Commenting template

ISO/IEC JTC 1/SC 22/WG XX NXXXX

			Balloted	**document:**	N4578		
				Vote:	Approve		
NB (ISO 3166)	**No.**	**Category**	**Clause, Sub-clause**	**Paragraph, Figure, Table**	**Comment and rationale**	**Proposed new text**	**Response**
AG	1	TL	7.22.4.7	page 352 para 2	Since quick_exit() disallows signal handlers to be called, what happens if a signal corresponding to a computational exception is generated during execution of one of the functions registered by at_quick_exit()?	Add a statement that the behaviour is undefined.	Agreed in principle In 7.22.4.7, add at the end: If a signal is raised while the quick_exit function is executing, the behavior is undefined.

AG	2	TL	7.25.1 The xtime structure	page 374 para 4	POSIX already defines two different structures to hold time, one of which, struct timespec, is almost identical to the xtime structure. It would be appropriate to merge the xtime and timespec structures.	Remove xtime from threads.h Change para 3 of <time.h> from "... which are arithmetic types capable of representing times; and struct tm which holds the components of a calendar time, called the broken-down time." To: " which are arithmetic types capable of representing times; struct timespec which is a structure type that holds a time specified in seconds and nanoseconds. The structure shall contain at least the following members, in any order. time_t tv_sec; long tv_nsec; and struct tm which holds the components of a calendar time, called the broken-down time." Globally replace "xtime" with "timespec", the "sec" member with "tv_sec, and the "nsec" member with "tv_nsec". [Note the remainder of this ballot uses xtime where appropriate. The global edit suggested here should be applied to these ballot comments if this comment is accepted]	Agreed in principle, See N1564

AG	3	TL	7.25.3.5 The cnd_timedwait function	page 376 para 1,2	"until after the time specified by the xtime object pointed to by xt" It is not clear whether xt specifies an absolute time or elapsed time from the start of the cnd_timedwait() call. I.e should applications just set xt to the length of the timeout, or should they call xtime_get(), add the length to the returned time, and then use that. The equivalent POSIX function pthread_cond_timedwait() takes an absolute time. There are good reasons for this: see the RATIONALE in the POSIX description of the function.	Clarify whether xt is an absolute time or the length of the timeout.	Agreed in principle, See N1564
AG	4	TL	7.25.3.6 The cnd_wait function	page 377 para 2	"If the mutex pointed to by mtx is not locked by the calling thread, the cnd_wait function will act as if the abort function is called." This requirement means mutexes must keep a record of ownership, which affects efficiency, and is inconsistent with cnd_timedwait() whose description states "The cnd_timedwait function requires that the mutex pointed to by mtx be locked by the calling thread."	Change "If the mutex pointed to by mtx is not locked by the calling thread, the cnd_wait function will act as if the abort function is called." to "The cnd_wait function requires that the mutex pointed to by mtx be locked by the calling thread."	AGREE

AG	5	TL	7.25.4.4 The mtx_timedlock function	page 379 para 2	"until the time specified by the xtime object xt has passed" It is not clear whether xt specifies an absolute time, or elapsed time from the start of the mtx_timedlock() call. I.e should applications just set xt to the length of the timeout, or should they call xtime_get(), add the length to the returned time, and then use that. The equivalent POSIX function pthread_mutex_timedlock() takes an absolute time. It would also make sense for mtx_timedlock() to be consistent with cnd_timedwait() in this regard. (The same issue has been reported separately for cnd_timedwait().)	Clarify whether xt is an absolute time or the length of the timeout.	Agreed in principle, See N1564
AG	6	TL	7.25.5.5 The thrd_exit function	page 381 para 2	Nothing is said about what happens if the last thread left running calls thrd_exit(). POSIX has the following requirement for the equivalent pthread_exit() function: "The process shall exit with an exit status of 0 after the last thread has been terminated. The behavior shall be as if the implementation called exit() with a zero argument at thread termination time." The C Standard should either require this behaviour, or should allow this behaviour and one or more other behaviours (and say the behaviour is implementation-defined).	Add a statement about the program terminating as if by a call to exit(0) after the last thread has terminated execution.	Agreed in principle, See N1564

AG	7	TL	7.25.5.7 The thrd_slee p function	page 382 para 2	"until after the time specified by the xtime object pointed to by xt" It is not clear whether xt specifies an absolute time, or elapsed time from the start of the thrd_sleep() call. Presumably it is intended to be elapsed time.	Clarify that xt specifies elapsed time.	Agreed in principle, See N1564
AG	8	TL	7.25.5.7 The thrd_slee p function	page 382 para 2	What happens if a signal handler is executed during execution of the thrd_sleep function? Does it return prematurely, or continue sleeping? If it returns prematurely, it would be useful for it to indicate the remaining sleep time. The equivalent POSIX function nanosleep() returns prematurely, and places the remaining time in an object pointed to by a second argument.	Either 1. specify that execution of a signal handler does not cause thrd_sleep to return prematurely, or 2. change the return type and/or arguments so that the remaining time can be returned to the caller; state that if execution of a signal handler interrupts thrd_sleep then it returns immediately; and describe how the function indicates whether it was interrupted and what the remaining time is.	Agreed in principle, See N1564
AG	9	TL	7.25.7.1 The xtime_ge t function	page 384 para 1	See also AG 2 -- Since xtime_get() is useful in its own right, not just with threads, it would be better for xtime_get() to be declared in <time.h> instead of <threads.h>. (Putting it in <time.h> would also mean xtime_get() becomes mandatory; if it is desirable for it to be optional, an alternative would be to have a new optional <xtime.h> header and put it in there.)	Move the declaration/description of the xtime type and the declaration of xtime_get() from <threads.h> to <time.h>. In <threads.h> require that it declare the xtime type and refer to <time.h> "(described in 7.26)".	Agreed in principle, See N1564

AG	10	TL	7.25.7.1 The xtime_get function	page 384 para 2	"sets the xtime object pointed to by xt to hold the current time" Since xtime represents time in seconds and nanoseconds, the "current time" here must be the number of seconds since a certain epoch, but nothing is stated about this epoch.	Either the epoch should be specified as a fixed time in the past (such as 1970-01-01T00:00:00 UTC as in POSIX) or the standard should state that the epoch is implementation-defined.	Agreed in principle, See N1564
AG	11	TL	7.25.7.1 The xtime_get function	7.25.7.1 The xtime_get function	This paragraph says that the value of base must be TIME_UTC, but TIME_UTC is not defined anywhere.	Add a requirement for TIME_UTC to be defined as a macro in the same header that xtime_get() and the xtime type are declared in. (I.e. in <threads.h> unless the latter are moved to a different header.)	Agreed in principle, See N1564