Doc No. P0489R0
Date: 2016-10-18
Project: Programming Language C++
Reply To: Barry Hedquist, beh@peren.com

Subject: WG21 Working Paper, Late Comments, ISO/IEC CD 14882

Attached is a WG21 Working Paper containing Late Comments on ISO/IEC CD 14882, Programming Language C++. These comments were not submitted in time to be registered as National Body Comments, but should be considered as possible issues against SC 22, N3151, ISO/IEC CD 14882.

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
					LATE COMMENTS	**THE FOLLOWING COMMENTS WERE SUBMITTED TOO LATE TO BE INCLUDED IN AN NB POSITION, BUT SHOULD BE CONSIDERED AS POSSIBLE ISSUES AGAINST THE WORKING PAPER SC22 N5131.**	
1		3.6.2 - 3.6.4		te	Several places in the standard do not correctly specify construction and destruction ordering in the presence of threads. For example, they often say “happens before” when they do not intend to allow memory_order_consume-based ordering. They allow concurrent construction at arbitrary points, and are unclear about the threads in which construction occurs, often allowing deadlock. CWG 1784 and 2046 point out additional issues.	Adopt wording improvements along the lines of P0250R2	
2		18.10.5		te	The current wording for signal handlers is inadequate. The current wording was never entirely consistent, e.g. it requires functions in the intersection of C and C++ with C linkage, something that doesn’t technically exist. The change to refer to C11 raises further questions as to whether thread_local variables are allowed in signal handlers. Whether or not a function is safe to use as a signal handler really depends only on whether it avoids certain constructs, not whether it is expressible in C.	Adopt P0270R1. This is a rebase of a document that was almost adopted in Oulu.	
3		8.5	p3	te	The current wording is unclear about when the `get` functions are called. It can be read as saying that it must be done eagerly, or as saying that it’s unspecified. Either option makes this unusable for types such as `expected`/`error_or`, where the value must not be accessed if there was an error.	Specify that each evaluation of the name calls the appropriate get function. This would further simplify the creation of reference wrappers by re-creating them for each use rather than having "variables".	
4					LBNL disagrees with the following USNB	The LBNL delegation would rather see these	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by ******)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
					comment: Adopt the full expression evaluation order proposal (P0145R3, failed to reach consensus when proposed during the plenary at the Oulu 2015 meeting (CWG Motion 9)).	features rigorously reviewed by the committee and added in a future standard.	
5					LBNL Disagrees with the following USNB Comment: Add requires clauses and expressions from the Concepts TS (N4377, failed to reach consensus when proposed during the plenary at the Jacksonville 2015 meeting (EWG Motion 1)).	The LBNL delegation would rather see these features rigorously reviewed by the committee and added in a future standard.	
6					LBNL disagrees with the following USNB Comment: Change statically-checkable conditions in Requires: clauses to Preconditions: clauses in the Standard Library (P0411R0, in the post Oulu 2015 meeting mailings).	The LBNL delegation would rather see these features rigorously reviewed by the committee and added in a future standard.	
7					LBNL disagrees with the following USNB comment: Add default comparison operators (N4475 and P0221R2, failed to reach consensus when proposed during the plenary at the Oulu 2015 meeting (CWG Motion 10)).	The LBNL delegation would rather see these features rigorously reviewed by the committee and added in a future standard.	
8					LBNL disagrees with the following USNB comment: Add **operator**.() (P0416R0, CWG declined to make a motion proposing this for C++17 at the Oulu 2015 meeting as the wording was not ready).	The LBNL delegation would rather see these features rigorously reviewed by the committee and added in a future standard.	
9					LBNL disagrees with the following USNB comment: Change structured bindings syntax from [ ] to { } (P0144R1, EWG strongly preferred [ ] over { } at the Jacksonville 2015 meeting).	The LBNL delegation would rather see these features rigorously reviewed by the committee and added in a future standard.	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by ******)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
10				te	Adopt operator dot as approved by the EWG and with the ambiguity problem found by CWG resolved.	Adopt the proposal as described in P0416r0 and P0252R2 with the revised wording addressing CWG's observations.	
11				te	Adopt "std::byte" as approved by EWG for C++17.	Adopt the proposal as described in P0298r1.	
12		8.5	1	te	Revert the structured binding syntax from [] to {}. The original proposal used {}, but it was asserted that [] was aesthetically better and introduced no problems. Unfortunately, using [] introduces ambiguities related to attributes and lambdas. These ambiguities can be handled, but they don't occur for {}. The similarity between [] for decomposition and for lambda capture was used as an argument for [] over {}, but that similarity is misleading (and therefore potentially confusing) because the semantics are totally different: introduction of new names as opposed to creating bindings to names in scope. The {} notation is more likely to fit with a future expansion into the area of functional-programming pattern matching.	Change the structured binding delimiters to curly braces.	
13		7.1.6	[1] and [3]	te	Whatever else "inline variables" do, they will cause harm by making it easier to introduce global variables and data races.	Remove "inline variables" from the CD. In other words, back out p0386r2. Instead, consider adjusting the definition of ODR to allow in-class initialization of static data members to be taken as definition and to allow constexpr values to be unified by the linker.	
14				te	Default comparisons, as described in P0221R3 proved controversial. The vast majority of negative comments related to operators < and <=. Operators == and != represent a coherent and useful subset of that proposal.	Adopt the (apparently) non-controversial default == and != from P0221R2	
15		27.10.8.4.11	3.1	ed	A "mismatched element" cannot be equal to an iterator.	Rephrase in terms of the iterators produced by `mismatch()`.	
16		27.10.8.4.11	3.3.1	ge	`path("c:foo").lexically_relative ("c:\\bar")==path("..\\foo")` is nonsense.	Don't construct `..` elements for the special leading components.	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by ******)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
17		27.10.8.4.11	3.3.2	te	The behavior illustrated in the last assertion is appropriate only for differing roots, and as given makes the function useless for relative paths.	Again, treat roots specially (instead of `begin()`).	
18		27.10.8.4.11	4	te	The behavior in the case corresponding to the previous comment is disastrously wrong.	Fix the underlying function as above.	
19		27.10.8.5	2	ge	It is surprising that a `path` is a container of `path`s.	Again, use string_type for individual components (including the special leading components for consistency).	
20		27.10.11	1	ge	The anemic status structure causes inefficiency and race conditions from multiple queries.	Add to it if at all possible (and then use `const file_status&` parameters everywhere, as befits a larger structure).	
21		27.10.12		te	This class is just a trivial wrapper for `path`.	Remove it and use `path` instead (or `string_type`; see below).	
22		27.10.13	6	te	It is cumbersome to always assemble the full `path` including the iterator's directory.	Follow, among many others, Python's `os.listdir()` and supply basenames; then just use `string_type`, since it's one component.	
23		27.10.14.1	28	te	This function name is an ugly implementation detail.	Call it `skip_subdirectory()` or just `skip()`.	
24		27.10.15.1	1	ge	`absolute(path("c:foo"),path ("d:\\bar"))==path ("c:\\bar\\foo")` is nonsense.	Some redesign is needed to address the unfortunate existence of drive-relative filenames that are neither absolute nor relative.	
25		27.10.15.4	4.2	ge	What attributes are copied?	Specify them.	
26		27.10.15.6	5	te	The time complexity assumes that the individual syscalls take constant time.	Specify count of syscalls, or drop the intuitively obvious complexity statement altogether.	
27		27.10.15.7	5	te	The restriction of attribute specification to copying from an exemplar prevents sensible security measures like mode 700 for race prevention.	Add overload that accepts a `perm`.	
28		27.10.15.10		ge	It's important that this function might misbehave if	Mention `create_directory_symlink()`.	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by ******)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
					`to` is a directory.		
29		27.10.15.13		te	Access to the file ID (device/inode pair, for POSIX) is superior to an equivalence test: it can be used as a lookup key and avoids re-checking a single reference file repeatedly.	Add a function that returns an unspecified type that supports all comparisons, possibly to replace `equivalent()`. Include support for `lstat()`.	
30		27.10.15.13	3	ed	The explicit check for `s1==s2` is ill-formed (there is no `operator==()`) and could race (if, say, the permissions are changed concurrently).	Omit the check: it's enough that they "resolve to the same file system entity", however the implementation can determine that.	
31		27.10.15.13	5	te	It is surprising that the `error_code` overload throws in certain, very specific, cases.	Rely on the normal error handling for non-existent paths.	
32		27.10.15.13	5	te	In particular, it is unhelpful to reject special files outright.	Let the implementation compare their identities.	
33		27.10.15.21		ge	It is surprising that, for example, a FIFO is both `is_fifo()` and `is_other()`.	Rename `is_other()` to `is_special()`, or remove it.	
34		27.10.15.25	5	te	Could a direction of error in the stored time be guaranteed?	Add a postcondition if so.	
35		27.10.15.26		te	Why is there no way to read permissions?	Add an overload to do so.	
36		27.10.15.26		te	The `error_code` overload should be `noexcept`.	Add it.	
37		27.10.15.26	2	te	Why have `symlink_nofollow`, `add_perms`, and `remove_perms` share a bitmask with the actual permissions?	Make these separate types (just `bool` for nofollow) and parameters.	
38		27.10.15.26	2	te	There is an implication that the operation is atomic, which is unimplementable.	Clarify that it is a non-atomic convenience (or that it can cause a file system race, if that is desired).	
39		27.10.15.29	3	ge	Several surprises can occur with symlinks: Consider `/box/src/`, a symlink `/box/link` → `/dest`, and `/dest/file`. Then `relative("/box/link/file","/box")` produces `../dest/file` instead of	Remove the function, and supply its specified expression as an example for `weakly_canonical()`; there is likely no consensus for a particular solution on which to standardize for the problem of generating relative paths in the presence of symlinks.	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by ******)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	Observations of the secretariat
					`link/file` (similarly for a source of `/box/src`). If the paths do not (yet) exist, they might be created including symlinks and have the answer be wrong.		
40		27.10.15.31	1	te	What happens if the path is `symlink/`? Some versions of Boost reject it with the (incorrect, according to POSIX) 'Not a directory: "symlink/"'	It is inconsistent to test `symlink` when `symlink/` was specified, so the obvious behavior is to remove the contents of the directory and then fail to remove the symlink.	
41		27.10.15.31	3	te	It would be useful to know the number of successful removals in case of error.	Return that number (so long as it is acceptable for error detection to be through the `error_code`).	
42		27.10.15.33	1	ed	`file_size()` has no argument.	Add `p`.	
43		27.10.15.33	3	ge	It is odd to be missing `ftruncate()` from POSIX given `truncate()`.	Add it, or a function like it.	
44		27.10.15.38		ge	The function name is inconsistent with other similar functions in this subclause.	Rename it to `system_absolute()`.	
45		27.10.15.38	6	ge	It appears that this function is unreliable exactly in the situation for which it is designed.	Remove it unless it can be made reliable.	
46		27.10.15.40	2	te	Do components for which errors occur (in determining whether they are symlinks) count as existing or not?	Presumably, specify that they do not, since they would be poor input to `canonical()` which expects them to exist (which would not be able to be verfiied).	
47		27.10.15.40	4	te	The file system's state can change at any time, so the suggested caching is incorrect.	Remove the suggestion, unless it is meant to be UB for such changes to occur.	

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by ******)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial

MB/ NC[1]	Line number (e.g. 17)	Clause/ Subclause (e.g. 3.1)	Paragraph/ Figure/ Table/ (e.g. Table 1)	Type of comment[2]	Comments	Proposed change	Observations of the secretariat

1 **MB** = Member body / **NC** = National Committee (enter the ISO 3166 two-letter country code, e.g. CN for China; comments from the ISO/CS editing unit are identified by ******)
2 **Type of comment:** **ge** = general **te** = technical **ed** = editorial