X/Open Base Working Group

Interpretation Requests for XPG4 Base and VSX4

Document Identifier: **XoTGBase94-001**

Version: **1.12**

Last Updated: **June 2nd 1995**

Edited by: Andrew Josey, Novell USG R&D Europe

Email : a.josey@xopen.co.uk

1. Introduction

Scope

This document records the resolutions for interpretation requests raised by members of the X/Open Base Working Group for XPG4. Some of these interpretations have been raised as a result of testing using VSX4, others relate to the text of the XPG4 Base volumes. These interpretations are resolved by the X/Open Base and Verification Working Groups†. The interpretations recorded in this document have been abridged for clarity.

If you have any comments on this document please direct them to the editor.

Interpretation Process

The process for interpretations is to circulate a request for comments to xotgbase@xopen.co.uk for up to 4 weeks, collecting comments and then to recirculate with proposed resolutions for a one week informal ballot. All interpretation requests should be sent to a.josey@xopen.co.uk and should follow the format being circulated.

Where interpretations require an IEEE interpretation, reviewer comments will be used to form an X/Open position on the problem which will then be included in the IEEE interpretation request (the text of which is included in the descriptions below). The proposed X/Open resolution then becomes a temporary interpretation (see below) while the IEEE interpretation is pending. While a temporary interpretation is in force the tests will remain unchanged.

The status of temporary interpretations is as follows. Interpretation requests which are pending on interpretations of superior standards will be assigned temporary interpretations, indicating the preferable behaviour, but acknowledging the existence of a grey area in the X/Open specification. Products will be allowed to display either behaviour until the interpretation of the superior standard is received. At this point, the temporary interpretation will become permanent, (possibly reversed from the temporary waiver) and any systems which display the wrong behaviour will be eligible for temporary waivers.

1.1 Document History

- Issue 1.0 January 14th 1993.

  First issue in troff format.

† Interpretations may also be sent direct to X/Open to the electronic mailbox interpretations@xopen.co.uk. This is a separate process, which can incur costs of up to US$300 per interpretation request and is documented in the XPG4 binder. X/Open Base Working group interpretation reference numbers are often quoted in support of these requests.

- Issue 1.1 January 19th 1993.

  Change of document identifier from VWG92-006 to VWG93-001.
  Correction of status for interpretation VWG/009/063092.
  Addition of interpretation VWG/033/111992.

- Issue 1.2 February 23rd 1993.

  Addition of interpretations VWG/034/111292 and VWG/035/221292.

- Issue 1.3 April 14th 1993.

  Addition of interpretations VWG/035/221292 through VWG/039/022093.
  Minor stylistic editorial changes.

- Issue 1.4 May 5th 1993.

  Addition of interpretations VWG/040/031793 through VWG/043/032593
  Addition of references to VSX4 assertions where relevant.

- Issue 1.5 July 16th 1993.

  Addition of interpretations VWG/043/050793 through VWG/047/062993.

- Issue 1.6 July 22nd 1993.

  Editorial corrections to VWG/047/062993.

- Issue 1.7 October 5th 1993.

  Addition of interpretations VWG/048/061193 through VWG/050/072293

- Issue 1.8 November 10th 1993.

  Addition of interpretations VWG/051/082093 and VWG/052/082193

- Issue 1.9 February 28th 1994.

  Addition of interpretations 54,55,56,57,59,60 and 61. Change of document identifier to XoTGBase94-001. Note that 53 is still open.

- Issue 1.10 June 15th 1994.

  Update to record outcome of XoTGBase meeting in March 1994.

  Addition of interpretations 64 and 67. Note that resolutions for 62,63,65 and 66 have not been recorded (they were discussed at the Base meeting, if anyone has the information let me know). Also note that there was no agreement on 58.

  Update to record outcome of recent POSIX interpretations. These require a review by XoTGbase to determine possible effects on the test suite or base volumes. Interpretations affected are: 6,8,10,12 and 14. Note that interpretation number 14 is closed, the test suite is correct.

- Issue 1.11 June 22nd 1994.

Addition of note that 53 is still open.

Change test suite status of number 6 to (ii) with note fixed in VSX4.3.4.

Change test suite status of numbers 60 and 67 (ii) on advice of GWC.

- Issue 1.12 June 2nd 1995.

  Update for VSX4.3.5. Add closed interpretations 70 and 71.

2. Key to Interpretation Resolutions

The following is a key to status of the interpretation resolutions with respect to the VSX test suite.

i. This is not a grey area since the interpretation agrees that the current test suite behaviour is correct.

ii. This is a Known Problem in a particular release of VSX where the interpretation requires a change to current test suite behaviour. An asterix denotes that this has been fixed in a later update of the test suite. See notes at the end of the interpretation text for more details.

iii. Retain a documented grey area where it has not yet been resolved and X/Open's recommendation was to leave the tests as they are.

iv. Retain documented grey area that is agreed.

v. (this clause is not in use in this issue of this document)

vi. An interpretation has been made, but the test should be left as it is - no other action is required.

vii. This interpretation is beyond the scope of VSX.

3. Interpretations Summary

Reference	Description	Status	Rationale
VWG/001/053192	system()	closed	(i)
VWG/002/060192	mbstowcs()	closed	(ii) *
VWG/003/060392	regcomp()	closed	(vi)
VWG/004/061792	strptime()	closed	(ii) *
VWG/005/063092	last close on terminal	See IEEE interp	(i)*****
VWG/006/063092	extra dirs on tar archive	See IEEE Interp	(ii) ****
VWG/007/063092	trig fns	closed	(ii) *
VWG/008/063092	fcntl()	See IEEE Interp	(iii)
VWG/009/063092	namespace	pending X3J11 interpretation	(ii) *
VWG/010/063092	write zero bytes with O_APPEND	See IEEE interp	(i)
VWG/011/063092	long timezone names	closed	(iv)
VWG/012/063092	cfsetospeed,cfsetispeed	See IEEE interp	(i)*****
VWG/013/063092	getopt	closed	(i)
VWG/014/063092	timestamps on read-only filesys	closed	(i)
VWG/015/063092	chars,bytes and multibyte chars	closed	(i)
VWG/016/070792	Header files	closed	(ii) *
VWG/017/070792	(i) Pathname resolution	fwd to IEEE	(iii)
	(ii) PATH_MAX	closed	(iv)
VWG/018/070892	wcswidth	closed	(ii) *
VWG/019/071492	iconv	closed	(vi)
VWG/020/071492	CRNCYSTR	closed	(ii) *
VWG/021/080792	wchar.h	closed	(iv)
VWG/022/080792	fnmatch	closed	(i)
VWG/023/080792	putenv	closed	(iv)
VWG/024/080892	FIPS requirements	closed	(vi)
VWG/025/081892	Feature groups	closed	(ii) ** ***
VWG/026/081892	y0,y1,yn	closed	(ii) *
VWG/027/310892	Encoding (i) (commands)	closed	(vii)
	(ii) Codesets & the Kernel	OPEN	
VWG/028/310892	vprintf	closed	(vi)
VWG/029/160992	_POSIX2_VERSION	closed	(ii) *
VWG/030/170992	fseek() on fifo	See IEEE interp	(i)*****

Continued overleaf......

Notes * = Fixed in VSX4.2.4;
** = partially fixed in VSX4.2.4, please refer to VSX release notes.
*** = fixed in VSX4.3.2
**** = fixed in VSX4.3.4
***** = fixed in VSX4.3.5

Reference	Description	Status	Rationale
VWG/031/091192	shmget	closed	(iv)
VWG/032/091192	fseek() on fifo	closed	(ii) ***
VWG/033/191192	i_noncanon	closed	(iv)
VWG/034/111292	strtol,strtoul	closed	(vii)
VWG/035/221292	alarm	closed	(vi)
VWG/036/022093	more -t,vi -t,ex -t	closed	(vii)
VWG/037/022093	vi ˆQ	closed	(i)
VWG/038/022093	xargs	closed	(vii)
VWG/039/022093	namespace	closed	(i)
VWG/040/031793	fscanf()	closed	(i)
VWG/041/031793	chroot()	closed	(ii) ***
VWG/042/031993	hash	closed	(vii)
VWG/043/032593	GTI Support for networked connections	closed	(i)
VWG/044/050793	strftime,strptime padding	closed	(vii)
VWG/045/050793	strftime,strptime %C	closed	(vii)
VWG/046/050793	era handling	closed	(vii)
VWG/047/062993	semctl()	closed	(iv)
VWG/048/062293	ps	closed	(vii)
VWG/049/062293	od	closed	(vii)
VWG/050/072293	getconf	closed	(vii)
VWG/051/082093/	sccs	closed	(vii)
VWG/052/082193/	character order vs. collation sequence	closed	(vii)
VWG/054/092993/	getlogin()	closed	(vii)
VWG/055/100693/	nl_langinfo()	closed	(vii)
VWG/056/101993/	what	closed	(vii)
VWG/057/102193/	compress	closed	(vii)
VWG/059/110293/	fgets()	closed	(i)
VWG/060/110293/	fprintf()	closed	(i)*****
VWG/061/110293/	fprintf()	closed	(vii)
VWG/064/011994/	sact	closed	(vii)
VWG/067/033094/	Enhanced I18N Feature group	closed	(i)*****
VWG/070/052594/	D_T_FMT	closed	(i)*****
VWG/071/061994/	iconv	closed	(iv)

VWG Reference	Description	PIN Reference
VWG03	regcomp()	PIN4.039
VWG05	last close() on terminal	PIN4.030
VWG12	cfsetospeed, cfsetispeed	PIN4.009
VWG17(ii)	PATH_MAX	PIN4.006
VWG23	putenv()	PIN4.007
VWG29	_POSIX2_VERSION	PIN4.005
VWG31	shmget()	PIN4.038, PIN4.042
VWG33	i_noncanon	PIN4.008
VWG67	Enhanced I18N Feature group	PIN4.002, PIN4.023, PIN4.024,
		PIN4.025, PIN4.033
		PIN4.034, PIN4.035, PIN4.036
VWG71	iconv()	PIN4.029

4. Interpretations Text

4.1 Interpretation Request ref: VWG/001/053192

Publication:	XSH4
Interface:	system()
Date requested:	05.31.92
Status:	Closed

[***Request text starts***]

VSX4 review comment:

The test suite expects *system()* to return ECHILD as a possible error condition but this does not seem possible according to the text in XSH4.

VSX4 developer's response:

The tests are not derived from the ERRORS section, but from the last sentence of the "RETURN VALUE" section:

"*If a child process cannot be created, or if the termination status for the command language interpreter cannot be obtained, system() returns -1 and sets errno to indicate the error.*"

XSH4 states in section 2.3:

"*Implementations will not generate a different error number from the ones described here for error conditions described in this document*"

The error conditions in question are described in the document under fork() and wait(), and the errno values required in those cases are EAGAIN and ECHILD respectively.

[***text ends***]

X/OPEN RESOLUTION:

The test suite is correct. It is understood that POSIX 1003.2 allows ECHILD to be returned as an error.

Test Suite Status: (i)

The March 1994 meeting of XoTGbase agreed to address this change into the upcoming revision of XPG4 for SPEC1170.

4.2 Interpretation Request ref: VWG/002/060192

Publication:	XSH4
Interface:	mbstowcs(), wcstombs()
Date requested:	06.01.92
Status:	Closed

[***Request text starts***]

In the XSH4 extension text for *mbstowcs()* and *wcstombs()* it is not clear what the return value should be when the first argument is null and the third argument is less than the length required to convert the entire string.

One interpretation is that when the first argument is null, the function should behave the same as when it is not null except that no values are stored.

The other possible interpretation is that the return value should be the length required to convert the entire string, regardless of the value of the third argument.

[***text ends***]

X/OPEN RESOLUTION:

If the first argument is null, the functions should return the length required to convert the entire string regardless of the value of the third argument, but do not store any values.

Test suite status: (ii)

This was a known problem with VSX4.2.3 (wcstombs.3) and has been fixed in VSX4.2.4.

The March 1994 meeting of XoTGbase agreed to address this change into the upcoming revision of XPG4 for SPEC1170.

4.3 Interpretation Request ref: VWG/003/060392

Publication:	XBD/XSH4
Interface:	REs, regcomp()
Date requested:	06.03.92
Status:	Closed

[***Request text starts***]

VSX4 review comment:

Lines 382-385 on page 108 of the XBD state that if a left-brace is not part of a valid interval expression, then it produces undefined results. Thus, VSX4 should not expect *regcomp(preg,expr,REG_EXTENDED)* called with the regular expression "*xyab{3,}jk{*" to return 45 (*REG_EBRACE*), 2 (*REG_BADPAT*), or any values that are sensible.

VSX4 developer's response:

Where XBD states that behaviour is undefined, the behaviour can still be defined for a particular interface by a more specific statement elsewhere. For example XBD4 states on page 101, lines 98-100 that applications cannot rely on *\e{1,2,3\e}* producing an error, but XSH4 states quite clearly that *regcomp()* returns *REG_BADBR* when *\e{\e}* contains more than two numbers.

In the case of *{}* imbalance, it is not clear whether XSH4 specifies the behaviour for regcomp(). The *REG_EBRACE* return is specified for \e{\e} imbalance, but it is not clear whether this only applies to BREs, or to *{}* imbalance in EREs as well.

[***text ends***]

X/OPEN RESOLUTION:

The more specific requirement that "*REG_EBRACE is returned for \e{\e} imbalance*" overrules the more general description about undefined results. XPG4 is unclear whether this applies to EREs, so we should assume that it does not, thus undefined results are valid for EREs, but not BREs.

Test suite status: (vi)

4.4 Interpretation Request ref: VWG/004/061792

Publication:	XSH4
Interface:	strptime()
Date requested:	06.17.92
Status:	Closed

[***Request text starts***]

VSX4 reviewers comments:

The definition for *D_FMT* in the XBD gives *%m/%d/%y*. In XSH4 *strptime()* requires the conversion specifier *%c* to give the date and time, using the locale's date format. The specifier *%y* specifies the year within the century *[0,99]*. However, the test case for test 6 expects 1991 (%Y), not 91 for tm_year. There is also a dependency on the assumption in this test case that tm_yday isn't set as a side effect.

VSX4 Developers comments:

This appears to be an error in XSH4. The *%c* format should complement the *%c* format for *strftime()*, which uses *D_T_FMT*.

[***text ends***]

X/OPEN RESOLUTION:

There is an error in XSH4. The %c specifier should use the date and time format specifier. XSH4 states "*%c date and time using locale's date format (for example, as %x%X).*" The error here is that "*date format*" should be "*date and time format*".

The contents of the tm struct member tm_year should hold the year of the current century, and thus the test is incorrect in testing for 1991, when 91 should be expected.

Test suite status: (ii)

This was a known problem with VSX4.2.3 and has been fixed in VSX4.2.4.

The March 1994 meeting of XoTGbase agreed to address this change into the upcoming revision of XPG4 for SPEC1170.

4.5 Interpretation Request ref: VWG/005/063092

Publication:	XSH4
Interface:	close()
Date requested:	06.30.92
Status:	Closed"

+++++Text of IEEE Interpretation Request:

[***Problem description***]

last close() on terminal

[***Request text starts***]

POSIX.1-1990 section 7.1.1.11 Closing a Terminal Device file states:

The last process to close a terminal device file shall cause any output to be sent to the device and any input to be discarded.

Is it required behaviour that in the case that output has previously been suspended by a call to tcflow(), that the close() will allow output to be restarted or is it permissible behaviour for the data to be discarded?

X/Open proposes that close() will allow output to be restarted.

+++++End of IEEE Interpretation request.

X/OPEN RESOLUTION:

An IEEE interpretation was received.

Test Suite Status: (i)Fixed in VSX4.3.5.

4.6 Interpretation Request ref: VWG/006/063092

Publication:	XSH4
Interface:	tar
Date requested:	06.30.92
Status:	Open - IEEE Interpretation received

+++++Text of IEEE Interpretation Request:

[***Problem description***]

extra directories on tar archives

[***Request text starts***]

POSIX.1-1990 Section 10.1.1 Extended tar Format does not make it clear whether implementations are permitted to store extra directories on a tar archive. For example, when archiving "*/dev/devicename*", is it permissible for tar to place an entry for "*/dev*" on the archive?

X/Open proposes that it is not permissible for an extra entry to be made on the archive. Only by explicitly specifying the directories should they be placed on the archive. Only by explicit specification can the directory permissions be recorded. Directories created in the course of extracting from an archive will have process environment specified permissions if they were not explicitly put on the archive.

[***text ends***]

+++++End of IEEE Interpretation request.

X/OPEN RESOLUTION:

The X/Open proposed behaviour is a temporary interpretation pending resolution by the IEEE.

The IEEE interpretation was received in 1Q94. It states that the request is outside of the scope of IEEE Std 1003.1-1990, that POSIX specifies the format of the archive not any requirement on the format-creating utility, and that the behaviour specified in the request is not in conflict with POSIX.

Test Suite Status: (iii) - this should likely be moved to (iv) [ED]

4.7 Interpretation Request ref: VWG/007/063092

Publication:	XSH4
Interface:	sin(),cos(),tan()
Date requested:	06.30.92
Status:	Closed

[***Request text starts***]

Problem description:

partial loss of accuracy in sin, cos and tan

Tests in ANSI.os/maths/sin, cos and tan for conditions which will produce partial loss of accuracy expect these functions either to return a valid value and set errno to ERANGE, or to return NaN with errno either left unchanged or set to ERANGE.

[***text ends***]

X/OPEN RESOLUTION:

The test suite requires correction.

Test Suite Status: (ii) fixed in VSX4.2.4.

4.8 Interpretation Request ref: VWG/008/063092

Publication:	XSH4
Interface:	fcntl()
Date requested:	06.30.92
Status:	Open - IEEE Interpretation received

+++++Text of IEEE Interpretation Request:

[***Problem description***]

fcntl() locking

During the development of a test suite for ISO POSIX.1 the following question was raised with regard to the fcntl() function and coalescence of adjacent locks.

[***Request text starts***]

POSIX.1-1990 Section 6.5.2.2 fcntl() :

Should adjacent locks be coalesced when F_GETLK is used to check for existence of locks?

X/Open would propose that they should not be so:

In fcntl(), page 107 line 161ff of the System Interfaces and headers document it says:

"*There will be at most one type of lock for each byte in the file. Before successful return from an F_SETLK or an F_SETLKW request when the calling process has previously existing locks on bytes in the region specified by the request, the previous lock type for each byte in the specified region will be replaced by the new lock type.*"

Meaning that if two locks for overlapping regions of a file are obtained by a single process, the region which overlaps will be owned by the new lock. Coalescing is not required.

[***text ends***]

+++++End of IEEE Interpretation request.

X/OPEN RESOLUTION:

The X/Open proposed behaviour is a temporary interpretation pending resolution by the IEEE.

The IEEE interpretation was received in 1Q94. It states that the request is outside of the scope of IEEE Std 1003.1-1990, that POSIX neither requires or prohibits the behavior.

Test Suite Status: (iii)

4.9 Interpretation Request ref: VWG/009/060392

Publication:	XSH4
Interface:	namespace
Date requested:	06.30.92
Status:	Pending X3J11 interpretation

[***Request text starts***]

Problem description:

reserved identifiers beginning with underscore

Where symbols beginning with underscore are declared in headers as identifiers with external linkage, the test suite reports the symbols as illegal if the leading underscore is not followed by another underscore or an upper-case letter.

[***text ends***]

X/OPEN RESOLUTION:

Based on informal discussion with X3J11 members, the tests are incorrect and we recommend removing them in a future release. For VSX4.2.3 waivers would be granted.

Currently awaiting formal notification of a pending X3J11 interpretation.

Test Suite Status: (ii) This is fixed in VSX4.2.4.

4.10 Interpretation Request ref: VWG/010/063092

Publication:	XSH4
Interface:	write()
Date requested:	06.30.92
Status:	Closed

[***Request text starts***]

Problem description:

write of zero bytes with O_APPEND

The test for write() with third argument zero on a file opened with O_APPEND expects that the file position will not be changed.

[***text ends***]

X/OPEN RESOLUTION:

There has been no formal request by X/Open on this issue to the IEEE. However there has been finalised another P1003.1 interpretation request on the behaviour on read()'s and write()'s of zero length. This has been applied to this question with the outcome that the test suite is correct.

Test Suite Status: (i)

4.11 Interpretation Request ref: VWG/011/063092

Publication:	XSH4
Interface:	tzname
Date requested:	06.30.92
Status:	Closed

[***Request text starts***]

Problem description:

long timezone names in XPG3 mode

In XPG3 mode, tests of time functions which check that the function has set tzname[] correctly expect timezone names of 14 characters to be accepted, since XPG3 does not place any limit on timezone name lengths. Many waivers have been granted against VSX3 for these tests.

[***text ends***]

X/OPEN RESOLUTION:

Leave tests as they are, this is in-line with the current XPG3 branding tool. Waivers will continue to be granted (much the same as the situation for chroot with shared libraries).

Test Suite Status: (iv)

4.12 Interpretation Request ref: VWG/012/063092

Publication:	XSH4
Interface:	cfsetispeed(), cfsetospeed()
Date requested:	06.30.92
Status:	Closed

+++++Text of IEEE Interpretation Request:

[***Problem description***]

During the test of ISO POSIX.1-90 the following question with regard to the cfsetospeed() function was raised.

[***Request text starts***]

In the test of cfsetospeed() with speed B0, the test is performed on the controlling terminal rather than the other side of the loopback in order to simulate what a normal application would do. The test expects this to cause a SIGHUP, but on some systems this only occurs when a modem disconnect is detected, not when one is generated, so the test relies on "reflection" of the modem disconnect by the loopback connection on such systems, and if this does not happen the test fails.

X/Open believes POSIX says that SIGHUP is generated when disconnected is detected and not before.

[***text ends***]

+++++End of IEEE Interpretation request.

X/OPEN RESOLUTION:

The IEEE interpretation has been received in 1Q94. This states that it is unspecified whether a modem disconnect is detected immediately when the output baud rate of the local terminal is set to B0. Since this is unspecified, it is conforming (but not required) for the local asynchronous serial communications port to perform a complete communications disconnect when its speed is changed to B0 without monitoring the response from the remote port. In this case the modem disconnect would be performed immediately, and if the local port were a controlling terminal the SIGHUP signal would be generated immediately for the controlling process.

Test Suite Status: (i) Fixed in VSX4.3.5

4.13 Interpretation Request ref: VWG/013/063092

Publication:	XSH4
Interface:	getopt()
Date requested:	06.30.92
Status:	Closed

[***Request text starts***]

Problem description:

getopt missing argument

XPG4.os/genuts/getopt test 1 expects that when a 'missing option argument' error condition occurs where the last argument contains the option whose argument is missing, then optarg will be set to point to the null terminator of the argument list. Some implementations do not alter optarg under these conditions (i.e. it remains pointing to whatever it pointed to before the getopt() call).

[***text ends***]

X/OPEN RESOLUTION:

Leave the test as it is. The XPG4 specification and POSIX.2 B.7.2 agree that the optarg is moved and optind is incremented by two.

Test Suite Status: (i)

4.14 Interpretation Request ref: VWG/014/063092

Publication:	XSH4
Interface:	time stamps
Date requested:	06.30.92
Status:	Closed

+++++Text of IEEE Interpretation Request:

[***Problem description***]

timestamps on read-only file systems

Tests for read() and the stdio reading functions expect that the st_atime field is not updated on a read-only file system. Some systems do update the st_atime of the in-core inode although the change is never made on disc. It is not clear whether this behaviour is allowed by POSIX.1.

[***Request text starts***]

POSIX.1-1990 Section 2.2.2.69 defines "read only filesystem":

A file system that has implementation defined characteristics restricting modification.

POSIX.1-1990 Section 2.3.5 "file times update" states

Updates are not done for files on read-only file systems.

Is it permissible for an implementation to update the st_atime attribute held in-core but yet prevent the update of that attribute for the file on the read-only filesystem?

The X/Open proposed resolution is that POSIX.1 does allow for Read-Only file systems, but also requires all reads to update st_atime. It is our opinion that what was meant was that st_atime would not be updated for Read-Only file systems.

[***text ends***]

+++++End of IEEE Interpretation request.

X/OPEN RESOLUTION:

The IEEE has ruled that the behaviour is non conforming. The test suite is correct.

Test Suite Status: (i)

4.15 Interpretation Request ref: VWG/015/063092

Publication:	XSH4
Interface:	bytes, chars and multibyte chars
Date requested:	06.30.92
Status:	Closed

[***Request text starts***]

VSX4 test reviewers comments:

We think printf (%.7S) should print the first 7 wide chars not print a string that contain 7 bytes"

XSH4/XBD Reviewers Response:

The issues revolves around the definition of byte, and character. Looking at the XBD it states:

‘‘*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...*

character: a sequence of one or more bytes representing a single graphic symbol or control code. This term corresponds in the ISO C standard to the term multibyte character, noting that a single-byte character is a special case of a multibyte 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’’

Note: the different semantics for byte and character with respect to storage space. sp Also the XPG states that "*All wide character codes in a given process consist of an equal number of bits. This is in contrast to characters, which can consist of a variable number of bytes. The byte or code sequence that represents a character can also be represented as a wide character code.*"

and that "*wchar_t - Integral type whose range and values can represent distinct wide character codes for all members of the largest character set specified among locales supported by the compilation environment*"

Nowhere does the XPG state that a byte and a wchar_t must be the same, and they only can be the same if the characters of all largest supported character set can be represented in a byte.

Looking now to the XSH4 fprintf() states

‘‘*S The argument must be a pointer to an array of type wchar_t. Wide character codes from the array, ... are converted to a sequence of bytes, and the resulting bytes are written. ...... If the precision specified no more than that many bytes are written and only complete characters are written. .... The conversion is the same as that expected from the wcstombs() function.*

RETURN VALUE

Upon successful completion, these functions return the number of bytes transmitted .....’’

wcstombs() states:

‘‘*....the wcstombs() function returns the number of bytes stored in the character array...*’’

The key here is "*the conversion is the same as that expected from the wcstombs() function*", this is exactly as the tests expect.

We should also consider the situation if it were allowed that a precision specifier of %.7S could result in 7 wide characters being translated to multibyte characters and output to a character string via sprintf. The application would not have any means of determining the number of bytes that each of the multibyte characters would require and, therefore, no means of determining the size of the buffer required. The whole purpose of the precision specifier is to allow the application to take control of these type of sizing and positioning aspects and this interpretation negates this capability. Application portability would be broken if this were allowed.

[***text ends***]

X/OPEN RESOLUTION:

Accept existing tests.

Test Suite Status: (i)

4.16 Interpretation Request ref: VWG/016/070792

Publication:	XSH4
Interface:	Header files
Date requested:	07.07.92
Status:	Closed

[***Request text starts***]

VSX4 test reviewers comments:

During VSX testing the following namespace problems occurred with the header files. Was it intentional to restrict the namespace in this manner for the following headers.

- *regexp.h*

  Do the namespace rules apply to *regexp.h* which has been marked *to be withdrawn*? This will cause problems for a large number of historical implementations which have supplied function code in the headers. Such implementations have defined a large number of symbols that would not be allowable in the XPG4 namespace.

  We would recommend that since this interface has been marked to be withdrawn, that the XPG3 requirements should suffice for this header, which would mean that the namespace rules be relaxed.

- *sys/msg.h*

  Historical practice has been to define a structure of name *msg* to represent an ipc message. Is this permissible for XSH4? Section 2.2.1 of XSH4 implies this is not permissible, but since many implementations have been implemented in this way, we would request that it be allowed.

- *sys/sem.h*

  Historical implementations have defined a structure of name *sem* to represent an ipc semaphore. Is this permissible for XSH4? Section 2.2.1 of XSH4 implies this is not permissible, but since many historical implementations have been implemented in this way, we would request that it be allowed.

- *ulimit.h*

  Can implementations provide additional symbols beginning UL_ ? Many other headers have allowable extensions to the symbol namespace, why should <ulimit.h> be restricted (cf to nl_types.h)?

Reviewer comments:

According to the namespace rules defined in XSH4 the above are not allowed.

[***text ends***]

X/OPEN RESOLUTION:

i. The *regexp.h* header is allowed dispensation from namespace rules since this conflicts with existing behaviour.

ii. The additional symbol *msg* is permissible in <sys/msg.h>.

iii. The additional symbol *sem* is permissible in <sys/sem.h>

iv. Only the symbols specified in the XSH4 manual page for <ulimit.h> are allowed.

Test Suite Status: (ii) This has been fixed in VSX4.2.4.

4.17 Interpretation Request ref: VWG/017/070792

Publication:	XSH4
Interface:	ENAMETOOLONG/POSIX_NO_TRUNC
Date requested:	07.07.92
Status:	Closed

[***Request text starts***]

VSX4 Test reviewers comments:

VSX generates a long pathname in order to test for the *ENAMETOOLONG* error condition for various interfaces. The long pathname generated by the test setup routine should take into account that normal pathname resolution may reduce the pathname to a length less than *PATH_MAX* which then doesn't satisfy the criteria for a name to test *ENAMETOOLONG*.

VSX Developers response:

POSIX.1-1990 requires the *ENAMETOOLONG* error when the length of a pathname argument exceeds *PATH_MAX*. It says nothing about the length "after pathname resolution", but refers explicitly to the contents of the argument.

VSX Test Reviewers further response:

Different functions say different things but they all basically say "*The length of the path argument (or string) exceeds {PATH_MAX}, or a pathname component is longer than {NAME_MAX} while {_POSIX_NO_TRUNC} is in effect.*" The key here is this is a *_POSIX_NO_TRUNC* issue. The purpose of *_POSIX_NO_TRUNC* is to determine if the implementation treats truncation of filenames as an error or not. This implementation treats filename truncation as an error but the above filename does not require truncation and is therefore not an error. We believe in order for this test to be valid the filename used must not be representable on the filesystem in question.

Interpretation Coordinators Comments:

There are two issues here:

i. Whether path resolution occurs before a pathname is checked for ENAMETOOLONG

ii. The definition of PATH_MAX is again at issue.

[***text ends***]

X/OPEN RESOLUTION:

(i) pathname resolution

Seek a POSIX interpretation for the issue ‘‘*can a pathname be shortened by normal pathname resolution prior to checking for ENAMETOOLONG*’’.

+++++Text of IEEE Interpretation Request:

[Problem description]

ENAMETOOLONG and pathname resolution

[***Request text starts***]

Can a pathname be shortened by normal pathname resolution prior to checking for ENAMETOOLONG?

X/Open proposes that it does not. It is our opinion that POSIX.1-1990 requires the ENAMETOOLONG error when the length of a pathname argument exceeds {PATH_MAX}. It says nothing about the length "after pathname resolution", but refers explicitly to the contents of the argument.

[***text ends***]

+++++END of IEEE Interpretation Request:

Grant a temporary interpretation to allow for this behaviour.

Test Suite Status: (iii) Retain as documented grey area until resolved. No change to test suite at this time. Waivers would be granted under "temporary interpretation" category.

(ii) PATH_MAX

The definition of PATH_MAX was debated at length during the development of XSH4. In the end it was decided to align with the draft POSIX 1003.1a. However since this is still a draft standard, for testing we should align with POSIX.1-1990 as published and interpreted. Therefore, we should allow either behaviour.

Test Suite Status: (iv) This is a permanent grey area. The tests will be left as they are, this way, if tests fail awareness of this issue will be raised, and implementors can then make the choice to align either with the draft P1003.1a or to apply for a waiver which would be granted.

4.18 Interpretation Request ref: VWG/018/070892

Publication:	XSH4
Interface:	wcswidth()
Date requested:	07.08.92
Status:	Closed

[***Request text starts***]

VSX4 Test reviewers comments:

wcswidth() takes two arguments, the pointer to the wide character string and the number of wide character codes whose screen width is to be determined.

One of the return values is -1 : if any of the wide-character codes in the wide character string is not a printing wide character.

Should an implementation check for non-printing characters after the *nth* code, or is it allowable to terminate the processing at the *nth* code, and return the screen width for the *n* characters. The existing test in VSX expects the scan to detect non-printing chars after the *nth* character and return -1.

Which is the correct behaviour?

Reviewer comments:

The text in XSH4 would seem to suggest that interpretation (i) below is required, rather than interpretation (ii).

i. wcswidth() should process the whole string beyond n characters and return -1 if any character is non-printing

ii. wcswidth() process only n characters, and ignore rest of string.

[***text ends***]

X/OPEN RESOLUTION.

Accept interpretation (ii). This will require a change to the test suite.

Test Suite Status: (ii) This has been fixed in VSX4.2.4.

The March 1994 meeting of XoTGbase agreed to address this change into the upcoming revision of XPG4 for SPEC1170.

4.19 Interpretation Request ref: VWG/019/071492

Publication:	XSH4
Interface:	iconv()
Date requested:	07.14.92
Status:	Closed

[***Request text starts***]

XSH4 reviewers comments:

The return value for *iconv()* is defined as the number of non-identical conversions made. Also, if an error occurs, *iconv()* returns -1.

The question is: when should *iconv()* return -1 and when should it return the number of non-identical conversions made.

The situation is clear when the output buffer is too small when *iconv()* is called to get the byte sequence needed to reset *iconv()*'s shift state and also when the cd argument is invalid; -1 should be returned in these cases.

When another error occurs though (for example an illegal character or the output buffer is too small) and the input is only partially processed, it seems silly to return -1 because the number of non-identical conversions may have some meaning and be useful.

[***text ends***]

X/OPEN RESOLUTION:

If the number of conversions was returned for both successful and non-successful calls, error checking would be very inconsistent: the programmer would see a return of -1 for a bad code for the first conversion, but a count for a bad code for subsequent conversions.

When some partial processing of the input has occurred before an unrecognised code is encountered, *iconv()* should return -1 and the input and output pointers should reflect the conversions already done. The input pointer should point to the erroneous code. There is no need to return a count of the conversions done. Since errno is set, an application should check errno when -1 is returned.

Test Suite Status: (vi)

4.20 Interpretation Request ref: VWG/020/071492

Publication:	XSH4
Interface:	langinfo.h / CRNCYSTR
Date requested:	07.14.92
Status:	Closed

[***Request text starts***]

VSX4 Test reviewers comment:

The documented requirements in the VSX installation guide for *CRNCYSTR* in the VSX installation guide and *currency_symbol* differ and make it impossible to pass both the *setlocale* and *localeconv* tests. Implementations derive the return value of *nl_langinfo(CRNCYSTR)* and the *localeconv* member *currency_symbol* from the same language database item *currency_symbol*.

VSX4 developers response:

XSH4 requires that *CRNCYSTR* contains a leading '+' or '-' to indicate positioning of the currency string, and that the positioning of *currency_symbol* is controlled separately for positive and negative values by *p_cs_precedes* and *n_cs_precedes*. If these two values differ, the implementation cannot determine whether to use a leading '+' or '-' in *CRNCYSTR*.

XSH4 reviewers comments:

The problem seems to be in the specification of *CRNCYSTR* in *langinfo.h*. The "+" and "-" components of the *CRNCYSTR* are defined as preceding the currency symbol, and denote whether the system should put the currency symbol before or after the value. The problem is that *localeconv()* allows the currency symbol to vary its position , according to whether it is a positive or negative value.

This causes problems in the setup of the VSX4 pseudolanguages. The pseudolangs *VSX4L1* and *VSX4L3* define *p_cs_precedes* and *n_cs_precedes* to 1 (to denote that the value precedes) [there is no ambiguity in this case]. In pseudolangs *VSX4L2* and *VSX4L3@dict p_cs_precedes* is 0 and *n_cs_precedes* is 1.

Some implementations of *nl_langinfo* look at the *p_cs_precedes* field of the *lconv* structure to determine whether the symbol is preceding or not, and then attaches "+" or "-" respectively, these systems have no problem with the existing setup. However should an implementation look at *n_cs_precedes* they would get different behaviour.

[***text ends***]

X/OPEN RESOLUTION:

There is an ambiguity in the specification. The VSX test setup should be changed so as to work around this ambiguity.

Test Suite Status: (ii) This has been fixed in VSX4.2.4.

The March 1994 meeting of XoTGbase agreed to address this change into the upcoming revision of XPG4 for SPEC1170. Further changes may be required in a future revision to remove CRNCYSTR.

4.21 Interpretation Request ref: VWG/021/080792

Publication:	XSH4
Interface:	wchar.h
Date requested:	08.07.92
Status:	Closed

[***Request text starts***]

VSX4 Test reviewers comment:

The prototype definitions of wcschr(), wcsrchr() and wcwidth() differ between <wchar.h> and the individual manual pages for these interfaces.

Interface	Man page	wchar.h
wcschr() arg2	wint_t wc	wchar_t wc
wcsrchr() arg2	wint_t wc	wchar_t wc
wcwidth() arg1	wint_t wc	const wchar_t wc

[***text ends***]

X/OPEN RESOLUTION:

The XSH4 manual page for <wchar.h> is inconsistent with the prototypes for wcschr(), wcsrchr() and wcwidth() on their individual manual pages.

These should be aligned in a future revision of the XPG with the prototypes as defined in the final ISO MSE.

For XPG4 either version of the prototype should be allowable.

Test Suite Status:

(iv) There are no tests for this at present. If in future any are developed they should take the above into account and document this issue as a permanent grey area.

The March 1994 meeting of XoTGbase agreed to address this change into the upcoming revision of XPG4 for SPEC1170.

4.22 Interpretation Request ref: VWG/022/080792

Publication:	XSH4
Interface:	fnmatch
Date requested:	08.07.92
Status:	Closed

[***Request text starts***]

VSX4 Test reviewers comment:

VSX4 XPG4.os/genuts/fnmatch test 1 expects a slash character in a bracket expression to explicitly match a slash character in a filename when FNM_PATHNAME is not set. The test case used is:

fnmatch("/usr/subdir[-./]file123", "/usr/subdir/file123", 0)

The XSH4 fnmatch description references the XCU discussion of pattern matching notation in sections 2.13.1 and 2.13.2. These sections are qualified by section 2.13.3. This last section states:

"*The slash character in a pathname must be explicitly matched by using one or more slashes in the pattern .... a slash cannot be included in a pattern bracket expression used for filename expansion ....*"

However, the XSH4 fnmatch discussion states:

"*If the FNM_PATHNAME flag is not set, the slash character is treated as an ordinary character.*"

Is it the intent of these statements to make a slash character legal in a bracket expression when FNM_PATHNAME is not set ?

VSX4 Developers comments:

It would seem that the intention is for the statement in XSH4 to modify the requirements made by the text it refers to in XCU4. If that were not the case, the statement in XSH4 would be redundant. The test is right to expect the slash to be treated as an ordinary character.

[***text ends***]

X/OPEN RESOLUTION:

In XCU4 section 2.13.3, the rules in 2.13.1 and 2.13.2 are qualified by the following rules when pattern matching notation is used for filename expansion. Since in the test case, FNM_PATHNAME is not set, the pattern matching notation is not used for filename expansion, therefore the test case and XSH4 are correct.

Test Suite Status: (i)

4.23 Interpretation Request ref: VWG/023/080792

Publication:	XSH4
Interface:	putenv
Date requested:	08.07.92
Status:	Closed

[***Request text starts***]

XSH4 reviewers comment:

There is a problem with the prototype for putenv() which is defined as:

```
int     putenv(const char *string);
```

The problem is that the address passed to putenv() lives beyond the life of the call in question.

[*This request has been abridged -Ed*]

[***text ends***]

X/OPEN RESOLUTION:

The prototype for putenv should be changed in a future revision of the XPG from

*int putenv(const char *string);*

to

*int putenv(char *string);*

For XPG4 either definition is allowed.

Test Suite Status:

(iv) This is a permanent grey area. The tests should be left as they are, and this handled through the waiver system.

[The VSX4 test is ANSI.hdr/misc/stlib_X.55]

The March 1994 meeting of XoTGbase agreed to address this change into the upcoming revision of XPG4 for SPEC1170.

4.24 Interpretation Request ref: VWG/024/080892

Publication:	XSH4
Interface:	FIPS requirements
Date requested:	08.08.92
Status:	Closed

[***Request text starts***]

VSX4 Test Reviewers Comments:

Many tests assume that that gid of a file created will be the gid of the parent directory , this is a FIPS requirement and should be only tested in FIPS mode. This should not be applicable to the XPG4 mode of the test suite.

VSX4 Developers comments:

Section 1.6 states that XSH4 has been aligned with the draft FIPS 151-2. Section 1.7 lists the extensions required by FIPS 151, including the group ID requirement. From these two sections the obvious conclusion a reader will make is that XSH4 requires all the listed FIPS 151 extensions.

[***text ends***]

X/OPEN RESOLUTION:

XSH4 requires support of all FIPS requirements. Hence leave the tests as they are. This requires the test system to be setup for test purposes in both XPG4 and FIPS modes in an environment which supports the creation of files with the group-id of the parent directory.

Typically this is done by a procedure such as setting the setgid bit on the parent directory.

Test Suite Status: (vi) No change required. Leave the tests as they are.

4.25 Interpretation Request ref: VWG/025/081892

Publication:	XSH4
Interface:	Feature Groups
Date requested:	08.18.92
Status:	Closed

[***Request Text starts***]

There seems to have been some confusion over the definition of Feature groups and the way they have been tested by VSX4.

[***text ends***]

X/OPEN RESOLUTION:

The feature groups requirements are as follows:

- If a feature group macro is set to other than -1 then all the functions in that group must be present as defined in XSH4. If an interface is not supported it shall indicate an error as defined on the individual manual page. For many of the functions this is done by setting errno to ENOSYS, however some functions have different return values; for example the POSIX2_CLB functions have the return values WRDE_NOSYS, FNM_NOSYS and REG_ENOSYS defined.
- If {_XOPEN_ENH_I18N} is set to -1, some of the functions in this group may be supported. This is to encourage convergence of implementations.
- If {_XOPEN_SHM} is set to -1, none of these functions are supported.
- If {_XOPEN_CRYPT} is set to -1, none of these functions are supported. Note that, the encrypt() function is explicitly permitted to fail and set errno to ENOSYS even when {_XOPEN_CRYPT} is set to something other than -1, when decryption is requested.
- If {_POSIX2_C_VERSION} is set to -1, then some of the functions in the POSIX2 CLB may be supported, in this case it is recommended that 'family' groups within the POSIX2 CLB be supported, i.e.

  glob(), and *globfree()*
  and/or
  regcomp(), *regerror()*, *regexec()* and *regfree()*
  and/or
  wordexp(), *wordfree()*
  and/or
  fnmatch()

  however, this is not a requirement.
- If a function is supported (even though all the Feature Group may not be supported) it shall behave consistently with the relevant description in the XSH4.

Test Suite Status:

(ii) Document as known problem. Currently VSX expects all or none of the feature group routines to be supported. This is not a requirement.

A future revision of the test suite will re-organise the tests in-line with the resolution not to assume such a grouping and also to check interfaces are consistent with the XSH4 if implemented when their Feature Group is not supported. This has been partly addressed by VSX4.2.4, please refer to the VSX release notes.

The March 1994 meeting of XoTGbase agreed NOT to address this change into the upcoming revision of XPG4 for SPEC1170, but to address this in a future revision.

4.26 Interpretation Request ref: VWG/026/081892

Publication:	XSH4
Interface:	y0,y1,yn
Date requested:	08.18.92
Status:	Closed

[***Request text starts***]

VSX4 Test Reviewers Comments:

It appears that the y0 test

> Test XOPEN.os/maths/y0. 9 fails as follows
> RETURN VALUES: expected: 0, observed: -4.7535e-156
> ERRNO VALUES: expected: 34 (ERANGE), observed: 0 (NO ERROR)
>
> Typical implementations with DBL_MAX of approx 1.8e+308 give ERANGE
> at about y0(1.0e+8).

is testing for PLOSS/TLOSS behavior when the System Interfaces and Headers does not allow it. Hence the VSX test for y0 needs to be corrected.

Reason:

There is no Return Value clause requiring a return value of 0 and errno to be set to [ERANGE] for large arguments for y0, y1, and yn.

[***text ends***]

X/OPEN RESOLUTION:

These tests for y0, y1, yn should be removed from VSX4. In a future revision of the XPG it is recommended that the phrase discussing "too large in magnitude" be deleted.

Test Suite Status: (ii) Fixed in VSX4.2.4.

4.27 Interpretation Request ref: VWG/027/310892

Publication:	XBD/XSH4
Interface:	encoding
Date requested:	08.31.92
Status:	See below

[***Interpretation Coordinator's Note***]

This request been divided into 2 problems to ease resolution; the current status is as follows:

i. Encoding (commands) - Closed

ii. Codeset & the Kernel - OPEN

[***Request text starts***]

XSH4/XBD Reviewers Comments:

On page 36 of System Interface Definitions (XBD4), in section 4.2 the first bullet item reads:

- *A single-shift encoding (where each character not in the initial shift state is preceded by a shift code) can be defined if each shift-code and character sequence is considered a multi-byte character. ... If the implementation supports a character encoding of this type, all of the standard utilities in the XCU specification will support it. Use of a single-shift encoding with any of the functions in the XSH specification that do not specifically mention the effects of state-dependent encoding is implementation-dependent.*

What is the precise meaning of "*all the standard utilities in the XCU specification will support it*"?

Utility support for a multibyte codeset might include:

- the sh utility accepting commands and performing correct IFS splitting on all words in that codeset, but switching its behavior when certain of the LC_* or LANG environment variables are changed
- the iconv utility converting from that codeset to some unspecified other codeset
- supporting these characters in file names (because of sh), and thus probably in the kernel
- handling data in such a codeset without destroying it
- being able to create and manipulate files of such data with vi
- being able to search for patterns containing multibyte characters in grep, vi, awk, and sed
- supporting multi-column characters in the curses functions

Is the intention that if an implementation supports XCU4 and any single-shift encoding at all, then all of the above are required?

On what environment variable does the codeset for each of these features depend? In other words, how does the kernel tell whether to look for ASCII file names or Kanji file names?

Is the intention that if an implementation supports XSH4 and any single-shift encoding, then XCU4 conformance and all of the above are required?

[***text ends***]

X/OPEN RESOLUTION:

i. If the implementation supports a single-shift encoding then all of the standard utilities will support it , where "all of the standard utilities" is the utilities defined in XCU4 minus the optional (dis), possibly unsupportable and withdrawn utilities (see XCU4 pages 2-4).

   It is the intention that if a system supports XCU4 and any single shift encoding that the following requirements are met:

   - the sh utility accepts commands and performs correct IFS splitting on all words in that codeset. Note that according to XCU4, Page 29, changes to the LC_CTYPE category (through LC_CTYPE, LANG or LC_ALL) do not affect the IFS splitting of the current shell. New started shells or shell-scripts will be affected.
   - if more than one codeset is supported the iconv utility may support conversion from that codeset to other codesets
   - supporting these characters in file names (because of sh)
   - handling data in such a codeset without destroying it
   - being able to create and manipulate files of such data with vi
   - being able to search for patterns containing multibyte characters in grep, vi, awk, and sed

   Test Suite Status: (vii) not applicable, there are no tests at the present time.

ii. On what environment variable does the codeset for each of these features depend? In other words, how does the kernel tell whether to look for ASCII file names or Kanji file names?

   [***Interpretation Coordinator Note***: a proposed resolution did not clarify the kernel behaviour. We are still seeking a proposal to agree upon for this question.]

4.28 Interpretation Request ref: VWG/028/310892

Publication:	XSH4
Interface:	vfprintf
Date requested:	08.31.92
Status:	Closed

[***Request text starts***]

XSH4 Reviewers Comments:

Is the header file required for the vfprintf(), vprintf(), and vsprintf() functions called <varargs.h> or <stdarg.h>?

Since ISO C requires the use of <stdarg.h>, we assume the SYNOPSIS section on page 499 of XSH4 was incorrectly changed at the last minute to require the obsolescent <varargs.h>.

[***text ends***]

X/OPEN RESOLUTION:

This is a fault in the XSH4 specification. The header should be <stdarg.h>. This should be corrected in a future revision of the XPG.

Test Suite Status: (vi) No change , there are no applicable tests.

The March 1994 meeting of XoTGbase agreed to address this change into the upcoming revision of XPG4 for SPEC1170.

4.29 Interpretation Request ref: VWG/029/160992

Publication:	XSH4
Interface:	_POSIX2_VERSION
Date requested:	09.16.92
Status:	Closed

[***Request text starts***]

VSX4 Reviewers comments:

There seems to be some confusion over the value of _POSIX2_VERSION and the return from sysconf(_SC_2_VERSION) for systems supporting the historical commands profile.

The test suite expects sysconf(_SC_2_VERSION) to return the same value as _POSIX2_VERSION. This does not seem to be the requirement in XPG4.

[***text ends***]

X/OPEN RESOLUTION:

- *sysconf(_SC_2_VERSION)* should return a value not more restrictive than the value of *_POSIX2_VERSION*;
- if _POSIX2_VERSION is -1, sysconf(_SC_2_VERSION) must also be -1.
- if _POSIX2_VERSION is undefined, sysconf(_SC_2_VERSION) may return -1 .
- _POSIX2_VERSION when set to the value -1 indicates that the XCU4 commands (and hence the POSIX 1003.2 DIS) are not supported on this implementation.
- The correct value for _POSIX2_VERSION for systems supporting XCU4 and hence the POSIX 1003.2 DIS is 199209L.

Test Suite Status: (ii) Fixed in VSX4.2.4

4.30 Interpretation Request ref: VWG/030/170992

Publication:	XSH4
Interface:	fseek
Date requested:	09.17.92
Status:	Closed

+++++Text of IEEE Interpretation Request:

[Problem description]

fseek() on a pipe or FIFO

A question arose during development of a POSIX.1-90 test suite. This is related to the following assertion from Draft 13 of POSIX 1003.3.1:

When the underlying open file descriptor references a pipe or FIFO, then a call to fseek() sets errno to ESPIPE, returns a non-zero value and the value of the file pointer is unchanged.

This particular issue has uncovered many ambiguities within POSIX and within many implementations. The following is a list of questions and some insights that may have a bearing on the issue.

[***Request text starts***]

(1) Is a pipe or FIFO a device or a special file type?

There seems to be some sloppy wording in POSIX regarding "devices which are incapable of seeking" and whether these include pipes and FIFOs. The definition of a device is "A computer peripheral or an object that appears to the application as such" and that of a FIFO is "A FIFO special file is a type of file". We understand from these definitions that a pipe or a FIFO cannot be classified as a device. If this is incorrect, then the behaviour of fseek() on a pipe or FIFO is implementation defined and just about any behaviour is acceptable after an attempt to perform a file positioning operation on a pipe or FIFO. We understand that the behaviour of a file positioning operation on a pipe or FIFO is well defined to indicate an ESPIPE error.

(2) How does a file positioning operation on a pipe or FIFO affect subsequent read operations?

POSIX states in the definition of file offset "There is no file offset specified for a pipe or FIFO". In tests for lseek() this has been understood by the various test suite developers to mean that subsequent reads from the pipe or FIFO are unaffected by the attempted lseek(), and this test has not given any problems.

POSIX.3.1 Draft 13 has extrapolated this text for lseek() into the equivalent assertion for fseek(). This extrapolation has a further problem in that a stream does not have file offset but has a file-position indicator. So does the POSIX.3.1 assertion really refer to the file position indicator or to the file pointer. The X/Open POSIX.1-90 test suite understands this statement to be the equivalent of the statement for lseek(), in that subsequent reads from the stream are unaffected by the attempted fseek(). This test, however, is exhibiting problems and is the subject for this interpretation.

(3) Could the assertion be refering to the underlying file descriptor?

If this were the case, then it would be necessary to access the pipe and FIFO both from a stream (to undertake the fseek()) and from a file descriptor (to undertake the read()) to ensure that data loss did not occur at the file descriptor level. Unfortunately, the synchronisation rules for file handles does not provide a mechanism for handing off file handles for buffered pipes. This makes it impossible to produce a POSIX conforming application to test this understanding of the assertion, though this doesn't seem to deter all test suite authors! The fact that such an assertion results in a pass is not entirely a surprise and only serves to prove how well behaved implementations are when the behavior is undefined.

(4) Could the assertion be refering to the continued ability to read data from the stream without any side-effects?

The X/Open POSIX.1-90 test suite believes this to be the case. While it is accepted that the POSIX does not explicitly state that this is the case, it seems valid to argue that side-effects that exhibit data loss after an error condition has been raised should be documented. This data loss seems particularly dangerous since there is no means of recovery once an application has erroneously undertaken an fseek() on a stream. However, these arguments have to be balanced against the fact that there is no explicit mention in the POSIX and in such cases the implementation could be argued to have a degree of freedom.

The X/Open proposed resolution is:

An fseek() on a pipe or FIFO should return ESPIPE and results in undefined behaviour to the stdio stream.

[***text ends***]

+++++Text of IEEE Interpretation Request:

X/OPEN RESOLUTION:

The IEEE interpretation has been received.

Test Suite Status: (i) Fixed in VSX4.3.5.

[The VSX4 test is ANSI.os/streamio/fseek.12]

4.31 Interpretation Request ref: VWG/031/091192

Publication:	XSH4
Interface:	shmget
Date requested:	09.11.92
Status:	Closed

[***Request text starts***]

XSH4/VSX4 Reviewers Comments:

The interfaces *shmget()* has a different interface in XSH4than what is generally used in the industry. In particular, the second parameter is: size_t size. In SVID this is: int size. So in systems where size_t is not an int, compatibility and portability is lost. We would request that the second parameter be changed to an *int*.

[***text ends***]

X/OPEN RESOLUTION:

Either definition should be allowed for XPG4. For systems using "int" instead of "size_t" a waiver would be granted.

Test suite status: (iv)

4.32 Interpretation Request ref: VWG/032/091192

Publication:	XSH4
Interface:	fseek
Date requested:	09.11.92
Status:	Closed

[***Request text starts***]

XPG3/VSX4 Reviewers Comments:

VSX4 is proposing to retain the test of *fseek* on a *FIFO* in the XPG3 mode of the test suite as well as XPG4.

We would request an interpretation that this test is not required in the XPG3 mode.

[***text ends***]

X/OPEN RESOLUTION:

The test should be "removed" from the XPG3 mode of the VSX4 test suite at the earliest convenience of the test suite developers, and in the interim waivers should be granted for this problem. Note: "removed" means preferably not run in the XPG3 mode, although a warning would be acceptable,again this is left to the discretion of the test suite developers.

Test suite status: (ii)

[The VSX4 test is ANSI.os/streamio/fseek.12]

4.33 Interpretation Request ref: VWG/033/191192

Publication:	XSH4
Interface:	i_noncanon
Date requested:	19.11.92
Status:	Closed

[***Request text starts***]

XSH4/VSX4 Reviewers Comments:

We would request an interpretation against the following test scenario in the X/Open test suite on the basis that the behaviour it requires is not useful for an application and places artificial restrictions on the implementation of the tty subsystem.

The test (POSIX.os/devclass/i_noncanon.14) is based on the following POSIX 1003.3.1 assertion.

When ICANON is clear, MIN > 0, TIME > 0, and data is in the buffer at the time of the call to read(), then the timer is started as if data had been received immediately after the call to read(). Posix Ref: Component Terminal Interface Assertion 7.1.1.7.1-35(C)

This assertion is derived from the sentence "*if data is in the buffer at the time of the read(), the result will be as if the data has been received immediately after the read()*" (from POSIX.1-90 7.1.1.7).

This is tested by writing *MIN-1* bytes, letting the timer expire and then writing another byte before doing a read from the other side of the tty loopback. In our implementation (which is *STREAMS* based) there are multiple processing modules and data is buffered in two places (potentially), the line discipline and the stream head. In this test scenario the *MIN-1* bytes are forwarded to the stream head when the first timer expired. When the subsequent byte arrives the timer is reset in the line discipline module. When the *read()* occurs only the data previously forwarded to the *STREAM* head is returned , and the other byte remains in the *STREAM* until the timer expires.

We would request an interpretation against this test as we feel the behaviour required is of no benefit for an application since the setting of *MIN > 0* and *TIME > 0* expects the number of bytes returned to be the smaller of *MIN* and the number of bytes actually *read()*. Thus an application in this situation is not relying on receiving exactly *MIN* bytes (if it did require this it could set *MIN > 0* and *TIME =0*), and thus this implementation behaves as an application might expect for the *MIN >0* and *TIME>0* scenario.

Since no application would count on the behaviour expected by the test, we don't think the test is useful and would request an interpretation to permit *STREAMS* based tty implementations. To specify expected behaviour when not useful is too restrictive on implementations.

VSX Developers Response:

From reading the text of the comments above, it seems that there may be a difference between the test and the implementation as to when the inter-byte timer comes into play and what affects this has on the buffering of data. Having looked at the specification in XPG4 and in POSIX.1-1990, I can see how this difference arises.

The implementation seems to be using the inter-byte timer irrespective of whether or not there is a pending read() on the serial line. The result of this is that bytes received with an inter-byte delay of less than *TIME* are considered to constitute a separate unit. Once there is a delay between bytes in excess of *TIME* a new unit is started.

The test assumes that the inter-byte timer is only activated by a *read()* and that bytes received prior to the *read()* are to be grouped together up to the size *MIN* and delivered to satisfy the *read()*.

The assumption that the test makes is based on the wording of the assertion which states that "*the timer is started as if data had been received immediately after the read()*". This is not explicitly stated in POSIX.1, though it could be reasoned that this is the case from the context and from the way that other *MIN*, *TIME* interactions behave.

The implementation assumption is that the inter-byte timer is activated when the first byte is received after the *MIN*, *TIME* settings are made and that when "*the timer expires before MIN bytes are received, the characters received to that point are returned to the user*". Thus, the implementation groups the characters in units ready for return to the user, even though there is no pending *read()* to be satisfied.

There are a number of pros and cons to each of these interpretations and both of them could be considered valid. The implementation's interpretation is possibly not enforceable as a sole interpretation since it relies on a scheme of two level buffering which all systems may not be able to support. However, this does not mean that it is an invalid interpretation.

As regards the use of these settings for *MIN* and *TIME* by applications, the major use would relate to record-based terminal I/O with a time-out mechanism in the case that a partial record is received. It is doubtful whether an application would use this *MIN*, *TIME* interaction without providing a mechanism to deal with the receipt of partial records. Thus, it can be surmised that this difference in implementation should not affect a well written application.

[***text ends***]

X/OPEN RESOLUTION:

A permanent waiver should be granted for this behaviour.

The tests will be left as they are and this issue dealt with through the formal waiver process.

Test suite status: (iv)

[The VSX4 test is POSIX.os/devclass/i_noncanon.14]

4.34 Interpretation Request ref: VWG/034/111292

Publication:	XSH4
Interface:	strtol and strtoul
Date requested:	11.12.92
Status:	Closed

[***Request text starts***]

XSH4/VSX4 Reviewers Comments:

There appear to be differences between *strtol()* and *strtoul()* on the handling of errors. Is this intentional or an oversight?

CAE Specification, System Interfaces and Headers, Issue 4, July 1992. *strtol()* on page 438 says "[EINVAL] The value of base is not supported." is an optional error, but *strtoul()* on page 440 says it is a required error.

strtoul() on page 440 says "[EINVAL] No conversion could be performed." is an optional error, but *strtol()* makes no mention of that error.

In addition, neither function says what should happen when base has the value of 1(one). I assume that they both parse the leading white-space characters, the optional sign, and the one or more 0(zero) characters, and return the value 0(zero) and update *endptr to point to the first character of the final string.

[***text ends***]

X/OPEN RESOLUTION:

For XSH4 "*EINVAL - the value of the base is not supported*" should remain optional for *strtol()*, and mandatory for *strtoul()*. In a future revision of the XSH the working group should consider making these interfaces consistent with the recommendation that they both become mandatory errors.

The optional error "*EINVAL - no conversion could be performed*" is mentioned in the RETURN VALUE section for *strtol()* (and thus it is defined). A future revision will add this error to the "may fail" section to correct this.

The behaviour when base=1 is not specified by this issue of the XSH for either function. In the case of *strtol()* the function may fail with EINVAL, however this is not mandatory.

In a future revision of the XSH the working group should consider adding a failure indication to these interfaces when base is unsupported by the implementation.

Test suite status: (vii)

The March 1994 meeting of XoTGbase agreed to defer this issue to a future revision.

4.35 Interpretation Request ref: VWG/035/221292

Publication:	XSH4/XPG3
Interface:	alarm
Date requested:	12.22.92
Status:	Closed

[***Request text starts***]

XSH4 Reviewers Comments:

XPG3(and XPG4) read:

The alarm function causes the system to send the calling process a SIGALRM signal after the number of real time seconds specified by "seconds" has elapsed.

The point is: is it allowed that the signal arrives earlier than specified in *alarm()*? This is common practice in UNIX systems due to the implementation of the alarm clocks. The signal can arrive one second earlier. Thus alarm(1) may deliver the signal immediately. This can cause a lot of problems and so our question: is this allowed?

[***text ends***]

X/OPEN RESOLUTION:

POSIX.1 and therefore XPG3 and XPG4 do not allow the signal to arrive earlier than specified.

Test suite status: (vi)

4.36 Interpretation Request ref: VWG/036/022093

Publication:	XCU4
Interface:	more -t, vi -t, ex -t
Date requested:	02.20.93
Status:	Closed

[***Request text starts***]

The "more -t tagstring" option says that it will

Write the screenful of the file containing the tag named by the tagstring argument.

We presume the following, none of which are stated in XCU4:

i. The file "*tags*" in the current directory is used to locate tags for this purpose (and for the *:t* interactive command), the format of that file is expected to be that produced by the *ctags* utility, and if such a file is unreadable for any reason, or the tag cannot be found in the *tags* file, or tag text cannot be found in the file indicated by the *tags* file, the behavior is the similar to that of the *vi* utility.

ii. The term "containing the tag" actually means "containing the line associated with the tagstring in the *tags* file".

iii. The *-t* option and the *:t* interactive command not only "write the screenful containing the tag", but also remain within the *more(1)* utility for further commands (unless *-e* is used).

iv. Since there could be multiple possible "screenfuls" containing the indicated tag, the screenful written is selected as if the command "*/tagtext*" were used (that is, centered on the screen where possible unless the line is less than one screenful away from the current position). Basically, the behavior is similar to the behavior of *vi*.

Are these presumptions correct?

[***text ends***]

X/OPEN RESOLUTION:

The presumption in (i) above is too strong. The text stating '*The file "tags" in the current directory...*' needs to be made to match the ex/vi specification saying that a file named "*tags*" will be searched in the current directory and in other implementation-dependent directories.

With the modification to (i) above, these presumptions seem appropriate and this behaviour should be allowed on an XCU4 conforming system, however these are not additional requirements, since POSIX.2 and XCU4 do not clearly specify the results in these cases.

These changes will be submitted to POSIX.2b and any revision of XCU4.

Test suite status: (vii)

4.37 Interpretation Request ref: VWG/037/022093

Publication:	XCU4
Interface:	vi ˆQ
Date requested:	02.20.93
Status:	Closed

[***Request text starts***]

The vi utility described in XCU4 and POSIX.2 have a new command <control>-Q (which performs the same function as <control-V>, that is, escape the special meaning of the following character).

Many implementations of vi do not disable the tty START and STOP flow control (because they might be used over slow modems, for instance), and POSIX.1 (and XPG4) implementations have permission not to be able to change the characters used for the START and STOP tty functions.

Given an implementation where the START character is unchangeably ˆQ, does vi's (new) requirement for the ˆQ command to do the "literal-next" function have to disable the START/STOP function of the tty driver, thus potentially flooding dial-in terminals? That is, is such an implementation permitted to treat ˆQ as a START character rather than a "literal next" character?

[***text ends***]

X/OPEN RESOLUTION:

Given the application usage note in XCU4, the interpretation is clear: Implementations are not required to disable tty START and STOP flow control.

Test suite status: (i)

4.38 Interpretation Request ref: VWG/038/022093

Publication:	XCU4
Interface:	xargs
Date requested:	02.20.93
Status:	Closed

[***Request text starts***]

We have found a conflict between XPG4 and POSIX.2 in the xargs utility.

The problem is that if xargs supports the '-*e*' option, then you cannot be POSIX compliant since it is possible that conforming POSIX applications will fail when using xargs.

XPG4 says:

-e[eofstr]

Use eofstr as the logical end-of-file string. Underscore(_) is assumed for the logical EOF string if neither -e nor -E is used. When the -eofstr option-argument is omitted, the logical EOF string capability is disabled and underscores are taken literally. The xargs utility reads standard input until either end-of-file or the logical EOF string is encountered.

POSIX.2 does not support -e option.

The problem:

Systems that support the -e option (e.g XPG4) also must support the logical EOF string which is assumed to be the underscore character. POSIX.2 does not support this feature and thus, does not support the logical EOF. Thus, POSIX compliant applications will terminate early when the underscore character is given to XPG4's xargs standard input.

Is XPG4 supposed to be POSIX.2 conforming? If so, can you confirm that this is a POSIX conformance problem.

In either case, does anyone have ideas how a system can get their xargs utility to be both XPG4 and POSIX.2 conforming?

[***text ends***]

X/OPEN RESOLUTION:

An interpretation request should be made to POSIX.2 with the recommendation that historical behaviour be allowed. In the meanwhile a temporary interpretation will allow this but not require it for XCU4.

Test suite status: (vii)

4.39 Interpretation Request ref: VWG/039/022093

Publication: XSH4

Interface: namespace

Date requested: 02.20.93

Status: Closed

[***Request text starts***]

POSIX 1003.4 and 1003.4a are being defined as extensions to POSIX 1003.1 (WG15 plans to modify IS 9945-1). Our system wishes to support both these emerging standards and XPG4.

A problem that we have observed is that adding the reentrant functions defined by these standards has implications on the namespace; (reentrant versions of functions have _r added as a postfix to the name). According to the namespace rules outlined in XSH4 this is not allowed.

We would like to add these functions to the POSIX (and thus XPG4) namespace and believe that this is the intent of the realtime extension. We would recommend that this be an allowable extension to the XPG4 namespace.

[***text ends***]

X/OPEN RESOLUTION:

The existing XPG4 namespace rules should remain as they are.

The POSIX.4 and POSIX.4a symbols can be made available to an application using the namespace enablers described in POSIX.4.

Test suite status: (i)

4.40 Interpretation Request ref: VWG/040/031793

Publication:	XSH4
Interface:	fscanf
Date requested:	03.17.93
Status:	Closed

[***Request text starts***]

There appears to be a conflict in the description of the "C" conversion character in *fscanf()*. Is this intentional or an oversight?

The referenced document is "*CAE Specification, System Interfaces and Headers, Issue 4, July 1992*".

fscanf() on page 152 in the last paragraph states:

"*Matches a sequence of characters of the number specified by the field width (1 if no field width is present in the directive).*"

fscanf() on page 153 in the first paragraph states:

"*If the field width is specified, the behaviour is unspecified.*"

Which statement is correct?

[***text ends***]

X/OPEN RESOLUTION:

%C is the wide equivalent of %c. The width determines the number to match, with a default of 1.

Test suite status: (i)

4.41 Interpretation Request ref: VWG/041/031793

Publication:	XSH4
Interface:	chroot
Date requested:	03.17.93
Status:	Closed

[***Request text starts***]

Our system gets an *UNRESOLVED* test for the testset *XOPEN.os/procenv/chroot.4*, because *chroot()* and shared libraries don't mix very well. Other good reasons for not attempting to test *chroot()* is that it historically has deleterious effects upon the correct operation of other standard functions, such as *getpwuid(), tmpfile(),* and *ctermid(),* and this concept can be extended to all of XSH4, since we do not know whether their implementation depends on particular files.

In fact, the test suite itself depends on *chroot()* messing up the semantics of *confstr(_CS_PATH),* although this is an unwarranted assumption!

We granted permanent waivers for this reason in XPG3, and the situation is unchanged in XSH4.

We believe that XSH4 should therefore have stated that the effects of *chroot()* upon subsequent calls to system-provided functions are undefined, and/or *chroot()* should have been withdrawn entirely.

We recommend that tests for the correct execution of *chroot()* be removed, or at least any unusual events observed be classified as "FIP" or "Warning".

Finally, since XPG4 states that "There is no portable use that an application can make of this interface", we believe the test suite avoid using chroot() entirely, since the result is a non-portable test suite.

The subsequent behavior of the system after any call to chroot() other than chroot("/") or equivalent should be considered a Grey Area in XSH4.

[***text ends***]

X/OPEN RESOLUTION:

A permanent waiver is granted for this behaviour. It is recommended that at the next upgrade of VSX4 that the test suite be changed to associate the current failure result with a FIP result, so that this may be resolved by signing off the test report.

Test suite status: (ii)

[The VSX4 test is XOPEN.os/procenv/chroot.4]

4.42 Interpretation Request ref: VWG/042/031993

Publication:	XCU4
Interface:	hash
Date requested:	03.19.93
Status:	Closed

[***Request text starts***]

In the Commands and Utilities, Issue 4, the *hash* utility under the APPLICATION USAGE section (page 381) states:

"*The hash utility may be implemented as an alias, for example, alias -t -, in which case utilities ...*"

I think there are 2 problems with this wording:

i. Is it really supposed to say "*alias -t -*", or is it just supposed to say "*alias -t*" ?

ii. Looking at the *alias* utility (page 87), I find no reference to a "*-t*" option. The OPTIONS section for *alias* says "*None*".

I believe that this is refering to the korn shell implementation (ksh version 11/16/88) where *hash* is a preset alias to "*alias -t*". The korn shell supports both the -t and -x options to *alias*.

I believe that either XCU Issue 4 should support and document the -t option to *alias* or that the *hash* utility be better documented with an enhanced description of the semantics of "*alias -t*" and where "*alias -t*" comes from (eg ksh88).

[***text ends***]

X/OPEN RESOLUTION:

There is no issue to be resolved here. This information points out that XCU4 implies that certain options are available that haven't been specified, but it doesn't request any interpretation of how an XCU4 conforming implementation should behave.

The options mentioned in the request will be considered for inclusion in a future revision of XCU4.

Test suite status: (vii)

4.43 Interpretation Request ref: VWG/043/032593

Publication:	XSH4/XPG3
Interface:	GTI Support for Network Connections
Date requested:	03.25.93
Status:	Closed

[***Request text starts***]

Our system does not support directly connected asynchronous terminals. It does however support network connections.

The network connections (using pseudo-ttys) offer partial support of the POSIX GTI, for all practical purposes the support is enough. They do not support hardware related features of RS232 connected devices, such as parity generation / detection and character size.

These network connections have been used to support running of the test suite, but due to only partial support of the GTI a group of tests still fail.

We would request an interpretation to allow systems whose hardware does not support directly connected asynchronous terminals to waive tests related to failures in the pseudo-tty loopback implementation.

Since there is no requirement for an implementation to support the POSIX General Terminal Interface (GTI) there should be no requirement for a pseudo-tty implementation to support it 100%.

[***text ends***]

X/OPEN RESOLUTION:

There is no requirement in the XPG that pseudo-ttys support the POSIX General terminal interface, however in a future revision of the XPG this may be considered.

For X/Open testing, there are several options available:

i. If two terminal devices which support the GTI are available, use them as a hardware loopback.

ii. If pseudo-terminals are available which support the GTI well enough to produce sensible test results, use them as a software loopback.

iii. Configure VSX so that all terminal related tests give Unsupported results

For systems that do not support any GTI conforming terminals, such as the submitter of this interpretation they should choose option 3, and configure the loopback ports as unsupported.

Test suite status: (i)

4.44 Interpretation Request ref: VWG/044/050793

Publication:	XSH4
Interface:	strftime,strptime
Date requested:	05.07.93
Status:	Closed

[***Request text starts***]

For *strftime()* and *strptime()*, the *%Od* and *%Oe* behavior on "single digit" values when alternate digits are part of the current locale:

Does padding with a space or the alternate zero ever make sense?

For example when using alt_digits defined as "0th","1st","2nd" ... does "0th1st" make sense for zero filled "1st" using alternate digits?

[***text ends***]

X/OPEN RESOLUTION:

Alt_digits are defined as alternative symbols for digits. The example does make sense, but as the XBD states it is only for illustrative purposes.

The padding behaviour is specified for the %Oe and %Od cases, albeit not that explicitly, we would recommend that a future revision of the XPG better specify this area.

Test suite status: (vii)

4.45 Interpretation Request ref: VWG/045/050793

Publication:	XSH4
Interface:	strftime,strptime
Date requested:	05.07.93
Status:	Closed

[***Request text starts***]

For *strftime()* and *strptime()*, the restriction to [0,99] for %C:

Are all tm_year values to be reduced (somehow) to/from [0,9999]? If so, what is the reduction algorithm? If not, is the intent "merely" that at least two digits are produced, and that there is no restriction for strptime()?

What happens if you use a negative tm_year such as -5001-1900 (5001BC) should it return 50 or -50?

[***text ends***]

X/OPEN RESOLUTION:

The range specification of [0,99] in the *strptime()* definition for %C is erroneous and should be [00,99].

If *strftime()* and *strptime()* both restrict %C to [00,99], it leaves us in the non-negative range; nothing is mentioned about "negative" centuries so it should not be allowed. So for the example 5001BC (an implementation-defined value if ever there was one!) the century value is 50.

As for reducing tm_year values, the intent is and that *strptime()* should operate under the same bounds as *strftime()*.

Test suite status: (vii)

4.46 Interpretation Request ref: VWG/046/050793

Publication:	XSH4
Interface:	era handling
Date requested:	05.07.93
Status:	Closed

[***Request text starts***]

When the current locale has no era-specific LC_TIME information, for those strings that have obvious parallels in the standard strings (ERA_D_FMT and D_FMT, ERA_D_T_FMT and D_T_FMT, ERA_T_FMT and T_FMT), can (or should or must or must not) *nl_langinfo()*, *strftime()*, and/or *strptime()* use the standard strings instead or should they return the equivalent string in the POSIX locale which would be "" ?

We assume that they should return a standard string instead (e.g. D_FMT when ERA_D_FMT is undefined) - Is this correct?

[***text ends***]

X/OPEN RESOLUTION:

The assumption is correct. The behaviour is as specified in XPG4:

XPG4 System Interfaces, p429, "Modified Directives" states:

If the alternative format or specification does not exist in the current locale, the behavior will be as if the unmodified conversion specification were used.

Test suite status: (vii)

4.47 Interpretation Request ref: VWG/047/062993

Publication:	XSH4
Interface:	semctl
Date requested:	06.29.93
Status:	Closed

[***Request text starts***]

Inconsistency between XPG4 and XPG3 semctl()

The XPG4 definition of the semctl() function is incompatible with the XPG3 definition in a way that makes an application written to XPG3 not runnable on XPG4. The XPG4 specification needs to be corrected as it breaks ABI's based on XPG3.

On examination we find that the XSH4 semctl definition is not compatible with the following specifications:

1. XPG3
2. SVID Issue 2
3. SVID Issue 3
4. SPARC ABI
5. SCD 1.x & SCD 2.0

Specifications 1,2 & 3 all define the last parameter to semctl() to be a union. Not an array of shorts etc, but a union. These are all source level interfaces so code will fail to compile when targeted to an XSH4 system.

Specifications 4 & 5 are binary interfaces. The SPARC ABI explicitly requires that all unions are passed as pointers, therefore passing an int and/or a union will not result in the same runtime behaviour.

XSH4 defines that when the optional fourth arg is used, it may be one of:

int
array of unsigned short
struct semid_ds

XSH4 says that the application writer may also define the fourth arg as the sumun union.

Therefore given that XSH4 is incompatible with XPG3, SVID-3, SCD and the SPARC ABI, we therefore require that X/Open declares this interface to be in error and revert to the traditional definition of semctl.

[***text ends***]

X/OPEN RESOLUTION:

The manual page for semctl() should be revised. The revised description follows; note that the section of application usage after the sentence ending "backwards compatibility" has been removed.

This change will be included in a future revision to the XPG4.

[Revised description for semctl() interface]

SYNOPSIS

#include <sys/sem.h>
int semctl(int semid, int semnum, int cmd, ...);

DESCRIPTION

The semctl() function provides a variety of semaphore control operations as specified by cmd. The fourth argument is optional, depending upon the operation requested. If required it is of type union semun, which must be explicitly declared by the application program.

```
union semun {
        int val;
        struct semid_ds *buf;
        unsigned short *array;
} arg;
```

The following semaphore control operations as specified by cmd are executed with respect to the semaphore specified by semid and semnum. The level of permission required for each operation is shown with each command, see Section 2.5 on page 27. The symbolic names for the values of command are defined by the <sys/sem.h> header.

GETVAL Return the value of semval, see <sys/sem.h>. Requires read permission.

SETVAL Set the value of semval to arg.val, where arg is the fourth argument to semctl(). When this command is successfully executed, the semadj value corresponding to the specified semaphore in all processes is cleared. Requires alter permission, see section 2.5 on page 27.

GETPID Return the value of sempid. Requires read permission.

GETNCNT Return the value of semcnt. Requires read permission.

GETZCNT Return the value of semzcnt. Requires read permission.

The following values of cmd operate on each semval in the set of semaphores.

GETALL Return the value of semval for each semaphore in the semaphore set and place into the array pointed to by arg.array where arg is the fourth argument to semctl(). Requires read permission.

SETALL Set the value of semval for each semaphore in the semaphore set according to the array pointed to by arg.array, where arg is the fourth argument to semctl(). When this command is successfully executed, the semadj values corresponding to each specified semaphore in all processes are cleared. Requires alter permission.

The following values of cmd are also available:

IPC_STAT Place the current value of each member of the semid_ds data structure associated with semid into the structure pointed to by arg.buf, where arg is the fourth argument to semctl(). The contents of this structure are defined in <sem/sys.h>. Requires read permission.

IPC_SET Set the values of the following members of the semid_ds data structure associated with semid to the corresponding value found in the structure pointed to by arg.buf, where arg is the fourth argument to semctl():

sem_perm.uid
sem_perm.gid
sem_perm.mode

The mode bits specified in section 2.5.1 on page 27 are copied into the corresponding bits of the sem_perm.mode associated with semid. The stored value of any other bits are unspecified. This command can only be executed by a process that has an effective user ID equal to either that of a process with appropriate privileges or to the value of sem_perm.cuid or sem_perm.uid in the semid_ds data structure associated with semid.

IPC_RMID Remove the semaphore identifier specified by semid from the system and destroy the set of semaphores and semid_ds data structure associated with it. This cmd can only be executed by a process that has an effective user ID equal to either t hat of a process with appropriate privileges or to the value of sem_perm.cuid or sem_perm.uid in the semid_ds data structure associated with semid.

APPLICATION USAGE

The fourth parameter in the SYNOPSIS is now specified as ... in order to avoid a clash with the ISO C standard when referring to the union semun (as defined in XPG3) and for backward compatibility.

[end]

The following points should also be noted as additional background rationale for this interpretation.

i. The union type *semun* is not defined in any header file in XPG3/XPG4 or SVID3 for that matter.

ii. The namespace requirements for XPG3 are not strict and hence the union type *semun* could be added to a header if desired (although not recommended).

The March 1994 meeting of XoTGbase agreed to address this change into the upcoming revision of XPG4 for SPEC1170.

4.48 Interpretation Request ref: VWG/048/061193

Publication:	XCU4
Interface:	ps
Date requested:	06.11.93
Status:	Closed

[***Request text starts***]

SVID Issue 3 and XPG 4 disagree about the meaning of the ’-g’ flag. SVID says "process groups", and provides ’-s’ to search for session groups. XPG4 says ’-g’ means session groups, and doesn’t provide any way to search for process groups.

POSIX.2, which has a more general-purpose formatting interface to ps, does not address this.

Resolve this inconsistency, probably either in favour of SVID3 or by ripping out these options and using the POSIX.2 approach.

[***text ends***]

X/OPEN RESOLUTION:

For XCU4 implementations are allowed to have -g mean either *session leaders* or *process group leaders*.

The working group recommends that this be revisited in a future revision of XCU4, possibly by adding a -j option or somehow making the definitions of *session leader* and *process group leaders* more clear in respect to the different implementations (SVID and BSD).

Test suite status: (vii)

4.49 Interpretation Request ref: VWG/049/061193

Publication:	XCU4
Interface:	od
Date requested:	06.11.93
Status:	Closed

[***Request text starts***]

In X/Open CAE Specification (1992), Commands and Utilities, Issue 4, page 534, od, option -c, it states:

> -*c* *Interpret bytes as characters in the ISO/IEC 646:1991 character set. ... This is equivalent to -t c in the POSIX locale.*

This wording appears to come from the older Issue 3 od specification, where the word ASCII was used. XPG3 supported only the concept of an ASCII system; while XPG4 is attempting to support any codesets. The re-wording of ASCII to ISO/IEC 646 has resulted in a contradiction between the two sentences quoted above: there is no requirement that the POSIX locale be 646 based; indeed an EBCDIC based system is explicitly valid.

We would suggest that the correct interpretation be that -*c* differs only from -*t c*, in that -*t c* issues a *setlocale(LC_CTYPE, "");* to the current locale, while -*c* issues a *setlocale(LC_CTYPE, "POSIX");*.

The wording of the first sentence quoted should probably be: Interpret bytes as characters in the coded character set associated with the POSIX locale.

[***text ends***]

X/OPEN RESOLUTION:

The wording should allow bytes to be interpreted as characters according to the code set of the current locale:

> -*c* *Interpret bytes as characters specified by the current setting of the LC_CTYPE locale category. This is equivalent to -t c.*

Test suite status: (vii)

The March 1994 meeting of XoTGbase agreed to address this change into the upcoming revision of XPG4 for SPEC1170.

4.50 Interpretation Request ref: VWG/050/072293

Publication:	XCU4
Interface:	getconf
Date requested:	06.28.93
Status:	Closed

[***Request text starts***]

We have identified the following set of problems with the XPG4 getconf utility.

i. There seems to be a discrepency between XPG4 and POSIX.2 in the documentation of the getconf utility.

   POSIX.2 states, in Draft12 , Section 4.26.4, page 305, lines 4586-4593, that

   - *Name column of POSIX.1(8) Table 2-3 (Minimum Values) ...*

   - *Variable column of POSIX.1 (8) Table 4-2 ... without the enclosing braces ...*

   shall be recognized as valid 'system_var' operands

   and lines 4596-4601

   > *The configuration variables listed in the Variable column of the POSIX.1(8) Table 5-2 ... without the enclosing braces shall be recognized as valid 'path_var' operands.*

   To me this means that the getconf should recognize, at least, the following:

   _POSIX_ARG_MAX
   _POSIX_CHILD_MAX
   _POSIX_LINK_MAX
   _POSIX_MAX_CANON
   _POSIX_MAX_INPUT
   _POSIX_NAME_MAX
   _POSIX_NGROUPS_MAX
   _POSIX_OPEN_MAX
   _POSIX_PATH_MAX
   _POSIX_PIPE_BUF
   _POSIX_SSIZE_MAX
   _POSIX_STREAM_MAX
   _POSIX_TZNAME_MAX
   _POSIX_JOB_CONTROL

_POSIX_SAVED_IDS
_POSIX_VERSION
_POSIX_CHOWN_RESTRICTED
_POSIX_NO_TRUNC
_POSIX_VDISABLE

However, in the XPG4 documentation "Commands and Utilities Issue 4", these variables are listed WITHOUT the leading '_' character. I hope this is just a mis-print in the XPG4 docs.

[***text ends***]

X/OPEN RESOLUTION:

It is recognised that this is a mis-print and that the variables should include the leading '_' character.

ii. [***Request text starts***]

This problem is somewhat related to Problem 1. There are several symbols listed as 'system_var's in the getconf utility documentation:

XOPEN_VERSION
XOPEN_XCU_VERSION
XOPEN_XPG2
XOPEN_XPG3
XOPEN_XPG4

These do not have the leading '_' character BUT the symbolic constant names that are defined in the <unistd.h> documentation appear with the leading '_' character. Should not these symbols have the leading '_' character in the getconf documentation?

[***text ends***]

X/OPEN RESOLUTION:

It is recognised that this is a mis-print and that the variables should include the leading '_' character.

iii. [***Request text starts***]

XPG4 getconf documentation lists a MN_NMAX 'system_var' variable that I could not find a reference or definition to in the "System Interfaces and Headers" documentation. Can anyone tell me what this is for, and where I should expect it to be defined?

[***text ends***]

X/OPEN RESOLUTION:

NM_MAX is a misprint of NL_MAX.

[***Request text starts***]

iv. There seems to be a discrepancy between the names of several configuration variables. This discrepancy is between the getconf utility documentation and the <limits.h> documentation in "System Interfaces and Headers".

In getconf, there is NL_SET_MAX and NL_TEXT_MAX but in <limits.h> there is NL_SETMAX and NL_TEXTMAX Which is correct? With or without the 2nd '_' character?

[***text ends***]

X/OPEN RESOLUTION:

The variables in <limits.h> are correct. This will be corrected in the next update to XCU4.

v. [***Request text starts***]

The definition of XPG4's sysconf() defines variables

_XOPEN_CRYPT
_XOPEN_ENH_I18N
_XOPEN_SHM

Should not these also be listed in the getconf utility documentation as a valid 'system_var' ?

[***text ends***]

X/OPEN RESOLUTION:

These should be listed.

Test suite status: (vii)

The March 1994 meeting of XoTGbase agreed to address this change into the upcoming revision of XPG4 for SPEC1170.

4.51 Interpretation Request ref: VWG/051/082093

Publication:	XSH4
Interface:	sccs
Date requested:	08.20.93
Status:	Closed

[***Request text starts***]

XPG4 CAE Specification "Commands and utilities, Issue 4" page 487 describes the "make command" special target .SCCS_GET.

.SCCS_GET: sccs $(SCCSFLAGS) get $(SCCSGETFLAGS) $@ -G$@

Existing implementations of the "sccs get" command do not support a "-G" option. Is this a typographical error? If it is not, can an explanation be provided as to what this option means?

[***text ends***]

X/OPEN RESOLUTION:

It is agreed that -G is a typographical error.

Test suite status: (vii)

The March 1994 meeting of XoTGbase agreed to address this change into the upcoming revision of XPG4 for SPEC1170.

4.52 Interpretation Request ref: VWG/052/082193

Publication:	XSH4
Interface:	character order vs. collation sequence
Date requested:	08.21.93
Status:	Closed

[***Request text starts***]

Based on various current implementations studied, it has been revealed that the use of the terms "character order", and "collation sequence" is inconsistent and unclear, in both XBD4 and POSIX.2.

Does the term "collating element order" refer to:

i. the order in which definitions of collation elements are listed after the order_start keyword in the locale definition file,

   OR

ii. the order of those collation elements after they have been sorted according to the primary, secondary, and any other weights?

This distinction is important in that the term "collating element order" is used to specify how to compute the list of characters matched by range expressions (in file name patterns and regular expressions).

For instance, suppose the following appears in the locale definition file:

```
order_start
<a>                    <a>;<a>
<b>                    <b>;<b>
<zot>                  <a>;<b>
<d>                    <d>;<d>
order_end
```

Note that the collation element <zot> has the same primary weight (and thus is in the same equivalence class) as <a>, but it is listed between <b> and <d> in the list.

Thus, if the weights are considered, the "collation sequence" is:

<a> <zot> <b> <d>

but when only the order of appearance is considered, the "character order" is:

<a> <b> <zot> <d>

So, does the regular expression "[b-d]" match a string consisting only of the character <zot>?

Relevant references:

Neither XCU4 nor POSIX.2 has a glossary definition of "collation sequence", "collating element order", or "character order", but on XBD4, page 55, near the top, appears the following paragraph:

The character (and collating element) order is defined by the order in which characters and elements are specified between the order_start and order_end keywords. This character order is used in range expressions in regular expressions (see Chapter 7). Weights assigned to the characters and elements define the collation sequence; in the absence of weights, the character order is also the collation sequence.

This seems to strongly imply that only the "character order" is relevant for range expressions, and that this term is to be distinguished from the "collation sequence" by the fact that weights apply only for the "collation sequence". From this, we could conclude that the regular expression "[b-d]" would indeed match the "<zot>" character.

However, when we follow the advice of this paragraph and "see Chapter 7", we find the following paragraph on page 98:

> *7. A range expression represents the set of collating elements that fall between two elements in the current **collation sequence**, inclusively. It is expressed as the starting point and the ending point separated by a hyphen (-).*

Note that here, the term "collation sequence" is used, implying the applicability of weights, contradicting the implication of page 55. But, later in the same section (in the fourth paragraph of part 7 on this page) it also states that

> *The order used is the order in which the collating elements are specified in the current collation definition. One-to-many mappings ... will not be performed.*

This implies that the mere ordering of the definitions (i.e., the "character order") is used, and not the "collating sequence".

So, is the order to be used for understanding range expressions the "character order", or the "collation sequence"?

X/OPEN RESOLUTION:

There is a pending POSIX 1003.2 interpretation on this issue. The IEEE PASC Reference for this interpretation is : 1003.2-92 #27.

The resolution is to await the POSIX interpretation response, with the interim recommendation that "character order" is relevant for range expressions.

Test suite status: (vii)

The March 1994 meeting of XoTGbase agreed to address this change into the upcoming revision of XPG4 for SPEC1170.

4.53 Interpretation Request ref: VWG/054/092993

Publication:	XSH4
Interface:	getlogin()
Date requested:	09.29.93
Status:	Closed

[***Request text starts***]

Is getlogin() supposed to find the login name associated with the controlling terminal of the calling process, or the login name associated with file descriptors 0, 1, or 2? Or does it matter?

The APPLICATION USAGE refers to the "name associated with the current login activity", and use of the controlling terminal might better serve this than simply looking up whatever of the currently open low-numbered file descriptors that happen to be terminals.

The ERRORS section has the X/Open extension error ENXIO for when the calling process has no controlling terminal, indicating that this might be important, but since it's a "may", that might only be an implementation hint. The rest of the page says nothing about terminals at all.

[***text ends***]

X/OPEN RESOLUTION:

The XPG is intentionally silent on this matter, and thus "it does not matter".

Test suite status: (vii)

4.54 Interpretation Request ref: VWG/055/100693

Publication:	XSH4
Interface:	nl_langinfo
Date requested:	10.06.93
Status:	Closed

[***Request text starts***]

XPG4 says that for the "C" ("POSIX") locale, nl_langinfo(YESEXPR) will return the string "ˆ[yY]". (A similar argument is obvious for NOEXPR, so I'll just discuss YESEXPR.) It's pretty explicit about this, currently.

While implementing an RE engine it has come to my attention that some constructs, in particular, a [abc...] construct now is more expensive in time and space than a short (a|b|c|...) list within the ERE. For example, a (a|b|c|...) builds a set of nodes that is based on exact matches of certain fixed set of characters.

Given the above, however, I'd be happier if we were allowed to match the intent, but not necessarily return a string that exactly matches the above.

One can argue that there is NO difference in behavior between "ˆ[yY]" and "ˆ[Yy]", yet I take XPG4 as saying that the second is invalid. Similarly, the ERE "ˆ([yY])" is also identical in behavior, but again does not match the specified string.

Given a ruling that the intent is important (not the exact sequence of bytes that provides it), then we would be free to use "ˆ(y|Y)" which produces a faster and smaller ERE engine.

[***text ends***]

X/OPEN RESOLUTION:

We should allow the standard to support "regular expressions equivalent to ˆ[Yy]".

An observation on the testable aspects: We'd recommend a FIP result when the regular expression the test receives from nl_langinfo(YESEXPR) is different from the required expression, and require the person to sign off that it is equivalent to the required expression.

Test suite status: (vii)

4.55 Interpretation Request ref: VWG/056/101993

Publication:	XCU4
Interface:	what
Date requested:	10.19.93
Status:	Closed

[***Request text starts***]

The "what" command, on page 809 of the XCU, gives an example for using a "what string" in a C source code file. The example uses

```
char ident[] = "@(#)string";
```

and claims that what will give output for

```
foo.c
foo.o
a.out
```

The problem begins when an aggressive optimizing compiler is used; if the variable "ident" is never used or referenced in any way by the code, it gets removed by the compiler and what cannot then find the string in a.out.

1. Do the examples have the power of Normative Text?
2. If so, this example goes beyond the fair scope of the ’what’ command and should be changed.

[***text ends***]

X/OPEN RESOLUTION:

1. No, as with many examples in the XCU, it is incomplete since it does not state how the programme should be compiled

2. A future revision is recommended to cleanup the example to more clearly represent the definition and not be specific to individual implementations or to add a note warning about possible effects of an optimized compiler.

Test suite status: (vii)

4.56 Interpretation Request ref: VWG/057/102193

Publication:	XCU4
Interface:	compress
Date requested:	10.21.93
Status:	Closed

[***Request text starts***]

1. The description of the command says that the "ownership modes" of the original file should be kept if the file is replaced by one with the extension .Z.

   I believe that this is a typo (missing comma) and that the "ownership" and the "modes" of the original file should be kept.

2. The description of the command says that the "change times" of the original file should be kept if the file is replaced by one with the extension .Z.

   I believe that the "access times" of the original file is meant because the "change times" of a file cannot be set. Thus the "access times" of the original file should be kept.

3. The STDIN sections of the commands say that stdin will be used if a file operand is **-**. The descriptions of the commands do not define where the processed input should be written to in this case. I suppose that the output should be written to stdout, but the STDOUT sections do not mention this case.

[***text ends***]

X/OPEN RESOLUTION:

1. The "ownership" and the "modes" of the original file should be kept.
2. The "access times" of the original file should be kept.
3. In the case mentioned, the output should be written to stdout for the compress command.

Test suite status: (vii)

4.57 Interpretation Request ref: VWG/059/110293

Publication:	XSH4
Interface:	fgets()
Date requested:	11.02.93
Status:	Closed

[***Request text starts***]

During the VSX4 testing of our system we have been coming across a number of UNRESOLVED results for 239 tests. We have reason to believe that this is due to the way these tests are written. The error message that we get is of the form:-

deletion reason: fgets() failed to read data from pipe - errno
11(EAGAIN)

An example of where we suffer this result is in test:-

/tset/ANSI.os/streamio/putc/T.fputc 2

The problem is that this tests (and others I have looked at which suffer this problem) all use fgets() over a pipe for the purposes of the various processes involved in the test to communicate. Now fgets() requires a newline for it to be able to read data and the tests that are failing are where a newline has not been written down the pipe after the data. The tests all open the pipe O_NONBLOCK which mean that when a fgets() is issued and there is no data to read (because of the missing newline) it will fail with EAGAIN.

There are two other conditions that allow fgets() to return. The first is the number of bytes requested are read. This is not the case for this and the other tests. The second is that fgets() encounters a end-of-file condition. This cannot be guaranteed to be the case and so should not be relied on in this and other tests.

[***Test Suite Developer's Comments***]

We have considered your fgets() problem and we have the following points to make:

1. Although the XPG appears to require that fgets() should give an EAGAIN error in the situation in question, we believe that this is accidental and not intentional. It is a consequence of the way the POSIX.1 standard was incorporated into XPG3, which has resulted in the XPG making additional requirements beyond POSIX.1 rather than just adopting POSIX.1 as was intended. The errors sections for the stream I/O functions were derived from those of the relevant "underlying functions" in such a way that it appears that the functions are required to give those errors under the conditions described, whereas in fact POSIX.1 only requires that *if* the functions give an error under the conditions described, then the value of errno must be the same as for the underlying functions.

   Furthermore the behaviour of streams with non-blocking file descriptors is unspecified in POSIX.1 because POSIX.1 only requires that its stream I/O functions meet the requirements of the C Standard, and the C Standard does not include the concept of non-blocking streams.

2. All systems that have so far been branded XPG compliant behave as the test suite expects, including older systems of your own. By changing to the behaviour that the XPG appears to require you may cause many other applications besides VSX to break.

3. If this issue had come to light during development of XPG3 it is likely that it would have been changed to require the traditional behaviour.

   If you have changed your system because you believe the new behaviour is required by the XPG, then in our opinion the change was unnecessary. The simplest and fastest solution to your problem would be to change back to the traditional behaviour.

[***text ends***]

X/OPEN RESOLUTION:

Adopt the test suite developers position.

A future revision of the XPG is recommended to change the EAGAIN error case for fgets() to reflect that an error should only occur if the the functions give an error under the conditions described.

Test suite status: (i)

4.58 Interpretation Request ref: VWG/060/110293

Publication:	XSH4
Interface:	fprintf
Date requested:	11.02.93
Status:	Closed

[***Request text starts***]

There appears to be a conflict in the writeup of "C" conversion character in fprintf().

fprintf() on page 134 in the first paragraph (talking about precision): "The precision is ignored in C conversion."

fprintf() on page 136 in the paragraph about 'C' conversion character: "If the precision is specified, its effect is undefined."

Which is correct?

Proposed resolution/action:

Remove both sentences. Then, that would leave "If a precision appears with any other conversion character, the behaviour is undefined." on page 134 first paragraph to cover this case.

Relevant references:

The proposed resolution keeps "C" consistent with "c".

[***text ends***]

X/OPEN RESOLUTION:

Remove both sentences. Then, that would leave "If a precision appears with any other conversion character, the behaviour is undefined." on page 134 first paragraph to cover this case.

Test suite status: Fixed in VSX4.3.5.

4.59 Interpretation Request ref: VWG/061/110293

Publication: XSH4

Interface: fprintf

Date requested: 11.02.93

Status: Closed

[***Request text starts***]

fprintf() on page 136 in the paragraph about ’C’ conversion character is in conflict with ISO C with respect to integral promotion rules and argument passing for functions with variable argument lists.

XSH4 says "The wchar_t argument" while C says that fprintf is passed a wchar_t promoted to an int argument.

Proposed resolution/action:

"The int argument is converted to a wchar_t which is then converted to an array of bytes representing a character, and the resulting bytes are written."

Relevant references:

The user calls fprintf with a wide character. By the integral promotion rules, the C language promotes that wchar_t to an int. fprintf sees the int. fprintf should then demote the int back to a wchar_t. This assumes that sizeof(wchar_t) <= sizeof(int).

Also, see the description on ’c’ on page 136 in XSH4.

[***text ends***]

X/OPEN RESOLUTION:

XSH4 is conceptually accurate, but could possibly be improved by changing "The wchar_t argument..." to "The (possibly promoted) wchar_t argument is converted to a wchar_t, then..."

Test suite status: (vii)

4.60 Interpretation Request ref: VWG/064/011994

Publication:	XCU4
Interface:	sact
Date requested:	01.19.94
Status:	Closed

[***Request text starts***]

The question has to do with the behavior of sact when it is asked to report on an s-file that is not currently checked out for editing. This behavior does not seem to be specified in XPG4. Existing implementations seem to write a message to standard error but exit with status zero. Actually, they do a bit more: they write the file name to standard output, but a message like "There are no outstanding deltas for <filename>" is sent to standard error.

Does this behavior conform?

[***text ends***]

X/OPEN RESOLUTION:

Printing informative messages to stderr is historic practice which was overlooked when the X/Open utility descriptions were aligned to POSIX.2. The next revision of XCU will have the following change:

Change the STDERR section to read "Used only for optional informative messages concerning SCCS files with no impending deltas, and for diagnostic messages."

Test suite status: (vii)

4.61 Interpretation Request ref: VWG/067/033094

Publication:	XSH4
Interface:	Enhanced I18N Feature group
Date requested:	03.30.94
Status:	Closed

[***Request text starts***]

This is a question regarding the enhanced I18N feature group, VWG/025 and partial support of the interfaces.

Is a system conforming that provides XPG4 multibyte interfaces, and partial implementation of the _XOPEN_ENH_I18N feature group, where the partial implementation is the fact that the functions wcscoll and wcsxfrm are supported in the C locale only, but not in other locales.

The system has _XOPEN_ENH_I18N to (-1L) in <unistd.h>.

At the moment such a system would fail the VSX4 tests because they expect the functions supported in either none or all locales (i.e. C locale plus vsx test locales).

We see two possible interpretations:

- Interpretation 1:

  One possible interpretation is that the XoTGbase interpretation VWG/025 allows this level of partial support, and that this should be allowed for the Enhanced I18N functions to allow divergence of implementations for those implementations not meeting full XPG4.

- Interpretation 2:

  The alternate interpretation is that functions in feature groups must be either not supported or fully supported, except in the specific case of encrypt() when decryption is requested.

  The text of the interpretation states, "If a function is supported (even though all the Feature Group may not be supported) it shall behave consistently with the relevant description in the XSH4."

  The implementation in question will require a check to be made on each change of locale to determine whether the function is supported for that locale. It is not expected that applications written to the XSH4 definitions would incorporate such a check, or may only incorporate a check in the C locale and then assume that the function is available in all locales. Also, if partial support of functions were allowed, there would have been no need to include an explicit statement about partial support of encrypt().

[***text ends***]

X/OPEN RESOLUTION:

Adopt interpretation #1 for this specific question.

Test suite status: (i) Fixed in VSX4.3.5.

4.62 Interpretation Request ref: VWG/070/052594

Publication:	XBD
Interface:	D_T_FMT
Date requested:	05.25.94
Status:	Closed

[***Request text starts***]

X/Open CAE Specification (1992) (XBD) page 72 defines the POSIX locale format for D_T_FMT to be:

%a %b %e %H:%M:%S %Y

which differs to the C locale definition as defined for XPG3 which is:

%a %b %d %H:%M:%S %Y

This has only recently been brought to light, VSX4 currently expects the latter definition which strongly implies that many implementations have retained the XPG3 definition.

We'd request consideration that either be allowable for XPG4. Only for days 1-9 of any month will there be any difference between %d and %e. Moreover, the reverse functions (strptime and getdate) will not balk at an extra space or a single leading zero.

[***text ends***]

X/OPEN RESOLUTION:

XPG4 is very clear and conforming implementations should implement D_T_FMT to be "%a %b %e %H:%M:%S %Y".

Since the change has minimal impact on applications, for those implementations who don't yet support POSIX.2 conformance it is acceptable to return the XPG3 D_T_FMT string. We'd recommend the test be changed to issue a warning in this case.

Test suite status: (i) Fixed in VSX4.3.5.

4.63 Interpretation Request ref: VWG/071/061994

Publication:	XSH4
Interface:	iconv()
Date requested:	06.19.94
Status:	Closed

[***Request text starts***] Problem:

The second argument of iconv() is "const char **inbuf" on SYNOPSIS of iconv() page and "char **inbuf" on iconv.h page. Since applications should be able to pass an argument of type char ** to iconv(), iconv() page should be corrected. APPLICATION USAGE already says on line 9358 that the type of inbuf is char **.

Proposed resolution: The synopsis on the iconv() manual page is in error and the synopsis on the <iconv.h> page is correct. Change "const char **inbuf" to "char **inbuf".

[***text ends***]

X/OPEN RESOLUTION:

The synopsis on the iconv() manual page is in error and the synopsis on the <iconv.h> page is correct.

A future revision of XSH4 shall revise the prototype for iconv() as follows:

Replace the argument type on the iconv() arguments in the synopsis with that specified on the function prototype in the <iconv.h> header.

For conformance to the current version of XSH4 either prototype is allowed.

Test suite status: (iv)