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

Interpretation Request #26
Topic: option constants **Relevant Sections:** unistd.h

"Options" has not been properly updated to account for the new meaning of <unistd.h> option constants. It only talks about options being supported or not supported; it doesn't distinguish between support indications made at compile time and at runtime. The language for options constants does not state the requirements for presense of headers, data types, and functions in the case that a symbolic constant is not defined.

In section 2.1.6 Change paragraphs 1 and 2 from: The symbolic constants defined in <unistd.h>, Constants for Options and Option Groups reflect implementation options for IEEE Std 1003.1-2001. These symbols can be used by the application to determine which optional facilities are present on the implementation. The sysconf() function defined in the System Interfaces volume of IEEE Std 1003.1-2001 or the getconf utility defined in the Shell and Utilities volume of IEEE Std 1003.1-2001 can be used to retrieve the value of each symbol on each specific implementation to determine whether the option is supported. Where an option is not supported, the associated utilities, functions, or facilities need not be present. To: The symbolic constants defined in <unistd.h>, Constants for Options and Option Groups reflect implementation options for IEEE Std 1003.1-2001. These symbols can be used by the application to determine which of three categories of support for optional facilities are provided by the implementation:

1. Option not supported for compilation.
The implementation advertises at compile time (by defining the constant in <unistd.h> with value -1, or by leaving it undefined) that the option is not supported for compilation and, at the time of compilation, is not supported for runtime use. In this case, the headers, data types, function interfaces and utilities required only for the option need not be present. A later runtime check using the fpathconf(), pathconf(), or sysconf() functions

defined in the System Interfaces volume of IEEE Std 1003.1-2001 or the getconf utility defined in the Shell and Utilities volume of IEEE Std 1003.1-2001 can in some circumstances indicate that the option is supported at runtime. (For example, an old application binary might be run on a newer implementation to which support for the option has been added.)

2. Option always supported.
The implementation advertises at compile time (by defining the constant in <unistd.h> with a value greater than zero) that the option is supported both for compilation and for use at runtime . In this case, all headers, data types, function interfaces and utilities required only for the option shall be available and shall operate as specified. Runtime checks with fpathconf(), pathconf(), or sysconf() shall indicate that the option is supported. 3. Option might or might not be supported at runtime. The implementation advertises at compile time (by defining the constant in <unistd.h> with value zero) that the option is supported for compilation and might or might not be supported at runtime . In this case, the fpathconf(), pathconf(), or sysconf() functions defined in the System Interfaces volume of IEEE Std 1003.1-2001 or the getconf utility defined in the Shell and Utilities volume of IEEE Std 1003.1-2001 can be used to retrieve the value of each symbol on each specific implementation to determine whether the option is supported at runtime. All headers, data types, and function interfaces required to compile and execute applications which use the option at runtime (after checking at runtime that the option is supported) shall be provided, but if the option is not supported at runtime they need not operate as specified. Utilities or other facilities required only for the option, but not needed to compile and execute such applications, need not be present.

If an option is not supported for compilation, an application that attempts to use anything associated only with the option is considered to be requiring an extension. Unless explicitly specified otherwise, the behavior of functions associated with an option that is not supported at runtime is unspecified, and an application that uses such functions without first checking fpathconf(), pathconf(), or sysconf() is considered to be requiring an extension. In <unistd.h> section: Constants for Options and Option Groups Replace the first six paragraphs with: The following symbolic constants, if defined in <unistd.h> shall have a value of -1, 0, or greater, unless otherwise specified below. If a symbolic constant is not defined or is defined with the value -1, the option is not supported for compilation. If it is defined with a value greater than zero, the option shall always be supported when the application is executed. If it is defined with the value zero, the option shall be supported for compilation and might or might not be supported at runtime. See Section 2.1.6 for further information about the conformance requirements of these three categories of support.

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

The current wording regarding options was motivated as a response to the number of amendments produced during the 1990s. The intent was that implementations of the standard need not change to show that they do not support a new feature added by an amendment or a companion standard building on this document (note that amendments are out of scope for this revision). It was envisaged that applications could make a runtime call to getconf within a Makefile build optional code, and/or possibly use the dlopen() function at runtime to link in objects built using the option. There was agreement that this is a perhaps a bit convoluted and that we should forward concerns to the sponsor about the current wording for option constants.