IEEE Standards Interpretation for IEEE Std 1003.1™-2001 IEEE Standard Standard for Information Technology -- Portable Operating System Interface (POSIX®)

Interpretation Request #107
Topic: test XSI requirements **Relevant Sections:** XCU test

The XSI requirements for test(1) are ambiguous. Lines XCU 35440-35446 list precedence rules in the Rationale, but this section is non-normative. The normative requirement on line 35303 that ‘combinations of primaries and operators shall be evaluated using the precedence and associativity rules described previously’ is lacking several of these precedence rules, since they are not mentioned previously in the normative Operands section.

Example 1: “test \(= \)”. $2, ‘=’ is a binary primary, yet $1 is ‘(‘ and $3 is ‘)’, so both lines 35295 and 35297 apply. Since there is no normative rule that the string comparison binary ‘=’ has higher precedence than parenthesis surrounding a one-argument expression, an implementation could perform the binary test of $1 and $3 (false, since ‘(‘ and ‘)’ are not the same string) or the unary test of $2 (true, since ‘=’ is a non-empty string). However, all XSI implementations I am aware of choose the latter (in other words, give binary string comparison a higher precedence than () grouping of a single argument), since that is the behavior required in a non-XSI implementation. My proposal would require returning 1.

Example 2: “touch file; test ! -a file”. $2, ‘-a’ is a binary primary, yet $1 is ‘!’, so both lines 35295 and 35296 apply. Implementations are allowed to have, and many XSI implementations actually do have, ‘-a’ as a unary primary, which makes the two-argument test ‘-a file’ well-formed. Since there is no normative rule that the ‘!’ operator has higher precedence than the ‘-a’ logical binary operator, an implementation could perform the dual unary test of $1 and $3 (both true, so the overall -a test is true), or perform the negated unary test of $2 and $3 (per the Rationale, -a file should return true if file exists

and the implementation provides this extension, so the overall ! test is false). There are existing implementations that give binary -a higher precedence than ! on a 2-argument test, probably because line 35295 is listed first; bash strives for XSI conformance, but "bash -c 'touch file; test ! -a file'" returns true. There are also existing implementations that follow the precedence mentioned in the rationale; GNU coreutils and zsh both return false. My proposal would require performing the two-argument test, which has unspecified results, but the overall expression would return 1 in implementations with a unary -a that returns true on file existance.

Example 3: "touch file; test string -a \(-a file". Here, even the Rationale doesn't provide a precedence between -a and (). If -a has higher precedence than parenthesis, then there is a valid parse (left-associative binary -a of unary tests on $1 and $3, followed by unary test on $5; all three strings are non-zero, return true). But if parenthesis have a higher precedence than any other operator, there is a parse error (no matching ')'), and the return value must be greater than 1. All XSI implementations I am aware of treat this as a syntax error (in other words, precedence is similar to C where () is higher than &&). My proposal would require returning greater than 1.

Example 4: "test \(\) = \) \)". Here, even the Rationale doesn't provide a precedence between = and (). If = has higher precedence than parenthesis, then $2 and $4 are string arguments to $3, and $1 and $5 form a matched set of parentheses, resulting in true. If parenthesis have higher precedence, then $1 and $4 are treated as a pair (the grammar does not allow for $1 and $2 to be a pair, since an expression must appear in between), causing a syntax error with the trailing ')' in $5, as well as with the invalid 2-argument test ') ='. Here, behavior between implementations that strive for XSI conformance differ, as bash and zsh return 0, while GNU coreutils complains of a syntax error. My proposal would require returning 0.

Reword the paragraphs at line XCU 35296-7:

If $2 is a binary primary, ‹XSI shading›but not '-a' or '-o',‹/XSI shading› perform the binary test of $1 and $3. ‹XSI shading›If $1 is '(', $2 is not a binary primary, and $3 is ')', perform the unary test of $2.‹/XSI shading›

Add a sentence to the paragraph at line XCU 35274:

‹XSI shading›The ! operator has higher precedence than any unary primary.‹/XSI shading›

Reword the paragraph at line XCU 35275:

‹XSI shading›(expression) True if expression is true. False if expression is false. The parenthesis have lower precedence than string comparison binary primaries, but higher precedence than all other primaries, and can be used to alter the normal precedence and associativity. It is a syntax error if parenthesis not consumed by a binary string compari-

son operator are not balanced.‹/XSI shading›

Add sentences to the paragraph at line XCU 35303:

‹XSI shading›Unary primaries shall have a higher precedence than any other binary primary, and both unary and binary primaries have a higher precedence than the unary string primary.‹/XSI shading›

Interpretation Response
The standard is unclear on this issue, and no conformance distinction can be made between alternative implementations based on this. This is being referred to the sponsor.

Rationale for Interpretation
None.