IEEE Standards Interpretations for IEEE Std 1003.2™-1992 IEEE Standard for Information Technology--Portable Operating System Interfaces (POSIX®)--Part 2: Shell and Utilities

Interpretations are issued to explain and clarify the intent of a standard and **do not** constitute an alteration to the original standard. In addition, interpretations are not intended to supply consulting information. Permission is hereby granted to download and print one copy of this document. Individuals seeking permission to reproduce and/or distribute this document in its entirety or portions of this document must contact the IEEE Standards Department for the appropriate license. Use of the information contained in this document is at your own risk.

IEEE Standards Department, Copyrights and Permissions, 445 Hoes Lane, Piscataway, New Jersey 08855-1331, USA

Interpretation Request #157
Topic: shell **Relevant Clauses:** 3.9.1

A) 3.9.1 (Simple Commands) has the text (page 135, lines 725 to 727): When a given simple command is required to be executed (i.e. when any conditional construct such as an AND-OR list or a case statement has not bypassed the simple command) ... Does this present a requirement on the time of execution of such a command in relation to tokenization and parsing, i.e. that it happens as soon as a simple command has been identified and it is known that it is the next simple command to be executed? Specifically, consider a script containing alias foo=bar; foo where foo is not previously defined as a function or alias. Which (if either) of the following behaviours are conforming? (a) The command foo is executed, the alias command not having been executed by the time the token foo was delimited; (b) The command bar is executed, the alias command having been executed by the time the token foo was delimited?

B) 3.2.3 (Double Quotes) (page 118, lines 55 to 59) says, concerning the $(...) construct The tokenizing rules in 3.3 shall be applied recursively to find the matching). Does the reference to the tokenizing rules in 3.3 include alias substitution (3.3.1)? If so, then: (a) Is the expansion of an alias encountered included in the text of the command to be substituted (in place of the original text), despite the provision to the contrary in 3.3(5) (page 120, line 125)? (b) May a) in the value of the alias terminate the $(...) construct? (c) Is alias expansion also applied when and if command substitution is applied? (d) Since alias substitution requires recognition of reserved words in correct grammatical context, its application here requires parsing of the command whose output shall be substituted to occur in the process of finding the matching). Is the behaviour in the event of a grammatical error defined (since recovery is necessary to continue to determine whether aliases should be substituted)? For example, may errors or diagnostics occur from the following? true || $(if; foo) If not, then: Is the reference to parsing of the command whose output is to be substituted in E.3.2 (page 823, line 2975) in error?

C) 3.3 (Token Recognition) allows the formation of a token that is delimited by the end of input. (a) This allows a token with an opening quote (single or double) but no closing quote. Must the end of input be considered to end the quoted text, or may an implementation consider this a syntax error. Specifically, must sh -c “echo ‘\$foo” output the four characters $foo (quote removal having been applied to the single ‘) without error, and may a conforming application use this construct? (b) If the end of input leaves a ${, $(, $((or ` construct unterminates, is the effect defined?

D) 3.2.3 (page 119, lines 71 to 74) provides that the constructs “`echo hello” and `echo “foo` produce undefined results. Are the results undefined if these constructs are encountered during token recognition, or only if expansions need to be performed on them? E) 3.3.1 (Alias substitution) does not clearly state what is to be done with the value of an alias that is substituted. Is it required that the resulting text itself be tokenized? If so, is the value of the alias treated as a separate input source, so that any final token is delimited by the end of the value of the alias; if not, what are the effects of the following definitions and uses of aliases?

```
alias foo=”’”
foo bar’
alias bar=”echo >”
bar>baz
```

Interpretation Response

A. The standard clearly states (page 120 line 149-150) that tokenization occurs before any grammatical rules are applied. All compound commands are tokenized in their entirety before they are executed. The grammar rules show that the whole line shall be read before tokenization. Therefore the command shall be “foo” and never “bar”. This also means that a function f() { alias foo=bar foo } will execute “foo” and not “bar”, since the entire function is tokenized before alias substition takes place.

B. The standard appears to mandate that alias substituion should occur, however, concerns have been raised about this which are being referred to the sponsor. Historically shells have not done this, and the standard would appear to be in error.

C. a. The standard does not speak to this issue of whether the single-quoted (or double-quoted, or back-quoted) string is correctly terminated by this end of token, and therefore it is acceptable for an implementation to treat this as an error. As such no conformance distinction can be made between alternative implementations based on this.
b. For ${, $(or $((the standard clearly states tha the effect is defined as a syntax error -- the tokenization completes, but the resulting grammar is in error. For ‘, the effect is as above (C subsection a). For ` , the effect should also be as above (the back-quoted string is a single token).

D. The standard clearly states that the constructs are undefined in all cases. E. The standard clearly states in 3.3.1 alias substitution along with 5.1, the definition for the alias utility, that the alias provides a string not a set of tokens that are interpreted/parsed when the alias substitution occurs. There is nothing in the standard that states that there is an end of token at the end of an alias.

Rationale for Interpretation
None.