From: Frank Farance
Organization: Farance Inc.
Telephone: +1 212 486 4700
Fax: +1 212 759 1605
E-mail: frank@farance.com
Date: 1996-09-13
Document Number: WG14/N605 X3J11/96-069
Subject: Extended trigraphs and backslash sequences.

ABSTRACT

The paper is combination of features provided by Clive Feather and Frank Farance. These features have been discussed over the E-mail reflector. The paper proposes two features: extended trigraph characters (which extend the source character set) and extended backslash characters (which extend the execution character set). There have been several attempts to extend the character sets using a single mechanism. A single mechanism doesn't work because: (1) silent changes are introduced because the mechamism causes ambiguties in preprocessing, OR (2) preprocessor-only interpretation, common in much publicly available source code, is changed silently, (3) ambiguities are introduced when converting, supposedly, common source code to other encodings, (4) the implementation model is too difficult to implement.

ISSUES

The follow are problems with the current proposal, WG14/N574, for extended characters.

Issue #1: The proposal introduces quiet changes to preprocessor behavior. Because characters outside the basic source character set are translated to the "??u" notation, programs that weren't affected previously by the preprocessor are changed. For example:

```
        printf("$2.50");
```

is *required* to be transformed to:

```
        printf("??u00242.50");
```

even if the "$" character is available in the native character set. This would be a surprise for anyone running there code through a standard-conforming preprocessor. I know that we don't make any requirements for preprocessor-only translations, but this would be a surprise for much code that uses the preprocessor.

Issue #2: It is possible to include *interesting* characters within strings:

```
        printf("??u0022");      /* line #1 */
```

This might look ok on a 7-bit system, but if this code were

translated for 16-bit systems (with a native 16-bit editor), the code might appear as:

```
printf("""");            /* line #2 */
```

One would hope that the meaning line #1 stayed the same, but the only workaround to this is to make a fairly complex character set translator (requires much context knowledge) to translate appropriately.

Issue #3: Text editors and translators all require many exceptions to handle all the special cases implied by the proposal. For example, converting to ISO 646:

(1) If the character is a C trigraph, then emit the appropriate trigraph.

(2) If the character is in the basic source set, then emit the character.

(3) If the character is not in the basic source set, then:

- If we are in an identifier, then we can only emit the 16-bit "??uxxxx".

- If we are not in an identifier, then we have choice between the 16-bit "??uxxx" and the 32-bit "??Uxxxxxxxx".

Issue #4: We can't use "??U0000xxxx" for identifiers even thought only 16 bits are specified.

Issue #5: One technique for hiding trigraphs is to take advantage of string concatentation. However, I think the following line changes meaning:

```
printf("?" "?" "u0021");
```

I think the problem with "??u" is that it behaves differently than trigraphs, which is a surprise to many people. I'm told the reason for not making them like trigraphs is that a trigraph model would force the compiler to process text as 32-bit characters. I don't believe this is true. You can use the same technique for extended characters in source code as the proposal describes for extended characters in identifies:

''[*Footnote: On systems in which linkers cannot accept extended characters, an encoding of the universal-character-name may be used in forming valid external identifiers. For example, some otherwise unused character or sequence of characters may be used to encode the "??u" in a universal-character-name.''

Overall, I think the cause for programmer confusion is related to the confusion over this proposal: are we describing source characters or execution characters? Source characters are used to describe the program (e.g., trigraphs). Execution characters are used for processing and I/O (e.g., backslash escapes). Some, but not necessarily all, source characters can be used in the execution character set. Some execution characters aren't available in the source character set and require backslashes to describe them. We should continue this conceptual model for extended characters (see below).

IMPROVEMENTS

The following improvments should be made to resolve these problems, i.e., changes to N574:

(1) The "??u" and "??U" sequences are extended trigraphs.

(2) Only 16-bit characters can be used in identifiers, but either form ("??uxxx" or "??U0000xxxx") can be used.

(3) If an extended trigraph describes a character in the basic C character set, the extended trigraph is equivalent to that character. For example, the following lines of code are equivalent:

```
printf(??u0022hello world??u0022);
printf("hello world");
```

(4) Add "\uxxx" and "\Uxxxxxxxx" to describe *execution* characters. For example:

```
printf("\u0022");
```

This code fragment would not suffer from misinterpretation as is was converted from ISO 646 to, say, Unicode.

(5) Only extended trigraphs that correspond to characters in "isspace" or "isgraph" can be portably included in source files.

(6) Change the conceptual model not to be biased towards an internal implementation of ISO 10646. For example, rather than ‘‘any character not in the basic character set is converted to a UCN’’, how about ‘‘UCNs are converted to the source character set when possible, otherwise, they are converted to implementation-defined sequences’’ (this is a summary of my model #4). In effect, it says ‘‘convert as much as I can to the local character set and mark the other expansions specially’’ (in contrast to the WG21 WP model that is a ISO 10646

conceptual model). This would eliminate the problems: (1) characters being unnecessarily converted in the preprocessor, (2) the ??u0022 ocurring in strings. Meanwhile this improvement would have a conceptual model that corresponds to most of the implementations.

Overall, these changes are easy to incorporate. These changes eliminate many second and third order effects that are not obvious when first reviewing the N574 paper.