IEEE Standards Interpretations for IEEE Std 1003.1c™-1995 IEEE Standard for Information Technology--Portable Operating System Interface (POSIX(R)) - System Application Program Interface (API) Amendment 2: Threads Extension (C Language)

Interpretation Request #23
Topic: pthread_attr_setschedparam **Relevant Clauses:** 13.5.1.2

There is no way to validate the priority value passed. When this function is called the implementation does not know whether the application has or will be changing the policy via pthread_schedsetpolicy(). Since it doesn't have a policy that it knows the application wants it cannot veify that the priority is indeed valid. This is one of the reasons POSIX .1b did not provide totally separate routines to set the policy and priority. In POSIX.1b, anytime the policy is changed, the new priority must also be specified. It is possible to change the priority only, but the priority is changed for the policy that is in effect at the moment.

There is no way to do this with POSIX.1c thread attributes. The priority specified in pthread_attr_setschedparam() can only be validated if one of two things are done: a) a sched_param is also passed to pthread_attr_setschedpolicy() so that attributes act the same way as ALL other scheduling functions. or b) it is mandated that the priority specified in pthread_attr_setschedparam() must be a valid priority for the scheduling policy currently in the "schedpolicy" attribute of the specified attributes. This will force the applications to always set the policy first and allow the implementation to provide error checking on the priority. How is an implementation supposed to provide correct error checking on the sched_param structure passed to pthread_attr_setschedparam()? A similar problem exists with pthread_attr_setscope().

Most implementations will allow applications to use SCHED_FIFO and SCHED_RR for threads of PTHREAD_SCOPE_PROCESS but not for threads of PTHREAD_SCOPE_SYSTEM. However, unless it is specified that the permission checking for the scheduling pol-

icy and priority are done according to the scope attributes currently set, an implementation cannot perform proper permission checking on the policy and priority. For example: an application may call pthread_attr_setschedpolicy() to change the policy to SCHED_FIFO while the scope attribute is currently set to PTHREAD_SCOPE_SYSTEM.

The next statement may be a call to pthread_attr_setscope() to change the scope to PTHREAD_SCOPE_PROCESS. This ordering of code is currently allowed by POSIX.1c and thus an implementation cannot do proper permission checking of scheduling values. Should the description of the schedpolicy and schedparam attributes state that they check permissions based on the current settings of the contentionscope and schedpolicy attributes in the attribute structure? This will force applications to make these calls in a predefined order such that an implementation can provide proper error checking.

Interpretation Response
The standard is clear that setting the scheduling policy or priority values in the scheduling attribute object does not actually change the value for any threads. It is only when the attribute object is used in a pthread_create that the parameters are used and it is at that time that the parameters shall be consistent. The scheduling policy and priorities of an existing thread may be changed using the pthread_setschedparm function which requires both a policy and a priority.

Rationale for Interpretation
None.