Report of the source code ad hoc working group **L2/22-072**

Proposal for amendments to UAX#9 and UAX#31

To: Properties & algorithms group, UTC
From: Robin Leroy, Mark Davis, Source code ad hoc working group
Date: 2022-03-10

The source code ad hoc working group recommends amendments to non-normative text in Unicode Standard Annexes #9 and #31.

Proposed amendment to UAX#9

The working group recommends that the note in Section 4 "Bidirectional Conformance", Clause UAX9-C2, be amended as follows:

Current:

> *Note:* Use of higher-level protocols is discouraged, because it introduces interchange problems and can lead to security problems. For more information, see Unicode Technical Report #36, "Unicode Security Considerations" [UTR36].

Proposed:

> *Note:* The use of higher-level protocols introduces interchange problems, since the text may be displayed differently as plain text; see Section 6.5. This can have security implications. However, where the semantics of segment order are more significant than those of displayed order, as is the case for program text, higher-level protocols are recommended. For more information, see Unicode Technical Report #36, "Unicode Security Considerations" [UTR36].

The working group recommends that the following example be added after that note:

> Example: The expression (1) with the identifier tav replaced by the Hebrew letter ת would render as plain left-to-right text as (2), which is misleading, as it looks like a comparison between x + 1 and ת, whereas it is a comparison between x + ת and 1.
>
> (1) x + tav == 1
> (2) x + 1 == ת
>
> It is better instead to display it as follows, applying protocol HL4 to treat the identifiers in isolation:
>
> x + ת == 1

Proposed amendment to UAX#31

The working group recommends that the following note and example be added to Section 4 "Pattern Syntax", Clause UAX31-R3, after the existing note:

> Note: Two implicit directional marks are among Pattern_White_Space characters; if these can be freely inserted between tokens, they can be used to restore the logical order of tokens which would be illogically reordered by the Unicode Bidi Algorithm in the absence of higher-level protocols.

Example: The expression (1) with the identifier tav replaced by the Hebrew letter ת would render as plain left-to-right text as (2), which is misleading, as it looks like a comparison between x + 1 and ת, whereas it is a comparison between x + ת and 1.

(1) x + tav == 1
(2) x + 1 == ת

Inserting a left-to-right mark after the identifier ת yields a better rendering:

x + ת == 1

Rationale

While the working group has yet to define the specifics of the desired rendering of source code, and of the means by which this rendering may be achieved, some broad outlines are already clear. Future guidance will make use of some mechanisms already described in these annexes, some of which have been standardized for very similar purposes; indeed these mechanisms are already in use in existing implementations: Visual Studio implements UAX#9 HL4; multiple syntaxes defined by the Locale Data Markup Language[1], Ada 2012[2] and later, Rust[3] 1.9 and later allow for implicit directional marks wherever whitespace is allowed.

Context for the possible applications of those mechanisms is however lacking in the annexes. The proposed amendments provide that context.

[1] See, *e.g.*, the syntaxes for UnicodeSet, plural rules, collation rules.
[2] ISO/IEC 8652:2012; see the *Annotated Ada Reference Manual*, 2.2(7.1/3).
[3] See *The Rust Reference*, 2.5.