IEEE P1003.2a Draft 8 – December 1991

2.14 Terminal Characteristics

⇒ **2.14 Terminal Characteristics.** *Add a new clause 2.14:*

This clause applies only to systems supporting the User Portability Utilities 7
Option. 7

The standard utilities historically have been implemented on a wide range of terminal types, but a conforming implementation need not support all features of all utilities on every conceivable terminal. This standard states which features are optional for certain classes of terminals in the individual utility description clauses. The implementation shall document which terminal types it supports and which of these features and utilities are not supported by each terminal. This implementation-defined list of terminals:

— Shall include at least one terminal type that is capable of supporting all of the standard utilities and all of their features, if the {POSIX 2_CHAR _TERM } option is provided.

— May group terminals in terms of families or equivalences to other documented terminal types.

— Need not consist of an exhaustive list of terminal models when the implementor considers that some terminal types are used too infrequently to be listed.

When a feature or utility is not supported on a specific terminal type, as allowed by this standard, and the implementation considers such a condition to be an error preventing use of the feature or utility, the implementation shall indicate such conditions through diagnostic messages or exit status values or both (as appropriate to the specific utility description) that inform the user that the terminal type lacks the appropriate capability.

This standard uses a notational convention based on historical practice that identifies some of the control characters defined in 2.4 in a manner easily remembered by users on many terminals. The correspondence between this '`control-`*char*'' notation and the actual control characters is shown in Table 2-101. When this standard refers to a character by its `control-` name, it is referring to the actual control character shown in the Value column of the table, which is not necessarily the exact control key sequence on all terminals. Some terminals have keyboards that do not allow the direct transmission of all the nonalphanumeric characters shown. In such cases, the system documentation shall describe which data sequences transmitted by the terminal shall be interpreted by the system as representing the special characters.

NOTE: The notation uses uppercase letters for arbitrary editorial reasons. There is no implication that the keystrokes represent control-shift-letter sequences.

Table 2-101 – Control Character Names

Name	Value	Name	Value	Name	Value
`control-A`	<SOH>	`control-L`	<FF>	`control-W`	<ETB>
`control-B`	<STX>	`control-M`	<CR>	`control-X`	<CAN>
`control-C`	<ETX>	`control-N`	<SO>	`control-Y`	<EM>
`control-D`	<EOT>	`control-O`	<SI>	`control-Z`	<SUB>
`control-E`	<ENQ>	`control-P`	<DLE>	`control-[`	<ESC>
`control-F`	<ACK>	`control-Q`	<DC1>	`control-\`	<FS>
`control-G`	<BEL>	`control-R`	<DC2>	`control-]`	<GS>
`control-H`	<BS>	`control-S`	<DC3>	`control-^`	<RS>
`control-I`	<HT>	`control-T`	<DC4>	`control-_`	<US>
`control-J`	<LF>	`control-U`	<NAK>	`control-?`	<DEL>
`control-K`	<VT>	`control-V`	<SYN>		

Terminal Characteristics Rationale. *(This subclause is not a part of P1003.2a)*

There are at least two types of terminals that might have difficulty with certain utility features: hard copy terminals and block-mode terminals.

It is the intention of the developers of this standard that the list of supported terminals and features be presented as tersely as the implementors may desire. Subsets of terminal types may be included, based upon marketing requirements. Generous use of correct global statements is preferable to itemized lists of features for each of hundreds of terminal types. Attention should be directed to clarifying nonsupported features on terminal types that would otherwise be expected to support such features, due to those terminals' similarity to documented types.

It is the intention of the developers of this standard that when a terminal cannot support a certain feature, the utility should produce a useful diagnostic status or message explicitly informing the user of this fact. It should not produce a syntax error, ''unknown command,'' or ''unknown feature'' message or status. (This intent is not in the normative text because it is difficult to describe except with examples.)

Terminals that do not have the ability to send some of the special characters are in wide industry use, although not on the historical systems that form the basis for this standard. For example, the 3270 family of mainframe synchronous terminals cannot directly send the control characters. Therefore, implementations have to document how to do so; it could be by using a series of keystrokes, or by special function keys. In the 3270 example, PF3 has been used as a `control-D` equivalent.

The {POSIX2_CHAR_TERM} option was added to allow a vendor supporting only block-mode terminals to still conform, even though some utilities, such as `vi`, have limited usefulness. The `vi` editor is the only utility in POSIX.2 that can be entirely omitted if this option is not supported; others, such as `ex`, `more`, and `talk`, still have useful feature subsets that are unrelated to the terminal variety.

The decision on whether to require conformance to {POSIX2_CHAR_TERM} is strictly one of marketing. Even when this option is provided, however, there is no guarantee that *every* terminal on the system will support all features; the option implies that the software drivers and other support features are installed, but not that any of the terminals were actually hooked up.

Section 3: Revisions to Shell Command Language

⇒ **3.1 Shell Definitions.** *Modify the contents of clause 3.1, Shell Definitions, to add the following definition in the correct sorted order [disregarding the subclause number shown here].*

3.1 Shell Definitions

3.1.101 alias name: A word consisting solely of underscores, digits, and alphabetics from the portable character set (see 2.4) and any of the following characters:

`!` `%` `,` `@` 7

Implementations may allow other characters within alias names as an extension. 7

Rationale. *(This subclause is not a part of P1003.2a)*

The characters allowed in alias names are from historical practice.

⇒ **3.3.1 Alias Substitution.** *Modify the contents of clause 3.3, Token Recognition, to add the following subclause:*

3.3.1 Alias Substitution

The processing of aliases shall be supported if the system supports the User Portability Utilities Option.

After a token has been delimited, but before applying the grammatical rules in 3.10, a resulting word that is identified to be the command name word of a simple command shall be examined to determine if it is an unquoted, valid alias name. However, reserved words in correct grammatical context shall not be candidates for alias substitution. A valid alias name (see 3.1.101) shall be one that has been defined by the `alias` utility (see 5.1) and not subsequently undefined using `unalias` (see 5.31). Implementations also may provide predefined valid aliases that are in effect when the shell is invoked. To prevent infinite loops in recursive aliasing, if the shell is not currently processing an alias of the same name, the word shall be replaced by the value of the alias; otherwise, it shall not be replaced.