping  
nmap <Leader>a :<C-U>call AraType()<CR>  
  
" Switch to English - function  
function! EngType()  
" To switch back from Arabic  
 set keymap= " Restore default (US) keyboard layout  
 set norightleft  
endfunction  
  
" Switch to Arabic - function  
function! AraType()  
 set keymap=arabic-pc "Modified keymap. File in ~/.vim/keymap/  
 set rightleft  
endfunction