L2/12-288

Title: Solidus: an undocumented bidi incompatibility

Authors: Roozbeh Pournader and Behdad Esfahbod (Google)

Date: 2012-07-30

Action: For UTC's information and potential action

Live link: http://goo.gl/TAuH3

In March 2005, Unicode 4.0.1 was released. In that version, the bidi class of U+002F SOLIDUS changed from ES to CS as a request from Microsoft, as a compromise between UTC members to remove a phrase in the UBA that used to allow implementations override the bidi classes of characters; Other vendors agreed to change their implementations. This was thought to be unifying all bidi implementations and make documents created in the future readable everywhere, at the cost of breaking tons of existing documents at the time. (The classes of a few other common characters were also changed, for the same reason.)

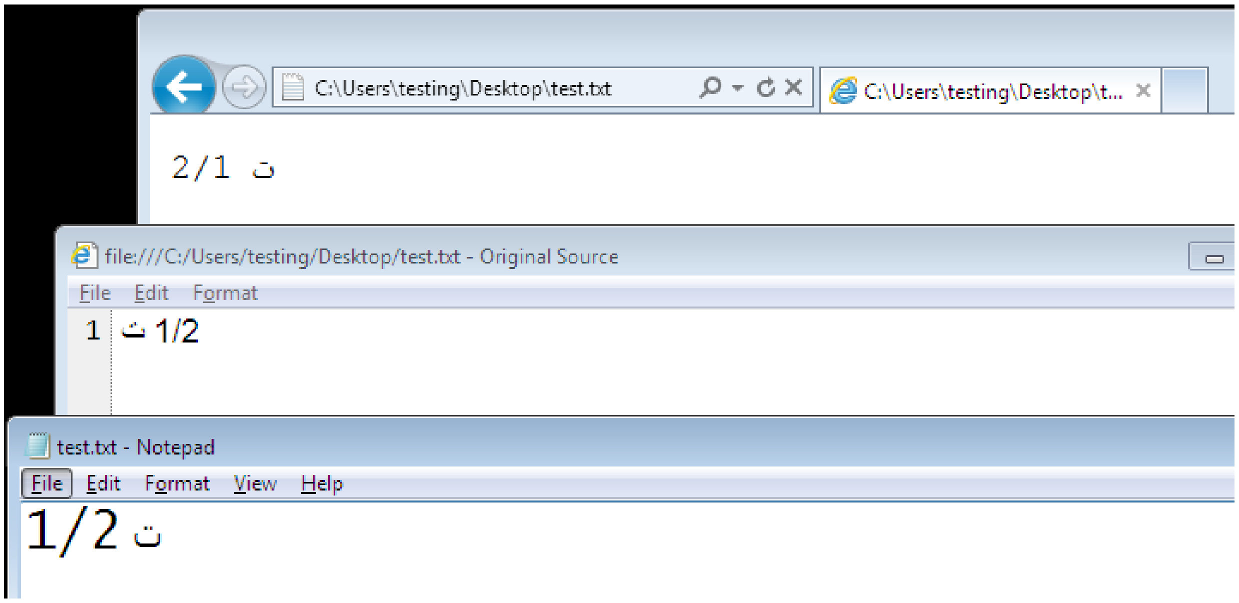
C:\Users\testing\Desktop\test.txt
C:\Users\testing\Desktop\t...
2/1 ت
file:///C:/Users/testing/Desktop/test.txt - Original Source
File Edit Format
1 ت 1/2
test.txt - Notepad
File Edit Format View Help
1/2 ت

Image 1: *The same text file containing the sequence <062A, 0020, 0031, 002F, 0032> opened in Internet Explorer 9, Internet Explorer 9's view source mode, and Notepad on the same Windows 7 machine, resulting in three different renderings. Note that the all three applications are assuming the file to be in LTR paragraph mode, which we guess is an application of rule HL1 (the file itself does not specify any such protocol).*

The negative consequences of this decision continue to be felt. Years later, the logical string "T 1/2", in which "T" is assumed to be an Arabic letter, is displayed differently in Microsoft Internet Explorer compared to anywhere else. While Chrome, Firefox, and even Windows's own Notepad show the string visually as "1/2 T", the strings is shown as "2/1 T" in Internet Explorer. (This was tested on Windows 7, with Internet Explorer 9, and latest stable versions of Chrome and

Firefox. It was also tested with a few older versions of Microsoft Windows and Internet Explorer, going back to Windows XP and Internet Explorer 7. They all resulted in the same rendering.)

Our guess is that either Internet Explorer still treats solidus as having the ES bidi class, while Windows's lower-level implementations consider it to be CS, or that Internet Explorer is treating all web pages as using a private high-level protocol possibly using the HL2 rule. We did not have time to try and figure out the exact rules, but there may even be subvariations of the rule used in different applications (see Image 1, above). Similar incompatibilities may exist for other characters that changed bidi class in Unicode 4.0.1, which included plus and minus signs and their variants.

This is not a rare pattern. It covers most combinations of solidi and numbers in Arabic script text. It appears in millions of documents. The most common uses are dates in Arabic and Persian. Solidi are also commonly used by some authors as a decimal separator (or even a thousands separator, usually figurable from the context) in Persian text.

A content author, not knowing about IE's different treatment of solidi, creates a simple web page, tests it on IE (a common browser in the Middle East), and pushes it to the web. Now, his readers will be seeing different renderings based on the platform they use to read the text.

Consider the following web page: http://www.denahospital.com/aboutus-history-fa.html, for a hospital in Shiraz. In the page, a few numbers, quoted in Iranian rials, are displayed as "16/500/000" in IE and "000/500/16" in other browsers. In the hospital web page, it's obvious from the context what the correct number is, and the experienced reader will figure out there's something wrong and will usually blame the poor web page author.

But the actual content is not always obvious from the context, nor is the reader is always experienced enough to figure out the problem. Consider the following web page http://www.bbc.co.uk/persian/iran/2010/07/100716_u01-press.shtml from BBC Persian. The first paragraph talks about businesses filing 1.3 million tax returns:

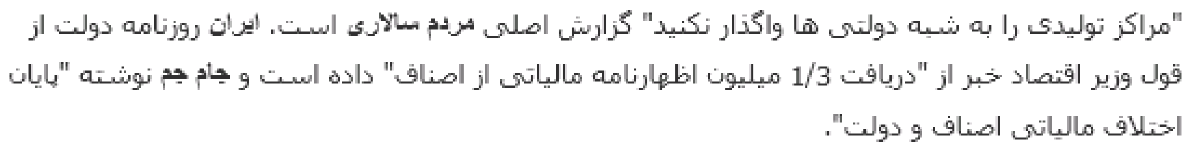
"مراکز تولیدی را به شبه دولتی ها واگذار نکنید" گزارش اصلی **مردم سالاری** است. **ایران** روزنامه دولت از
قول وزیر اقتصاد خبر از "دریافت 1/3 میلیون اظهارنامه مالیاتی از اصناف" داده است و **جام جم** نوشته "پایان
اختلاف مالیاتی اصناف و دولت".

Image 2: *Internet Explorer 9 under Windows 7*

Or is it 3.1 million tax returns?

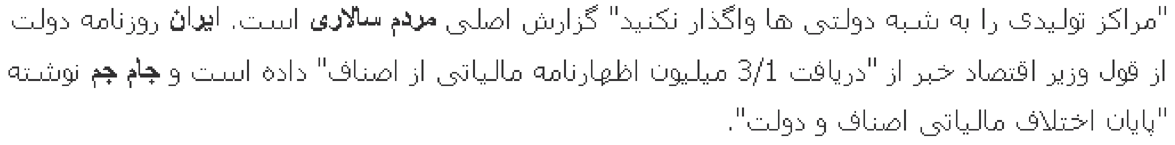
"مراکز تولیدی را به شبه دولتی ها واگذار نکنید" گزارش اصلی **مردم سالاری** است. **ایران** روزنامه دولت
از قول وزیر اقتصاد خبر از "دریافت 3/1 میلیون اظهارنامه مالیاتی از اصناف" داده است و **جام جم** نوشته
"پایان اختلاف مالیاتی اصناف و دولت".

Image 3: *Firefox 14 under Windows 7*

As we don't know which tools were used by the authors in creating, converting, and testing the page content (assuming the page was even tested), we cannot know if it was really 1.3 million returns, or 3.1 millions. But what we can be sure of, is that the author didn't care which character to put first in the logical stream, the most significant digit or the least significant digit. He only cared about seeing the number right. We can also be sure that almost all readers of the news, if using IE, assumed 1.3 million returns, and if using Chrome or Firefox, assumed 3.1 million returns.

More advanced content creators, when they learn about such issues, simply give up and learn to not use solidi around numbers. They will spell out dates, and use dots, commas, or other separators for separating digits. It's only the very select few who learn about the bidi formatting codes, or directional spans, and then figure out how to use them properly.

Our view

Even if knowledge of Unicode bidi issues was widely available, we cannot expect all simple context creators to test their very simple text on a few different browsers and a few different operating systems. Unicode is supposed to be a standard, after all. We cannot expect every bidi speaker in the world who ever uses a cellphone to write a text message or email to know about different versions of the bidi algorithm and their difference in handling "very plain" text files only containing Arabic or Hebrew text.

Also, note that bidi controls are not available in every environment. Content creators who had been using legacy character sets like ISO/IEC 8859-6 have no access to such characters. The only tool they had to get their document to display right are visual bidi hacks. So even if they want to create a document that is both logically and visually correct, they would not be able to do that. In such environs, content creators always opt for the correct visual representation, as that is what is eventually seen by the reader.

Very few content creators actually try to get both visual and logical representations of data to be correct. Even at Google, some translators sometimes encode localized strings visually. When facing a choice between logical and visual representation of text, content creators always opt for the correct visual. In a way, for almost all bidi content on and off the web, the actual **"source" text is the visual representation of text at the time the author was creating the document.** It is only with auto-generated text that the source may be logical or half-logical/half-visual. Even the existence of bidi control characters or markup is not a convincing signal that the logical order of the string matches its semantics.

In the meanwhile, most content creators continue to be unaware of the details of the Unicode Bidi Algorithm. We have found that very few people can learn to use the bidi algorithm properly, and even the very few select power users who have worked with the algorithm for years have blind spots. We consider the proper solution to be neither educating every bidi content creator about the details of the algorithm, nor actively improving the Unicode Bidi Algorithm. We believe the best solutions are based on making sure the original author's intent is preserved as much as possible, and also making data entry and text editing as easy and intuitive as possible for content creators.

Potential actions

- Add notes to UAX #9 mentioning that incompatible implementations of the UBA exist and suggest solutions for content authors for getting around the existing incompatibilities. Specifically, mention incompatibility issues with Solidus and how content authors can get around them by using bidi formatting codes.
- Extend BidiTest.txt (or add a new bidi test file) to add tests containing actual characters, instead of just bidi classes. This is to help implementations that wish to conform to the Unicode Standard check that they are assigning proper bidi classes to characters common in bidi contexts.

Acknowledgments

Mark Davis, Aharon Lanin, and Luke Swartz reviewed this document, contributed their insight, and helped the authors with earlier drafts of this document. They may not agree with all the points raised in this document, or not agree as strongly as we do.