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

Interpretation Request #38
Topic: number expressions **Relevant Sections:** XCU test/ shell arithmetic expansion

What constitutes a number in both of these cases, test(1) and shell arithmetic expansion, does not appear to be specified. This was brought to my attention when I noticed the differing implementations. Firstly, zsh, which will return 0 on the first command, and the arithmetic expansion will expand to 1. As opposed to FreeBSD's--and likely BSDs in general--/bin/sh, which will cause an error to be echoed to stderr and a non-zero result in both cases. At first I contacted the zsh developers, as I assumed this was specified behaviour after checking /bin/sh's and bash's test implementation. This was not the case, and Dan Nelson on the zsh work list responded with examples of other shells that behaved the same way as zsh. (A couple notable examples being GNU's test and pdksh). He also said that he believed this fell into undefined behaviour, and, therefore, zsh was not incorrect. I then contacted the FreeBSD-standards list to try to confirm that this was undefined behaviour. Those who responded believed that this was undefined, and thought that I should at least bring this question to the austin group mailing list.

Clarification of what constitutes a number in XCU's test(1) and shell arithmetic expansion, and perhaps other related locations. Specifically, a given number should be considered valid if strtol(str, &end, 0) accepts str as a valid number. This would allow the usage of hexadecimal and octal numbers in test and arithmetic expansion, which is likely a feature that would be welcomed by all. In the case of an invalid number, the utility should throw an error and return non-zero, perhaps a standard error code for an invalid numbers should be allocated, if one is not already. An invalid number in arithmetic expansion should cause an error to be thrown before the command is executed, and result in a non-zero value, and perhaps a different value than the former so that a distinction can be made between where the invalid number error occurred--before or during the execution of a command. An important note to make is the contrast between test(1) and

expr(1), where expr defines a valid integer, and test does not.

From http://www.opengroup.org/onlinepubs/000095399/utilities/expr.html: integer - An argument consisting only of an (optional) unary minus followed by digits. I also recommend that expr(1) be updated to allow hexadecimal and octal numbers in the format that strtol(str, &end, 0) accepts, so as to be consistent with the newly defined behaviour in test(1) and shell arithmetic expansion.

Interpretation Response
The standard does not speak to this issue of what constitutes a number in XCU's test(1) and shell arithmetic expansion, and as such no conformance distinction can be made between alternative implementations based on this. In the event that the primary operand to the primary operators (-gt, -ge, -lt, -le, -eq, -ne) are not integers, implementations are free to provide extensions that would recognize those values or to treat them as errors. Point 6 in the Utility Argument Syntax portion of the Utility Conventions (subclause 12.1 in the Base Definitions volume of the standard) states that unless otherwise specified, operands that are to be treated as numeric values are to be interpreted as decimal integers. Since the test utility doesn't specify different behavior, the n1 and n2 operands to the -eq, -ne, -gt, -ge, -lt, and -le binary primaries are required to be treated as decimal integers. The description of Arithmetic Expansion (subclause 2.6.4 in the Shell and Utilities volume of the standard), however, explicitly requires that decimal, octal, and hexadecimal constants have to be recognized when performing arithmetic expansions.

Rationale for Interpretation
None.