X/Open Preliminary Specification

File System Safe UCS

Transformation Format (FSS-UTF)

X/Open Company Ltd.

X/Open Preliminary Specification

File System Safe UCS Transformation Format (FSS-UTF)

ISBN: 1-872630-96-0

X/Open Document Number: P316

Published by X/Open Company Ltd., U.K.

Any comments relating to the material contained in this document may be submitted to X/Open at:

X/Open Company Limited
Apex Plaza
Forbury Road
Reading
Berkshire, RG1 1AX
United Kingdom

or by Electronic Mail to:

XoSpecs@xopen.co.uk

Contents

Chapter	**1**	**Introduction**	**1**
	1.1	Background	1
	1.2	Scope	1
Chapter	**2**	**Transformation Format**	**3**
	2.1	Criteria	3
	2.2	Specification	4
	2.3	Example	6
		Glossary	**9**
		Index	**11**

List of Figures

2-1	Transformation Format	4

Preface

X/Open

X/Open is an independent, worldwide, open systems organisation supported by most of the world's largest information systems suppliers, user organisations and software companies. Its mission is to bring to users greater value from computing, through the practical implementation of open systems.

X/Open's strategy for achieving this goal is to combine existing and emerging standards into a comprehensive, integrated, high-value and usable system environment, called the Common Applications Environment (CAE). This environment covers the standards, above the hardware level, that are needed to support open systems. It provides for portability and interoperability of applications, and allows users to move between systems with a minimum of retraining.

The components of the Common Applications Environment are defined in X/Open CAE Specifications. These contain, among other things, an evolving portfolio of practical application programming interfaces (APIs), which significantly enhance portability of application programs at the source code level, and definitions of, and references to, protocols and protocol profiles, which significantly enhance the interoperability of applications.

The X/Open CAE Specifications are supported by an extensive set of conformance tests and a distinct X/Open trademark - the XPG brand - that is licensed by X/Open and may be carried only on products that comply with the X/Open CAE Specifications.

The XPG brand, when associated with a vendor's product, communicates clearly and unambiguously to a procurer that the software bearing the brand correctly implements the corresponding X/Open CAE Specifications. Users specifying XPG-conformance in their procurements are therefore certain that the branded products they buy conform to the CAE Specifications.

X/Open is primarily concerned with the selection and adoption of standards. The policy is to use formal approved *de jure* standards, where they exist, and to adopt widely supported *de facto* standards in other cases.

Where formal standards do not exist, it is X/Open policy to work closely with standards development organisations to assist in the creation of formal standards covering the needed functions, and to make its own work freely available to such organisations. Additionally, X/Open has a commitment to align its definitions with formal approved standards.

X/Open Specifications

There are two types of X/Open specification:

- *CAE Specifications*

  CAE (Common Applications Environment) Specifications are the long-life specifications that form the basis for conformant and branded X/Open systems. They are intended to be used widely within the industry for product development and procurement purposes.

Developers who base their products on a current CAE Specification can be sure that either the current specification or an upwards-compatible version of it will be referenced by a future XPG brand (if not referenced already), and that a variety of compatible, XPG-branded systems capable of hosting their products will be available, either immediately or in the near future.

CAE Specifications are not published to coincide with the launch of a particular XPG brand, but are published as soon as they are developed. By providing access to its specifications in this way, X/Open makes it possible for products that conform to the CAE (and hence are eligible for a future XPG brand) to be developed as soon as practicable, enhancing the value of the XPG brand as a procurement aid to users.

- *Preliminary Specifications*

  These are specifications, usually addressing an emerging area of technology, and consequently not yet supported by a base of conformant product implementations, that are released in a controlled manner for the purpose of validation through practical implementation or prototyping. A Preliminary Specification is not a ''draft'' specification. Indeed, it is as stable as X/Open can make it, and on publication has gone through the same rigorous X/Open development and review procedures as a CAE Specification.

  Preliminary Specifications are analogous with the ''trial-use'' standards issued by formal standards organisations, and product development teams are intended to develop products on the basis of them. However, because of the nature of the technology that a Preliminary Specification is addressing, it is untried in practice and may therefore change before being published as a CAE Specification. In such a case the CAE Specification will be made as upwards-compatible as possible with the corresponding Preliminary Specification, but complete upwards-compatibility in all cases is not guaranteed.

In addition, X/Open periodically publishes:

- *Snapshots*

  Snapshots are ''draft'' documents, which provide a mechanism for X/Open to disseminate information on its current direction and thinking to an interested audience, in advance of formal publication, with a view to soliciting feedback and comment.

  A Snapshot represents the interim results of an X/Open technical activity. Although at the time of publication X/Open intends to progress the activity towards publication of an X/Open Preliminary or CAE Specification, X/Open is a consensus organisation, and makes no commitment regarding publication.

  Similarly, a Snapshot does not represent any commitment by any X/Open member to make any specific products available.

X/Open Guides

X/Open Guides provide information that X/Open believes is useful in the evaluation, procurement, development or management of open systems, particularly those that are X/Open-compliant.

X/Open Guides are not normative, and should not be referenced for purposes of specifying or claiming X/Open-conformance.

This Document

This document is a Preliminary Specification (see above). It is intended for systems programmers with experience in the C programming language.

Typographical Conventions

The following typographical conventions are used throughout this document:

- **Bold** font is used in text for filenames, keywords, type names, data structures and their members.
- *Italic* strings are used for emphasis or to identify the first instance of a word requiring definition. Italics in text also denote functions, which are shown as follows: *name*( ).
- Code examples are shown in `fixed width` font.
- Ranges of values are indicated with parentheses or brackets as follows:

— (a,b) means the range of all values from a to b, including neither a nor b

— [a,b] means the range of all values from a to b, including a and b

— [a,b) means the range of all values from a to b, including a, but not b

— (a,b] means the range of all values from a to b, including b, but not a.

Trade Marks

UNIX® is a registered trade mark of UNIX System Laboratories, Inc. in the U.S.A. and other countries.

X/Open™ and the ''X'' device are trade marks of X/Open Company Limited in the U.K. and other countries.

Referenced Documents

The following standards are referenced in this specification:

ISO C
: ISO/IEC 9899: 1990, Programming Languages — C (which is technically identical to ANS X3.159-1989, Programming Language C).

ISO 8859
: ISO 8859-*: 1987 Information processing — 8-bit single-byte coded graphic character sets — Parts 1 to 9 inclusive.

ISO/IEC 10646-1
: ISO/IEC DIS 10646-1: 1993, Information technology — Universal Multiple-Octet Coded Character Set (UCS).

Internationalisation Guide
: X/Open Guide, February 1992, Internationalisation Guide (ISBN: 1-872630-20-0, XO/GUIDE/91/030).

Chapter 1

Introduction

This chapter explains the need for a Universal Character Set Transformation Format.

1.1 Background

In the past most information processing systems were based on the 7-bit ASCII coded character set (codeset). Most languages require characters additional to those in the ASCII codeset. The ISO 8859 standards define codesets, each of which caters for a group of languages (the section on Character Sets in the **Internationalisation Guide** gives details). All the ISO 8859 codesets contain ASCII as a subset.

The referenced ISO/IEC 10646-1 standard offers a Universal Character Set (UCS) that provides the capability to encode multi-lingual text within a single codeset, by using either two octets (UCS-2) or four octets (UCS-4). ASCII-based operating systems need to cope with the representation and handling of the large number of characters possible with this new standard. The encoding of the ISO/IEC 10646-1 standard does not protect the null byte (byte with all bits zero) nor the ASCII slash character (/). An annex to the ISO/IEC 10646-1 standard specifies a UCS Transformation Format called UTF-1; however, the encoding of UTF-1 does not protect the slash character (/). The encoding for both the ISO/IEC 10646-1 standard and UTF-1 are therefore incompatible with existing UNIX implementations.

1.2 Scope

There is considerable work required to adapt existing programming languages, operating systems and utilities to the requirements of the ISO/IEC 10646-1 standard; that is outside the scope of this specification.

This document addresses the problem of the handling of the UCS by historical operating systems and utilities. In particular, this specification defines how data must be represented within the file system by specifying a File System Safe UCS Transformation Format (FSS-UTF) that is compatible with UNIX systems, supporting multi-lingual text in a single encoding.

This specification is based on the assumption that there is a requirement to maintain the existing operating system software investment, while at the same time taking advantage of the use of the large number of characters provided by the UCS. The objective is to define a transformation format that is usable on historical operating systems' file systems in a non-disruptive manner.

UCS is the process code for the FSS-UTF, which is intended to be used as a file code. It is an intermediate step towards full UCS support, providing a common and compatible encoding during the transition stage.

Chapter 2

Transformation Format

This chapter describes the criteria met by the FSS-UTF and gives the FSS-UTF specification.

2.1 Criteria

The FSS-UTF meets the following criteria:

- It is compatible with historical file systems (which disallow the null byte and the ASCII slash character as part of the filename).
- It is compatible with existing programs. The existing model for multi-byte processing is that ASCII characters do not occur in a multi-byte encoding. An FSS-UTF representation of a non-ASCII character contains no ASCII code values. If the UCS value is in the range [0x00,0x7F], the transformation is also in this range; otherwise, the transformed byte sequence does not contain any bytes in the range [0x00,0x7F].
- It is easy to convert from and to UCS.
- The first byte indicates the number of bytes to follow in a multi-byte sequence.
- The FSS-UTF is not extravagant in terms of the number of bytes used for encoding.
- It is possible to find the start of a character efficiently starting from an arbitrary location in a byte stream.

2.2 Specification

The FSS-UTF encodes UCS values in the range [0,0x7FFFFFFF] using multi-byte characters of lengths 1, 2, 3, 4, 5 and 6 bytes.

For all encodings of more than one byte, the initial byte specifies the number of bytes used by setting 1 in the equivalent number of high-order bits. The next most significant bit is always 0. For example, a 2-byte sequence starts with 110 and a 6-byte sequence starts with 1111110.

The following table shows the format of the first byte of a character; the free bits available for coding the character are indicated by `X`.

Single byte	`0XXXXXXX`	seven free bits
First of two bytes	`110XXXXX`	five free bits
First of three bytes	`1110XXXX`	four free bits
First of four bytes	`11110XXX`	three free bits
First of five bytes	`111110XX`	two free bits
First of six bytes	`1111110X`	one free bit

All subsequent bytes in multi-byte characters have the following format:

`10XXXXXX`	six free bits

For all subsequent bytes the two most significant bits are set to 10. Therefore, every byte that does not start with 10 is the start of a UCS character sequence.

Figure 2-1 illustrates the FSS-UTF:

No. of bytes in char.	No. of bits available for coding	Byte Sequence in Binary
1	7	`0VVVVVVV`
2	11	`110VVVVV 10VVVVVV`
3	16	`1110VVVV 10VVVVVV 10VVVVVV`
4	21	`11110VVV 10VVVVVV 10VVVVVV 10VVVVVV`
5	26	`111110VV 10VVVVVV 10VVVVVV 10VVVVVV 10VVVVVV`
6	31	`1111110V 10VVVVVV 10VVVVVV 10VVVVVV 10VVVVVV 10VVVVVV`

Figure 2-1 Transformation Format

The UCS value is the concatenation of the `V` bits in the multi-byte encoding. When there are multiple ways to encode a value, for example, UCS 0, only the shortest encoding is legal. The range of values possible with each type of byte sequence is shown in the following table.

No. of bytes in sequence	Hexadecimal Value Minimum	Maximum
1	00000000	0000007F
2	00000080	000007FF
3	00000800	0000FFFF
4	00010000	001FFFFF
5	00200000	03FFFFFF
6	04000000	7FFFFFFF

2.3 Example

The example below implements the ISO C standard *wctomb*() and *mbtowc*() functions, to demonstrate the algorithms for converting from UCS to FSS-UTF and converting from FSS-UTF to UCS. The example includes error checks, some of which may not be necessary for conformance:

```
typedef
struct
{
   int      cmask;
   int      cval;
   int      shift;
   long     lmask;
   long     lval;
} Tab;

static
Tab    tab[] =
{
   0x80,  0x00,  0*6,  0x7F,          0,          /* 1 byte sequence */
   0xE0,  0xC0,  1*6,  0x7FF,         0x80,       /* 2 byte sequence */
   0xF0,  0xE0,  2*6,  0xFFFF,        0x800,      /* 3 byte sequence */
   0xF8,  0xF0,  3*6,  0x1FFFFF,      0x10000,    /* 4 byte sequence */
   0xFC,  0xF8,  4*6,  0x3FFFFFF,     0x200000,   /* 5 byte sequence */
   0xFE,  0xFC,  5*6,  0x7FFFFFFF,    0x4000000,  /* 6 byte sequence */
   0,                                             /* end of table */
};

int
mbtowc(wchar_t *p, char *s, size_t n)
{
   long l;
   int c0, c, nc;
   Tab *t;

   if(s == 0)
       return 0;

   nc = 0;
   if(n <= nc)
       return -1;
   c0 = *s & 0xff;
   l = c0;
   for(t=tab; t->cmask; t++) {
       nc++;
       if((c0 & t->cmask) == t->cval) {
          l &= t->lmask;
          if(l < t->lval)
              return -1;
          *p = l;
          return nc;
       }
```

```
        if(n <= nc)
            return -1;
        s++;
        c = (*s ^ 0x80) & 0xFF;
        if(c & 0xC0)
            return -1;
        l = (l<<6) | c;
    }
    return -1;
}

int
wctomb(char *s, wchar_t wc)
{
    long l;
    int c, nc;
    Tab *t;

    if(s == 0)
        return 0;

    l = wc;
    nc = 0;
    for(t=tab; t->cmask; t++) {
        nc++;
        if(l <= t->lmask) {
            c = t->shift;
            *s = t->cval | (l>>c);
            while(c > 0) {
                c -= 6;
                s++;
                *s = 0x80 | ((l>>c) & 0x3F);
            }
            return nc;
        }
    }
    return -1;
}
```

Glossary

byte
An individually addressable unit of data storage that is equal to or larger than an octet, used to store a character or a portion of a character; see **character**. A byte is composed of a contiguous sequence of bits, the number of which is implementation-dependent. The least significant bit is called the *low-order* bit; the most significant is called the *high-order* bit. Note that this definition of *byte* deviates intentionally from the usage of *byte* in some international standards, where it is used as a synonym for *octet* (always eight bits). On a system based on the ISO POSIX-2 DIS, a byte may be larger than eight bits so that it can be an integral portion of larger data objects that are not evenly divisible by eight bits (such as a 36-bit word that contains four 9-bit bytes).

character
A sequence of one or more bytes representing a single graphic symbol or control code. This term corresponds to the ISO C standard term *multibyte character* (multi-byte character), where a single-byte character is a special case of a multi-byte character. Unlike the usage in the ISO C standard, *character* here has no necessary relationship with storage space, and *byte* is used when storage space is discussed.

character set
A finite set of different characters used for the representation, organisation or control of data.

coded character set
A set of unambiguous rules that establishes a character set and the one-to-one relationship between each character of the set and its bit representation.

null byte
A byte with all bits set to zero.

Index

ASCII ... 1
  code values ... 3
  slash character ... 3
byte ... **9**
character ... **9**
character set ... **9**
coded character set ... **9**
codeset ... 1
conversion
  from FSS-UTF to UCS ... 3, 6
  from UCS to FSS-UTF ... 3, 6
file system
  historical ... 3
FSS-UTF ... 1, 3
  criteria ... 3
  specification ... 4
null byte ... 3, 9
operating system ... 1
transformation format ... 1, 3
  criteria ... 3
UCS ... 1
  value ... 4
universal character set ... 1
UNIX ... 1
UTF-1 ... 1