Title: Support for character sets in SC22 standards

Date: 2002-08-07

Source: Keld Simonsen

SC22 is the subcommittee in ISO and IEC responsible for standardizing programming languages, operating systems and software environments. These include programming languages like Fortran, COBOL, C, C++, Ada, APL, LISP, Basic, Pascal, PL/1, ECMAscript, C#, and the POSIX Operating system.

My background for writing this paper is that I am a member of the C, C++, POSIX and internationalization working groups of SC22, as well as the programming language bindings WG, and I have done programming also in Fortran, Algol, Simula, Basic, COBOL, and assembler. I am the project editor of a number of internationalization standards in SC22, and I am also active in the ISO subcommittee on character sets, SC2, where I am a coeditor of IS 10646. I have also done extensive localization of Linux for my native environment.

Some of these standards have something like 30 to 50 years of usage behind them, while some of them are desings of quite new origin. This is influencing how the standards is handling character sets, as there has been a growing emphasis on doing more sophisticated character set support as the computer usage has become very widespread in the world economy and computers have become a household commodity, where adequate support for the native languages of all the world has become an absolute necessity.

Traditionally the paradigma has been that ISO programming language standards should be independent of coded character sets, so that a language standard could apply to all platforms, and all vendors of programming language technology could have a fair chance of conforming to the standards on any platform. Some recent designs, such as ECMAscript and C# however only uses one coded character set, namely ISO/IEC 10646. As standards are mainly made for enabeling portablility of programs across platforms, I would recommend that each of the programming languages keep their capabilities wrt handling of coded character sets, such that SC22 would have to deal with two models for character handling: one that is coded character set neutral, and one that only uses IS 10646. It is noted that the character set neutral model of cause can handle IS 10646.

IS 10646 support

IS 10646 is a very important character set now in the marketplace, and will have even greater importance in the future. One could ask if we could not just do everyting in 10646, and forget the other character sets. But at least we are not there now, and my opinion is that we will have to deal with other character sets for a long time, maybe 30 years. Also IS 10646 has many coding forms, and is available in a number of versions, so just to be able to portably deal with 10646 requires a machinery which is quite like a general character set handling model. Restricting programming language standards to only 10646 would not buy us much, while retaining the general abstract character set model would keep an easy migration path open to have more programs use IS 10646. There is also a risk that other new character sets may be invented, that are better than IS 10646, or become widespread in use, though I doubt that this may happen.

It is noted that some programming languages cannot handle one of the more widespread forms of 10646. An example is C and C++ which lacks support for UTF-16. I would recommend that such support be added, in a way that is character set neutral, such that portability can be obtained, and that other machinery, as internationalization, can be applied as prescribed by the language.

APIs

With the advent of IS 10646 with its support for myriads of scripts and associated complex character handling, there is a need to be able to handle this adequately in a standardized way. Full character handling should be an integral part of the standards, so that portability can be achieved.

Furthermore there is a smaller need to in some cases deal explicitely with a specific character set, such as a file in a different character set than the system character set, or a file that has been obtained from another machine, eg via the network. For such cases it would be advisable that there be explict APIs to deal with coded character sets, while the general character handling be done as abstract characters, where the encoding of the character is not known.

Internationalization

One important aspect of using characters in programming languages, is the internationalization aspects, where the program behaves differently according to cultural preferences, eg all the text is in Japanese, or Russian or French, dependent of the preferences of the user. It is also here important that the character set support and the internationalization support be integrated with eachother.

Portability and registries.

Dealing with different character sets, and different cultural preferences may lead to loss of portability. To ensure that the program can run with the same results with the same cultural environment on dfferent platforms utilizing different character sets, it is necessary to have a worldwide recognized registry of these items, and that the programming language standards support these registered items. WG20 is maintaining such a registry as described in ISO/IEC 15897 - the cultural registry. This has a number of locales and definitions of more than 150 character sets described with mapping of each character to IS 10646, and it is harmonized with the specifications used on the Internet.