Document Number: WG21/N0952
X3J16/96-0134
Date: 25 September 1996
Project: Programming Language C++
Reply to: Dan Saks
dsaks@wittenberg.edu

X3J16 Meeting No. 21
WG21 Meeting No. 16
8 - 12 July 1996

Sheraton Towers
Stockholm, Sweden

1 Opening activities

Clamage convened the meeting as chair at 09:10 (MESZ) on Monday, 7 July 1996. Lajoie was the vice-chair, and Saks was the secretary.

Ericsson (represented by Jonnson) hosted the meeting.

1.1 Opening comments

1.2 Introductions

Saks circulated an attendance list each day, which is attached as Appendix A of these minutes.

1.3 Membership, voting rights, and procedures for the meeting

Clamage explained that this is a simultaneous meeting of WG21 and X3J16. WG21 and X3J16 will conduct separate votes on each formal motion.

Clamage explained who has voting rights in X3J16: an X3J16 member organization may vote at this meeting if it has paid its dues and has met X3's attendance requirements. Representatives of new X3J16 member organizations may not vote at this meeting.

1.4 Distribution of position papers, WG progress reports, WG work plans for the week, and other documents not distributed before the meeting

1.5 Approval of the minutes of the previous meeting

Saks submitted the minutes from the previous meeting (N0880 = 96-0062) for approval with this correction:

> Add the following agenda items before item 9:
>
> 7 WG sessions (if any time is left)
>
> 8 Distribution of formal motions

He explained that nothing happened here, but the numbering was obviously incorrect.

Motion by Saks/Bruck:

> Move we approve N0880 = 96-0062 with this correction as the minutes of the previous meeting.

Motion passed X3J16: lots yes, 0 no, 0 abstain.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

1.6 Agenda review and approval

Clamage submitted the proposed agenda (N0947 = 96-0129) for approval. Saks suggested elevating the work of the drafting committee to scheduled agenda items. (See further discussion under agenda item 1.10 below). Specifically, he recommended adding:

3.1 Drafting Formal Motions (9 pm)
4.1 Drafting Formal Motions (7 pm)
7.1 Drafting Formal Motions (10 pm)

Also, Plum asked to schedule a US TAG meeting immediately after adjournment on Friday.

Motion by Saks/Lajoie:

> Move we accept N0947 = 96-0129 with these additions as the agenda for this meeting.

Motion passed X3J16: lots yes, 0 no, 0 abstain.
Motion passed WG21: 6 yes, 0 no, 0 abstain.

1.7 Report on the WG21 Sunday meeting

Harbison summarized the outcome of Sunday's WG21 meeting. He said WG21 discussed the schedule for the drafting committee meeting(s). Further discussion is scheduled for item 1.10, below.

Harbison said this is his last meeting as WG21 convener. If all goes as planned, Plum will be confirmed as the new convener at the SC22 meeting in September, and will be convener at the November meeting.

Harbison also explained that Soop presented a liaison request from SC24/WG6 for extensions to support multithreading and concurrency. Bruck offered to look into the issue, but WG21 decided not to do anything about this until after this IS (international standard) is complete.

Finally, Harbison said WG21 decided to try to get all substantive technical issues resolved at this meeting. The hope is that no open issues will be carried forward to future meetings.

1.8 Liaison reports

==== WG14+X3J11 (C) ====

Plum reported that WG14 and X3J11 met in Amsterdam, Netherlands in June. They are working on the revision of the C standard, which they refer to as C9x. Their work on extended characters in identifiers and literals may affect C++, and he referred to issue to the C Compatibility WG to consider this week.

Plum said WG14+X3J11 also discussed the following issues that may also affect C++:
-- complex numeric types
-- IEEE arithmetic
-- LIA (language independent arithmetic)
-- future standard status of errno

Regarding the last item, Plum said he proposed to WG14+X3J11 the possibility of deprecating errno. It inhibits optimization on many architectures. He suggested discussing this in the C Compatibility WG.

Bruck asked Plum if WG14+X3J11 discussed the integer division issue he raised. Plum said he couldn't remember if they discussed it this time. He thought the answer from a previous meeting was "it's the same as in

Fortran".

Hartinger said he had heard that long long was accepted. Plum said he didn't think all the issues were resolved, but something along these lines will be part of C9x.

Stroustrup asked about the proposal to add classes to C9x. Plauger said it's a dead issue; the chief proponent (Jervis of Sun Microsystems) is no longer on the committee.

Plum also said that a complex arithmetic type, without a distinct imaginary type, is now part of C9x. WG14+X3J11 discussed what you must do to write complex math that compiles as both C and C++. The consensus was that it's possible, but just barely.

1.9 New business requiring actions by the committee

Clamage explained that Lajoie has resigned as vice-chair of X3J16. He said X3 no longer requires that technical committees have a vice-chair; however, he thought we should still have one. Clamage said we have two volunteers. We also need a new IR (international representative) to replace Plum should he become the convener. Harbison said he hoped we could get one of the volunteers to offer to be IR instead of vice-chair.

1.10 Drafting committee

Saks briefly explained that the drafting committee is responsible for preparing the formal motions in advance of the voting so that all committee members have the opportunity to understand the issues and consider how they will vote. Normally, the drafting committee meets on Wednesday evening to draft the formal motions. Unfortunately, our host scheduled a reception for Wednesday evening, so we scheduled the work for Monday and Tuesday nights as well. Saks asked each WG to do what it could to draft on Monday and Tuesday night, so that we can recover time lost Wednesday to the reception.

2 WG sessions

The committees recessed to WGs at 10:10 on Monday.

3 Technical session

4 WG sessions

The committees reconvened at 8:50 on Wednesday.

Clamage explained that X3J16 is responsible to respond to all of the ANSI (US) public comments. We have drafted replies to nearly all the comments, and they are available by ftp. X3J16 should vote to approve the replies. Clamage asked that anyone who presents a proposal that closes an issue should please note this for the record. We need to be sure we've addressed them all.

5 Working Paper for Draft Proposed Standard

Koenig presented the project editor's report (N0981 = 96-0163). He said there was an editing session immediately after the Santa Cruz meeting (in March). He thanked those who participated, and also those who helped edit the draft in the weeks that followed.

Koenig said he made numerous "bold changes" since the last WP. (A bold change is one that incorporates the effect of an editorial box without a specific vote by the committees.) He listed the bold changes, using the notation B(C) to mean "editorial box B in clause C": 6(5), 12(12.1), 13(12.4), 14(12.8), 15(12.8), 45(20.1.4), 46(20.1.4), 47(20.4.1), 48(20.4.1), 51(21.1.2), 56(22.1.1.1.1), 60(21.2.1.1.1), 61(22.2.1.5.1),

62(22.2.1.5.2), 63(22.2.2.1.2), 65(22.2.2.1.2), 67(22.2.2.2.2), 69(22.2.2.2.2), 73(22.2.3.1.1), 74(22.2.5.1.2), 76(23.3.1), 77(23.3.2), 78(24.2), 82(24.4.2.1), 86(24.5.3), 88(24.5.3.2), 89(24.5.3.2), 97(26.2.1), 101(27.1.2), 109(27.4.3), 119(27.6.1.2.1), 120(27.6.1.2.2), 121(27.6.1.2.2), 127(27.6.1.3), 128(27.6.2.3), 129(27.6.2.4.1), 130(27.6.2.4.1), 137(27.8.1), 138(27.8.1.1), 139(27.8.1.1), 141(27.8.1.4), 143(27.8.1.4), 145(27.8.1.4).

6 General session I

6.1 Core Language I WG

Lajoie reported that there was only one core issue from the US public comments still open, core-78, regarding whether extern "C" is part of the type system. She summarized a few alternative proposals (N0950 = 96-0132). The WG recommended the first of those, namely, that language linkage (extern "LANG") affects functions types (rather than just declarations).

Lajoie presented several examples from the paper, including:

```
extern "C" typedef void FUNC();
FUNC* pf1;
void (*pf2)();
pf1 = pf2;  // ill-formed
```

Someone noted that extern "C" and extern "C++" are equivalent on some systems, and bemoaned the lack of an implicit conversion between the pointers pf1 and pf2. Lajoie explained that there may be architectures where the conversion is impossible. Implementations may allow the conversion using

```
pf1 = reinterpret_cast<FUNC*>(pf2); // undefined
```

Lajoie explained the effect of language linkage on overload resolution using the example from the paper. She also explained that

```
void f(int);
extern "C" void f(int); // ill-formed
```

is ill-formed (as always) because it attempts to overload functions that differ only in language linkage. Unruh noted that

```
extern "C" void f(int);
void f(int);            // okay
```

is okay because the second declaration redeclares f with the same linkage as in the prior declaration.

Bruns objected that the cast’s behavior should be "implementation-defined", not "undefined". Lajoie explained that "implementation-defined" means that all implementations must support it in some way. She said the WG did not want to obligate all implementations to support it, and undefined behavior avoids any such obligation. Unfortunately, if the behavior is undefined, a compiler need not issue a diagnostic if it does not allow the conversion.

Lajoie said the WG preferred that an implementation issue a diagnostic if it can’t do the conversion, but the WP has no name for behavior that needs to be diagnosed if and only if it is not supported. Plum said WG14+X3J11 considered using the term "implementation-specified" to describe such behavior. Glassborow suggested that using static_cast is a way to get a compiler diagnostic if the implementation does not support the conversion.

Stroustrup this proposal is both a step forward and a step backward. It

breaks every C++ program ever written under Unix.

Lajoie said the WG would reconsider the use of "undefined" and whether static_cast is more appropriate than reinterpret_cast.

Straw Vote: Who favors making language linkage part of function types as above? lots yes, 0 no, 3 abstain.

(See Motion 2 for the final form of the proposal.)

Lajoie presented the WG's resolution to issue core-420, namely, language linkage is ignored for member functions and operator functions (N0950 = 96-0132). Some people were confused about what "ignored" means here. Lajoie it means that means extern "C++" is redundant.

Straw Vote: Who favors this proposal? 14 yes, 5 no, 8 abstain.

Ball and others objected that ignoring language linkage for operator functions makes it impossible to pass the address of an operator function to a C function. He asked for another straw vote on "language linkage is ignored for member functions" by itself.

Straw Vote: Who agrees that language linkage should be ignored for member functions? lots yes, 0 no, 0 abstain.

Lajoie said the WG would simply treat "language linkage is ignored for member functions" as an editorial clarification.

Lajoie presented the WG's resolution to issue core-556 (from N0941 = 96-0123). (See Motion 3.)

Straw Vote: Who agrees with this recommendation? lots yes.

Miller presented a proposal to clarify name lookup in using declarations (N0905 = 96-0087). (See Motion 4.) The proposal addressed deficiencies in the wording of clause 3.4.2.2 paragraph 2 [namespace.qual]. In particular, the current wording prevents a using declaration from referring to a set of overloaded functions, as in:

```
namespace N {
    void f(int);
    void f(float);
}

void g() {
    using N::f;   // ill-formed: not a single-selection context
}
```

Miller summarized the two options from the paper and asked the committee to choose one:

Option 1: Import all names.
-- This is consistent with the treatment of overloaded functions (it brings in multiple declarations).
-- It makes the result of
```
        using X::m;  ...m...;
```
just like
```
        ...X::m...
```
-- Stroustrup believes this was the original intent.

Option 2: Import only non-hidden names.
-- This is consistent with lookup in other contexts.
-- It requires a syntax change to allow an elaborated-type-specifier in a using declaration.
-- It offers fine-grained control over the effect of a using declaration.

Stroustrup said a using declaration is supposed to introduce names, not types, into scope. He said he favored Option 1.

Straw Vote: Who favors...
    Option 1? 13
    Option 2? 1
    No change except clarification? 6

Lajoie presented the WG's proposed resolution to issue core-646, namely, that a using declaration cannot refer to a hidden base class member. (See Motion 5.)

Straw Vote: Who agrees with this proposal? 7 yes, 2 no, lots abstain.

Lajoie presented a proposal regarding issue core-636. The issue is whether a typedef-name can be used to declare an operator function. She gave this example:

```
typedef int FUNC();
class X {
public:
    operator FUNC;  // ill-formed?
};
```

Ball noted that this is already a syntax error. Lajoie withdrew the proposal.

Lajoie presented proposal regarding issue core-641, to clarify which allocation and deallocation functions are predeclared. (See Motion 6 for details.)

Corfield asked if a program can call ::operator new(size_t) without including <new.h>. Lajoie said it could. She added that this proposal clarifies that new with placement is not predeclared.

Straw Vote: Who favors this proposal? 21 yes, 4 no.

Lajoie presented the WG's proposed resolution to issue core-453 (N0942 = 96-0124), namely, that a program can call operator new only for new expressions or from other standard library functions. It cannot call operator new to allocate temporaries or to allocate data structures for exception handling or run-time type information. (See Motion 7.)

Straw Vote: Who agrees with this proposal? lots yes, 0 no, 8 abstain.

Anderson presented the WG's proposal regarding issue core-639, to clarify the lifetime of declarations in conditions. He presented this example:

```
void f(int i) {
    while (T t = i) { }
}
```

and explained that the proposal clarifies that t is constructed and destroyed (after }) on each iteration.

No one objected.

Lajoie presented the WG's proposal to resolve issue core-598 regarding reference initialization and rvalues. She gave this example:

```
class X {
    const int& ri;
    X() : ri(16) { }  // ill-formed
                  ^ temporary destroyed here
```

```
};
```

and this example:

```
const T& f(...) {
    return T();       // ill-formed
}
^ temporary destroyed here
```

She said the WG recommended that both should be ill-formed.

Adamczyk said he didn't want to vote for this without seeing the proposed WP changes. He expected that it's very hard to specify what it means to return a temporary.

Corfield asked what the WG proposed to do about

```
const T& f(...) {
    T t;
    return t;
}
```

He observed that the return value is not bound to a temporary, but it has the same problem as the second example above.

Ball suggested that what the WG is proposing may not be computable.

Lajoie agreed to take this back to the Core WG.

Straw Vote: Who wants to make these programs ill-formed? 9 yes, 15 no.

Lajoie presented a proposal to specify the exception specifications of implicitly-declared functions (N0903 = 96-0085). (See Motion 8.)

Lajoie presented this example (similar to one near the top of page 2 in N0903 = 96-0085):

```
struct A {
    virtual ~A() throw(X);
};

struct B {
    virtual ~B() throw(Y);
};

struct D : A, B { }; // ill-formed
```

Gibbons thought the error should be generated when the destructor is generated, not when the class is defined.

Straw Vote: Who favors this proposal? 18 yes, 1 no, 6 abstain.

Stroustrup presented his proposal to acknowledge that garbage collection is possible in a C++ implementation (N0932 = 96-0114). He explained that a common complaint about C++ is that it can't do garbage collection. He says this is not true. It's permitted but not required, and it's in commercial use.

Stroustrup explained that the proposal has only one normative part: to add __COLLECTING as a macro that programmers can test to determine if collecting is on. He said the WG discussed the meaning of the value of __COLLECTING. They agreed that if __COLLECTING is 1, then collecting is on. However, there was disagreement about whether a value of 0 should mean "not on" or "not guaranteed to be on". Stroustrup said he recommended the latter, which is the same guarantee as in the current WP. Koenig said he preferred that 0 mean "not on".

Straw Vote: Who favors this proposal? 8 yes, 10 no, 13 abstain.

6.2 Core Language II WG

Adamczyk presented a pair of proposals to make "narrowing" conversions less favorable in overload resolution (N0922 = 96-0104 and N0923 = 96-0105). He said the WG was sympathetic to the goals of the paper; however, the suggested approach didn't solve enough of the problems and adds additional complexity to overload resolution. He suggested that the "right" solution would be to eliminate implicit narrowing conversions, but that's not practical. However, we can deprecate them.

Straw Vote: Who favors deprecating implicit narrowing conversions?
7 yes, 12 no, 9 abstain.

Adamczyk then presented a proposal to change the type of string literals to be const-qualified (N0896 = 96-0078). Under this proposal, string literals would have type "array of const char" and wide string literals would have type "array of const wchar_t". (See Motion 9.)

Adamczyk said this issue has been raised before and rejected. However, this proposal has a new "wrinkle" -- it includes a standard conversion from string literal to char *, or wide string literal to wchar_t *, which would allow much existing code to continue to work as before.

For example,

```
char *p = "abc";                  // still OK
```

would work as before, but

```
char *q = i ? "abc" : "de";       // becomes ill-formed
char (&r)[4] = "abc";             // becomes ill-formed
```

would not. Furthermore,

```
catch (char *)
```

would no longer catch

```
throw "abc";
```

Straw Vote: Who favors this proposal?
WG21 only: 4 yes, 0 no, 2 abstain.
X3J16 only: 18 yes, 7 no.

Someone noted that, given:

```
void f(void *);
void f(char *);
```

this proposal makes the call f("abc") ambiguous.

Adamczyk presented a proposal to require that programs #include <typeinfo> before using typeid. (See Motion 10.) He explained that the WG agreed that you can't really do anything with typeid before including <typeinfo>, so there's little reason to allow even mentioning typeid prior to including <typeinfo>. About the only thing you can do is:

```
int main () {
    &typeid(int) == &typeid(int);
    return 0;
}
```

Straw Vote: Who favors this proposal? 23 yes, 5 no.

Adamczyk next presented a proposal to tighten the definition of old-style casts in terms of new-style casts. He explained that the current WP says an old-style cast can be any combination of static_cast, reinterpret_cast, and const_cast. The WG thought this was too broad. It allows:

```
int *p;
(float)p;     // OK by WP (reinterpret_cast + static_cast)
```

The WG recommended restricting the possible meanings for an old-style cast to more limited combinations of new-style casts. (See Motion 11 for those combinations.)

Straw Vote: Who favors this proposal? lots yes, 0 no.

Adamczyk presented a proposal to specify that converting a null pointer value to a cv-qualified pointer type is an atomic conversion. For example, the conversion

```
0 --> const char *
```

would be a single atomic conversion, not the conversion sequence

```
0 --> char * --> const char *
```

This will affect overload resolution. In particular, given:

```
void f(char *);
void f(const char *);
```

then f(0) will be ambiguous. (See Motion 12.)

No one objected.

Adamczyk presented a proposal to clarify that *all* semantic constraints on a default argument are checked at the argument's point of declaration. (See Motion 13.) Under this proposal, the following becomes ill-formed:

```
struct B;
extern B b;
struct A {
    void f(B = b);    // ill-formed
};
struct B { };
```

Adamczyk said the disadvantage of this rule is that writing mutually-dependent cases (as in issue core-531 of N0902 = 96-0084) becomes difficult, if not impossible.

Corfield expressed concern about applying this rule to templates. Adamczyk asked Corfield to discuss this off-line.

Straw Vote: Who favors this proposal? 17 yes, 5 no.

6.7 C Compatibility WG

Plum presented a proposal to close various core issues without action, and to refer other core issues to the editor. (See Motion 16.)

No one objected to this proposal.

Nelson presented a proposal to change the syntax for UCN (universal-character-name) to use \ instead of ??. (See Motion 17.) This clarifies that a UCN is not a trigraph.

No one objected.

Nelson then presented a proposal to clarify the phases of translation. (See Motion 18.) This proposal makes the file-to-host character mapping of translation phase 1 implementation-defined. It also clarifies that producing UCNs via macro "games" would not be portable.

No one objected.

6.3 Core Language III WG

Bruck presented the WG's proposed resolution to various exception handling issues (from N0969 = 96-0151). (See Motion 19.)

Hartinger raised a concern about the resolution to issue 541, which states that a function-try-block in main() can't catch an exception thrown from the constructor or destructor of a global static. In particular, he asked how a program catches such an exception. Bruck said it can't. Unruh added that such exceptions result in calling terminate().

Bruck explained that the proposal for issue 631 requires that exception specifications must match in a function's declaration and definition. A diagnostic is required for a mismatch in a single translation unit, but not across translation units (as with all other ODR violations).

No one objected to this proposal.

Spicer presented resolutions to various (minor) template issues. (See Motion 20.)

He noted that the proposed resolution to issue 6.45 eliminates "guiding declarations". For example, in

```
template <class T> T max(T, T);   // 1
int max(int, int);                // 2
```

the declaration on // 2 used to guide instantiation of the template on // 2. Now it is a function unrelated to the template.

No one objected to this proposal.

Unruh proposed new rules to define the term "dependent name". (See Motion 21.) He explained that these rules:
-- are based on syntax
-- determine dependency at template definition time

Gibbons added that this proposal eliminates hidden dependency. Also:
-- dependency propagates up the syntax tree
-- dependency stops at an explicit cast

No one objected to this proposal.

Stroustrup presented a proposal to refine the template compilation model and affirm separate compilation of templates (based on N0906 = 96-0088, the "SGI proposal").

Stroustrup explained that one of the problems the WG addressed was that compilers distinguish template definitions that are supposed to be in the current translation unit from those that are supposed to be found in other translation units. The WG had to devise a way to distinguish the two, and decide which would be the default.

Stroustrup said the WG decided that the inclusion model would be the default. That is, by default, template definitions must be available,

and are only accessible, in the translation unit where they appear. The WG proposed to use a new keyword "export" to indicate templates whose definitions may be compiled separately. For example:

```
export template <class T> void f(T);

export template <class T> class X;
```

"export" could appear on either the template declaration or definition to provide greater flexibility for both library vendors and users. The ODR applies as ever.

Wilkinson explained changes to the current proposal compared with what he presented at the technical session on Monday. Among other things, they removed the part about intermediate context.

Gibbons explained that the WG is actually splitting the proposal into two parts: (1) affirm the separation model and (2) everything else. (See Motions 22 and 22b.) Stroustrup encouraged members to accept the proposal as a whole package.

Gibbons also said the WG recommended changing the default linkage for inline functions to extern because:
-- it's necessary for the SGI proposal
-- it's more consistent with the behavior of inline member functions

Straw Vote: Who favors the model-independent portions of the SGI proposal (including extern inline)? lots yes, 0 no, 1 abstain.

Anderson wondered if changing the default linkage for inline functions breaks code.

Straw Vote: Who favors extern inline by itself? lots yes, 0 no, 6 abstain.

(See Motion 23.)

WG21+X3J16 discussed the pros and cons of the separation model. Ball and Spicer expressed concern that we don't fully understand the model and it could lead to further problems. Bruck said separation is based on sound engineering.

Straw Vote: Who favors the SGI proposal with separate compilation?
WG21+X3J16: lots yes, 7 no, 1 abstain.
WG21 only : 4 yes, 1 no, 2 abstain.

Gibbons: presented a proposal to replace name injection at template instantiation with special lookup rules (N0878 = 96-0060). (See Motion 24.)

Straw Vote: Who favors this proposal?
WG21+X3J16: 15 yes, 1 no, 10 abstain.
WG21 only : 4 yes, 0 no, 2 abstain.

Gibbons presented a proposal to extend the special lookup rules to function names outside templates. (See Motion 25.) People discussed this for a while, with some expressing reluctance about it.

Straw Vote: Who wants to...
adopt this proposal now? 10.
defer consideration? 9.

6.4 Library I WG

Dawes reported that the WG resolved 56 issues. He presented the WG's recommendations to close various issues from Clause 17 (see Motion 26),

clause 18 (Motion 27), clause 19 (Motion 28), clause 20 (Motion 29), and clause 21 (Motion 30). Dawes also recommended adding relational operators to standard library classes (from N0967 = 96-0149). (See Motion 30b.)

Dawes explained that the WG discussed the standard library's header inclusion policy. He offered the following background:

-- A template in the C++ standard library may be specialized by users, but it may be specialized only in its "home" namespace. Thus, there's no conforming way a standard library implementor can use names of standard library templates without exposing those names to users.
-- A given C++ standard header usually needs access to names from other standard headers (because signatures from one header often use names from another).

Dawes presented alternative header inclusion policies:

1) No standard library header may include any other standard library header (this is policy in Standard C).

2) Any standard library header may include any other standard library header. This leads to the following problem. This program:

```
#include <string>
using namespace std;
int main() {
    cout << "Hello, C++ World" << endl;
    return 0;
    }
```

shouldn't compile because it uses, but doesn't include, <iostream>; however, it might compile inadvertently because <string> might include <iostream.h>.

3) For each standard library header, specify the other standard headers it must include, and let it include no others.

Clamage suggested another policy:

4) All standard library names are predeclared in namespace std, thus eliminating the headers.

Koenig suggested yet another policy:

5) The standard library has a namespace with a documented name in which all specializable templates are defined. Users who want to specialize a standard template must do so in that namespace.

Straw Vote: Who favors policy 2? lots yes, 3 no, 6 abstain.

Straw Vote: Who favors leaving this issue open? 11 yes, 6 no.

6.5 Library II WG

Plum presented the WG's recommendation to close some iostream and locale issues (from N0920 = 96-0012) (see Motion 31), and also remove operators << and >> from class locale (see Motion 32).

No one objected to these recommendations.

Plum reported that there were no more unresolved locale issues. Applause.

Schwarz said the WG had numerous recommendations which closed many iostreams issues. He offered to go over resolutions for anyone who was

interested. No takers. (See Motions 33 through 39.)

He did go into detail on one issue; the effect of width on the char inserter. Given

```
cout << setw(10) << 'a';
```

the library as in the current WP does not pad the output to write 'a' in a width of 10. This is "wrong" but should be retained because it is historically a "feature".

Straw Vote: What prefers that the padding apply to inserted characters?
23 yes, 0 no.

Issue 27-501 in N0964 = 96-0146 reflects the result of this straw vote. (See Motion 34.)

Schwarz also explained a proposal regarding insertion and extraction of char, signed char and unsigned char (N0918 = 96-0100).

No one objected to this proposal.

The committees recessed at 17:30 on Wednesday and reconvened at 08:40 on Thursday.

6.8 Out of the Blue

Simonsen offered to maintain a WEB site for WG21.

No one objected to having him do so. (See Motion 47.)

Simonsen also asked WG21's permission to put the draft and other committee documents on WG14's WEB site. (See Motion 48.)

Straw Vote: Who wants to grant this request? lots yes, 0 no.

6.5 Library II WG (continued)

Schwarz resumed his summary of the WG's proposals regarding iostreams. Plum observed that the WG wasn't unanimous on all of the issues, but all were willing to support the entire package.

Schwarz explained that basic_ios currently has an operator bool(), where the classic design used operator void *(). This operator was to support testing a stream in a conditional, as in:

```
while (cin >> n)
    ...
```

Unfortunately, this allows cout >> 1 as a valid expression. Thus, the library will revert to using operator void *() and operator!(). (See issue 27-203 in N0964 = 96-0146.)

Koenig suggested that it should use operator const void *() rather than operator void *().

Straw Vote: Who favors this proposal (as per the WG's recommendation)?
15 yes, 0 no.

6.6 Library III WG

Becker summarized the WG's recommendation to close numerous issues from, and fix assorted problems in, clauses 23 through 26. (See Motions 40 and 45.)

6.2 Core II WG (more)

Adamczyk presented a proposal to change the pseudo-prototype for built-in operator[], and others (N0923 = =96-0105). (See Motion 14.) He presented this example from the paper:

```
struct S {
    operator char *();
    char& operator[](unsigned);
};
...
S s;
s[5]; // ambiguous
```

s[5] might mean either s.operator[](5) or (s.operator char *())[5]. s.operator[](5) has an exact match on the first argument, and uses a standard conversion on the second. (s.operator char *()) uses a user-defined conversion on the first argument, and an exact match on the second.

Adamczyk explained that this proposal changes the pseudo-prototype for built-in [] from

```
T& operator[](T*, I);
```

where I is the promoted integral type, to

```
T& operator[](T*, ptrdiff_t);
```

The proposal also makes a similar change to the pseudo-prototypes for built-in +, +=, - and -= with a pointer parameter.

He said this change:
-- improves the situation for those who have ptrdiff_t == long
-- allows writing portable code with just one function, namely,

```
    char& operator[](ptrdiff_t);
```

Straw Vote: Who favors this proposal? lots yes, 2 no.

Adamczyk also introduced a proposal to repair some minor problems with reference binding in overload resolution (N0972 = 96-0154). (See Motion 15.)

No one objected.

6.7 Remarks from the WG21 Convener

Harbison explained what would happen if WG21 were to "vote out" the committee draft (CD) at this point. An informal editing committee would edit the draft over the weekend following the meeting. Koenig (the project editor) would then continue editing the draft over the following weeks. Volunteers would then review the resulting draft. Harbison (as convener) would then attach the disposition of comments (from the previous ballot) and forward the draft to SC22 for a CD ballot to begin at the end of September.

Plum said he thought WG21 agreed on Sunday to try to get WG21+X3J16 to agree that all substantive changes will be made this week (regardless of when we actually do the editing). He asked for a straw vote to support this. Much discussion followed.

Harbison asked WG chairs for an estimate of the number of issues that remained open. Lajoie said about 15 core issues (excluding template issues) were still open. Gibbons said the only remaining template issue regards intermediate context. Spicer noted that the template issues list has been empty at end of every meeting for the past year. He

thought more template issues would come up before the next meeting.

Dawes said all important issues from clauses 17 through have been cleared, but the header inclusion issue will probably come up again. Schwarz reported that 2 or 3 little issues remain for input/output. Becker said clauses 23 through 26 have no major issues left open.

WG21+X3J16 discussed the CD ballot. Corfield said the UK preferred to wait until November to vote on submitting the CD. Plum said he'd go along, but urged that we hold a straw vote to affirm that we will fix bugs, but not invent anything new in November. Plauger agreed with Plum, and urged that we word the resolution as strongly as possible that we will stop inventing.

Harbison said he'd meet with heads of national delegations to draft such a resolution.

The committees recessed at 11:50 on Thursday.

7 WG sessions (if any time is left)

8 Distribution of formal motions

The committees reconvened at 16:10 on Thursday.

9 General session II

Harbison presented wording for the resolution to submit the WP for a second CD ballot. (See Motion 49.) He pointed out that this resolution acknowledges that all CD ballot comments have been resolved.

Harbison explained that, if we can finish editing the draft during the week in Hawaii, we could print the draft over the weekend and save almost two of the four months we'd have lost by slipping the project schedule one meeting. He asked that the host in Hawaii (Plum Hall) work out ways to get this done. This plan would also limit the scope of changes we would accept. He also recommended that NBs (national bodies) that want changes in the draft should come to the meeting with prepared wording.

Anderson wondered what we will be able to work on at the meeting in Nashua (March, 1997) while the CD ballot is in progress. Harbison replied that we must be careful about making changes during CD ballot, so should not take any votes to change the CD at that point. In response to concerns from Plum, Harbison suggested that X3J16 could work on drafting replies to US public comments.

Koenig asked what WG21 is allowed to do at the meeting during the CD ballot. Plum said WG21 could do editorial work. Koenig expressed surprise; he thought we could make no changes. Plauger explained that WG21 can fix typos and improve wording, but make no substantive changes. If something is demonstrably broken, we can discuss ways to fix it, but not make changes.

9.1 Core Language I WG

Lajoie explained an open issue regarding the maximum length for bitfields. C has a restriction that a bitfield cannot be longer than an int. She said the WG presented a resolution at the last meeting requiring that the size of a bitfield not exceed the size of its type. The vote split 18-to-18, so the issue went back to the WG. The WG's consensus was that the vote didn't mean "no limits," so they're now proposing that the maximum size of a bitfield is the size of the longest integral type in the implementation, i.e., long, regardless of the bitfield type (enum, int, whatever).

Clamage asked if this means a program can declare a 64-bit char. Lajoie said yes; the current WP has no restriction and this proposal would still allow it.

Koenig asked how many bits does a 64-bit char bitfield have. Lajoie explained that the memory has the indicated size, but fetching the value converts it to an rvalue and thus loses the extra bits. A program cannot access the lost bits in a portable way.

Wilkinson asked if there's any specification of where the significant bits are found. Lajoie said no. Charney asked what is the size of such a bitfield. Lajoie explained that sizeof yields the size of the type, not the size of the bitfield.

Spicer requested a straw vote on both this proposal and the one from the last meeting.

Koenig suggested that if bits beyond the size of the underlying type are just padding, then there's no obvious reason to have an upper bound at all. Lajoie replied that that is the status quo. Koenig asked if the WP says the additional bits are padding. Lajoie said no.

Straw vote: Who favors this [new] proposal? 9 yes, 6 no, 7 abstain.

Straw vote: Who favors the proposal from the last meeting? 8 yes, 7 no, 10 abstain.

Lajoie explained (again) that the status quo imposes no limit on the size of a bitfield, and leaves the semantics unspecified. Plum suggested voting to affirm the status quo. Stroustrup said the status quo needs to be clarified with respect to "padding."

Straw Vote: Who favors the status quo with clarification regarding "padding"? 7 yes, 6 no, 9 abstain.

Straw Vote: Who favors...
1) no size limit (status quo), with padding? 9
2) limit is sizeof(long), with padding? 6
3) limit is sizeof(underlying type) 10

Straw Vote (runoff between 1 and 3):
Who favors 1? 13
Who favors 3? 10

Lajoie agreed to keep the status quo but add editorial clarification regarding "padding".

9.2 Core Language II WG

No discussion.

9.4 Library I WG

Schwarz explained that an ad hoc group met to discuss the policy for including standard headers within other standard headers. They considered seven possible policies, and agreed to one suggested by Koenig:

-- standard templates live not in namespace std, but in namespace std::implementation
-- no other change to the header inclusion policy for the time being

Clamage suggested abbreviating the name "implementation". Plum said the group considered "_std" as the name. Schwarz suggested voting separately on the name. Plum said his vote would depend on the name.

Members discussed the proposal at length. Some members expressed un-

easiness about about approving this change without the time to consider all the ramifications. They also discussed whether we could remove this change at the next meeting.

Koenig suggested dropping the issue and leaving it to any NBs that want to raise this as a concern.

Straw Vote: Who favors this proposal? 3 yes, lots no.

9.3 Core Language III WG

Gibbons explained that the WG failed to consider an alternative proposal for fixing name injection for friends (N0913 = 96-0095). This proposal doesn't eliminate injection, but modifies it to reduce surprises. Some WG participants felt that all injection was bad, and they preferred to eliminate it entirely. Thus, the WG did not propose this alternative. Gibbons suggested that people should consider it before voting on the formal motion (Motion 24).

Spicer (the author of the alternative proposal) said he had concerns about the issue, but hadn't seen the text of the motion. He said he'd comment on the proposal on Friday. (See the discussion on Motion 24.)

Glassborow asked if the WG decided to change its recommendation. Gibbons said they did not.

9.5 Library II WG

No discussion.

9.6 Library III WG

Vandevoorde presented a proposal to establish anti-aliasing properties of valarray arguments, based on work done by Cray and SGI. The proposal allows a function receiving a valarray argument to assume that only that function accesses that valarray, and that multiple valarray arguments do not refer to the same valarray. For example, given

```
void f(valarray& a, valarray& b);
```

f can assume that a and b are not aliases for each other or any global valarray.

Straw Vote: Who agrees with this proposal? 14 yes, 0 no, 7 abstain.

The committeea continued to discuss the proposal for some time. Some contrasted this with the "restrict" keyword considered for C by WG14+ X3J11. Vandevoorde explained that this anti-aliasing is just for valarray; it's not as powerful as "restrict." However, it does allow using "restrict" where available.

Straw Vote (again): Who favors this proposal? 8 yes, 0 no, 15 abstain.

[No formal motion came from this.]

9.7 ANSI C compatibility WG

No discussion.

The committees recessed to 17:50 on Thursday.

10 WG sessions (if any time is left)

No time.

The committees reconvened at 8:45 on Friday.

Plum reported that Nelson has offered to be the new US International Representative (IR), and Miller has offered to be the new vice-chair for X3J16. He requested a brief X3J16 meeting to consider their offers, and approve the US delegation to WG21.

Straw Vote (X3J16 only): Who supports Nelson for IR? lots yes, 0 no.

Straw Vote (X3J16 only): Who supports Miller for vice-chair? lots yes, 0 no.

X3J16 thanked Plum for his past work as IR and Lajoie for her work as vice-chair. Applause.

Lajoie said she will handle the post-meeting mailing.

Motion (to approve the new US delegation to WG21) by someone/somebody else:

Move we approve Nelson, Clamage, Plum, Koenig, as Saks as the US delegation to WG21.

Motion passed X3J16: lots yes, 0 no, 0 abstain.

X3J16 thanked Vilot for his work as chair of the Library WG. Applause.

Plum closed the X3J16 meeting.

11 Review of the meeting

Clamage explained that we will conduct each X3J16 vote first. The results of each vote will determine how the US IR (Plum) will cast his vote in WG21.

Clamage counted 31 X3J16 voting members and 7 WG21 delegations.

Wilkinson explained that changes to N0973 = 96-0155 will appear as N0973R1 = 96-0155R1. N0973R1 = 96-0155R1 will be in the mailing, with a note that N0973 = 96-0155 was a first draft distributed only at the meeting.

11.1 Formal motions

1) Motion (to accept the WP) by Lajoie/Bruck:

Move we accept N0926 = 96-0108 as the current WP.

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

==== Core I (Lajoie) ====

2) Motion (to clarify that extern "C" affects function types) by Lajoie/Hartinger:

Move we amend the WP as follows

-- insert the following as a new paragraph before 7.5 [dcl.link] paragraph 1:

All function types, function names, and variable names have a language linkage, the specific semantics of which are implementation-defined. [Note: a particular language linkage may be associated with a particular form of representing external names, with a particular calling convention, etc.] The default language linkage of all function types, function names, and

variable names is C++ language linkage. Two function types with different language linkages are distinct types even if they are otherwise identical.

-- replace the last three sentences of 7.5 [dcl.link] (former) paragraph 2 with:

In a linkage-specification, the specified language linkage applies to all function types, function names, and variable names introduced by the declaration(s). [Example:

```
extern "C" void f1(void(*pf)(int));
                        // the name f1 and its function
                        // type have C language linkage;
                        // pf is a pointer to a C function
extern "C" typedef void FUNC();
FUNC f2;                // the name of the function f2 has C++
                        // language linkage and the function's
                        // type has C language linkage
extern "C" FUNC f3;     // the name of the function f3 and the
                        // function's type have C language
                        // linkage
void (*pf2)(FUNC*);     // the name of the variable pf2 has C++
                        // language linkage and the type of pf2
                        // is pointer to C++ function that takes
                        // one parameter of type pointer to C
                        // function
```

-- end example]

A non C++ linkage is ignored for the names of class members and for the top level function type of a class member function. [Example:

```
typedef void FUNC();
extern "C" {
    typedef void FUNC2();
    class C {
        void mf1(void(*pf)());
                        // the name of the function mf1 and the
                        // member function's type have C++
                        // language linkage; the parameter has
                        // type pointer to C function
        FUNC2* mf2(FUNC* pf);
                        // the name of the function mf2 and the
                        // member function's type have C++
                        // language linkage; the parameter has
                        // type pointer to C++ function
        static FUNC* q; // the name of the data member q has C++
                        // language linkage and the data
                        // member's type is pointer to C++
                        // function
    };
}
```

-- end example]

-- replace 7.5 [dcl.link] (former) paragraph 6 with:

[Note: because the language linkage is part of the function type, when a pointer to C function (for example) is dereferenced, the function to which it refers is considered a C function.]

and remove the editorial box.

-- add after the third sentence of 5.2.2 (expr.call) paragraph 1:

Calling a function through an lvalue whose function type has a language linkage that is different from the language linkage of the function type of the called function's definition is ill-formed; no diagnostic is required.

Motion passed X3J16: 28 yes, 3 no.
Motion passed WG21: 6 yes, 0 no, 1 abstain.

3) Motion (to define when a function is used) by Nelson/Lajoie:

Move we amend the WP by replacing 3.2 [basic.def.odr] paragraph 2 with:

An expression is 'potentially evaluated' unless either it is the operand of the sizeof operator ([expr.sizeof]), or it is the operand of the typeid operator and does not designate an lvalue of polymorphic class ([expr.typeid]). An object or non-overloaded function is 'used' if its name appears in a potentially-evaluated expression. A virtual member function is used if it is not pure. An overloaded function is used if it is selected by overload resolution from a reference in a potentially-evaluated expression. [Note: This covers operator overloading, allocation function for placement new, non-default initialization, user-defined conversions, as well as calls to named functions.] In addition, a copy-assignment function for a class can be used by an implicitly-defined copy-assignment function for another class; see [class.copy]. Also, an allocation or deallocation function for a class can be used by a new expression appearing in a potentially-evaluated expression; see [expr.new] and [class.free]. Similarly, a deallocation function for a class can be used by a delete expression appearing in a potentially-evaluated expression; see [expr.delete] and [class.free]. A default constructor for a class can be used by default initialization; see [dcl.init]. A constructor for a class can be used by various constructs; see [class.init]. A destructor for a class can be used by various constructs; see [class.dtor].

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 1 abstain.

4) Motion (to clarify name look up in a using declaration) by Miller/Lajoie:

Move we amend the WP as described in Option 1 of N0905 = 96-0087, and in addition:

-- replace the first 2 sentences of 3.4.2.1 [class.qual] paragraph 1 with:

If the nested-name-specifier of a qualified-id nominates a class, the name specified after the nested-name-specifier is looked up in the scope of the class (10.2). The name shall represent one or more members of that class or of one of its base classes (10).

-- delete the second note from 3.4.2.1 [class.qual] paragraph 1, which reads:

[Note: 10.2 describes how name look up proceeds in class scope.]

-- replace the first 11 sentences of 3.4.2.2 [namespace.qual] paragraph 2 with:

> Given X::m, where X is a namespace, let S be the set of all declarations of m in X and in the transitive closure of all namespaces nominated in using-directives in X and its used namespaces, except that using-directives are ignored in any namespace, including X, directly containing one or more declarations of m. If S is the empty set the program is ill-formed. Otherwise, if S has exactly one member, or if the context of the reference is a using-declaration (7.3.3), S is the required set of declarations of m. Otherwise if the use of m is not one that allows a unique declaration to be chosen from S, the program is ill-formed.

Motion passed X3J16: 29 yes, 2 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

5) Motion (to clarify name look up of a class member nominated by a using declaration) by Lajoie/Anderson:

Move we amend the WP by adding the following to the end of clause 7.3.3 [namespace.udecl] paragraph 15:

> The base class members mentioned by a using-declaration shall be visible in the scope of at least one of the direct base classes of the class where the using-declaration is specified.

Motion passed X3J16: 30 yes, 1 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

6) Motion (to clarify which allocation and deallocation functions are predeclared) by Lajoie/Hartinger:

Move we amend the WP by replacing the first 5 sentences of 3.7.3 (_basic.stc.dynamic_) paragraph 2 with:

> The library provides default definitions for the global allocation and deallocation functions. Some global allocation and deallocation functions are replaceable (_lib.new.delete_). A C++ program shall provide at most one definition of a replaceable allocation or deallocation function. Any such function definitions replace the default version provided in the library (_lib.replacement.functions_). The following allocation and deallocation functions are implicitly declared in a program
>
> ```
> ::operator new(size_t)
> ::operator new[](size_t)
> ::operator delete(void*)
> ::operator delete[](void*)
> ```
>
> [Note: a new-expression or delete-expression that refers to one of these functions without including the header <new> is well-formed.]

Motion passed X3J16: 27 yes, 4 no.
Motion passed WG21: 5 yes, 0 no, 2 abstain.

7) Motion (to clarify that operator new is only called as the result of a new expression or by library functions) by Lajoie/Bruns:

Move we amend the WP as proposed in N0942 = 96-0124, core issue 453.

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

8) Motion (to describe the exception specification of implicitly declared special member functions) by Anderson/Lajoie:

Move we amend the WP as proposed in N0903 = 96-0085.

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 1 abstain.

==== Core II (Adamczyk) ====

9) Motion (to make the type of string literals const) by Glassborow/Bruns:

Move we change the WP as follows:

-- In 2.13.4 [lex.string] paragraph 1, change "array of n char" to "array of n const char" and change "array of n wchar_t" to "array of n const wchar_t".

-- Add to 4.2 [conv.array] as a second paragraph:

A string literal (_lex.string_) that is not a wide string literal can be converted to an rvalue of type "pointer to char"; a wide string literal can be converted to an rvalue of type "pointer to wchar_t". In either case, the result is a pointer to the first element of the array. [Note: this conversion is deprecated. See Annex D.]

-- Add a new section to Annex D:

D.n [depr.string]

The implicit conversion from const to non-const qualification for string literals (_ptr.array_) is deprecated.

Motion passed X3J16: 25 yes, 6 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

10) Motion (to require inclusion of <typeinfo> before use of typeid) by Adamczyk/Ball:

Move we amend the WP by replacing 5.2.8 [expr.typeid] paragraph 6, with the following:

If the header <typeinfo> (_lib.type.info_) is not included prior to a use of typeid, the program is ill-formed.

Motion passed X3J16: 23 yes, 8 no.
Motion passed WG21: 5 yes, 1 no, 1 abstain.

11) Motion (to correct the definition of old-style cast in terms of new casts) by Adamczyk/Lajoie:

Move we amend the WP by replacing 5.4 [expr.cast] paragraph 5, with the following:

The conversions performed by
-- a const_cast (_expr.const.cast_),
-- a static_cast (_expr.static.cast_),
-- a static_cast followed by a const_cast,
-- a reinterpret_cast (_expr.reinterpret.cast_), or
-- a reinterpret_cast followed by a const_cast,
can be performed using the cast notation of explicit type conversion. The same semantic restrictions and behaviors apply. If a conversion can be interpreted in more than one of the ways listed above, the interpretation that appears first in the list is used, even if a cast resulting from that interpretation is ill-formed. [Example: continue with the existing example.]

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

12) Motion (to make conversion of a null pointer value to a qualified pointer type an atomic conversion) by Adamczyk/Bruns:

Move we amend the WP as follows:

-- Add at the end of 4.10 [conv.ptr] paragraph 1:

The conversion of a null pointer constant to a pointer to cv-qualified type is a single conversion, and not the sequence of a pointer conversion followed by a qualification conversion (_conv.qual_).

-- Add at the end of 4.11 [conv.mem] paragraph 1:

The conversion of a null pointer constant to a pointer to member of cv-qualified type is a single conversion, and not the sequence of a pointer to member conversion followed by a qualification conversion (_conv.qual_).

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

13) Motion (to define the semantics of default arguments) by Adamczyk/Lajoie:

Move we amend the WP by replacing 8.3.6 [dcl.fct.default] paragraph 5, sentence 1 with the following:

A default argument expression is implicitly converted (_conv_) to the parameter type. The default argument expression has the same semantic constraints as the initializer expression in a declaration of a variable of the parameter type, using the copy-initialization semantics (_dcl.init_). The names in the expression are bound, and the semantic constraints are checked, at the point of declaration.

Motion passed X3J16: 25 yes, 6 no.
Motion passed WG21: 6 yes, 0 no, 1 abstain.

14) Motion (to change the pseudo-prototypes for built-in [], etc.) by Adamczyk/Gibbons:

Move we amend the WP as follows:

-- In 13.6 [over.built] paragraph 14, replace the whole sentence with "For every T where T is a cv-qualified or -unqualified object type there exist candidate operator functions of the form" and replace "I" with "ptrdiff_t" in the 5 subsequent lines.

-- In 13.6 [over.built] paragraph 22, replace the whole sentence with "For every pair (T VQ), where T is a cv-qualified or -unqualified object type and VQ is either volatile or empty, there exist candidate operator functions of the form" and replace "I" with "ptrdiff_t" in the 2 subsequent lines.

-- In 13.6 [over.built] paragraphs 6, 7, 12, and 15 change "complete object type" to "object type".

Motion passed X3J16: 29 yes, 2 no.
Motion passed WG21: 6 yes, 0 no, 1 abstain.

15) Motion (to fix small problems with reference binding in overload

resolution) by Adamczyk/Wilcox:

Move we amend the WP as indicated in N0972 = 96-0154.

Gibbons observed that this change will cause functions that may not be callable to be selected in overload resolution. Adamczyk concurred.

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

==== C Compatibility (Plum) ====

16) Motion (to resolve several Core issues affecting C Compatibility) by Plum/Nelson:

Move that we close the following issues with no further action:

core-537 (can the implementation accept other constant expressions?)

and refer the following to the Project Editor for editorial action:

core-607 (UK issue 288, USA public comment #7 and #21: definition needed for character set(s))
core-600 (Swedish issue: should the value returned by integer division and remainder be defined by the standard?)
core-634 (do the phases of translation need to discuss shared libraries?)
core-643 (the term "integer code" needs to be defined)

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

==== C Compatibility (Nelson) ====

17) Motion (to change the syntax for universal-character-name) by Nelson/Plum:

Move we amend the WP by changing all occurrences of "??u" to "\u" and all occurrences of "??U" to "\U". (The affected clauses are 2.2 [lex.charset], footnote 18 in 2.10 [lex.name], and A.2 [gram.lex].)

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

18) Motion (to clarify phases of translation) by Nelson/Benito:

Move we amend clause 2.1 [lex.phases] of the WP as follows:

-- under phase 1, add to the end of sentence 1: "in an implementation-defined manner."

-- under phase 2, after sentence 1, add the following new sentence: "If a character sequence results which matches the syntax of a universal-character-name, the behavior is undefined."

-- under phase 4, after sentence 1, add the following new sentence: "If a character sequence is produced by token concatenation (_cpp.concat_) which matches the syntax of a universal-character-name, the behavior is undefined."

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

==== Core III (Bruck) ====

19) Motion (to resolve various exception handling issues) by

Bruck/Gibbons:

Move we amend the WP according to the changes proposed in N0969R2 = 96-0151R2.

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

==== Core III (Spicer) ====

20) Motion (to resolve small template issues) by Spicer/Ball:

Move we amend the WP as indicated in N971 = 96-0153.

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 1 abstain.

==== Core III (Unruh) ====

21) Motion (to refine the definition of dependent names) by Unruh/Miller:

Move we amend the WP as indicated in N0921R1 = 96-0103R1.

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

==== Core III (Stroustrup and Wilkinson) ====

22) Motion (to refine the template compilation model) by Austern/Gibbons:

Move we amend the WP as indicated in part 2 of N0973R1 = 96-0155R1.

Motion passed X3J16: 30 yes, 1 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

22b)Motion (to refine the template compilation model and affirm separate compilation) by Wilkinson/Corfield:

Move we amend the WP as indicated in part 1 of N0973R1 = 96-0155R1.

Motion passed X3J16: 21 yes, 9 no.
Motion passed WG21: 6 yes, 1 no, 0 abstain.

==== Core III (Gibbons) ====

23) Motion (to change the default linkage for inline functions) by Gibbons/Corfield:

Move we amend the WP as indicated in N0975 = 96-0157.

Motion passed X3J16: 31 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

24) Motion (to replace friend name injection with special lookup rules) by Gibbons/Glassborow:

Move we amend the WP as indicated in N0968 = 96-0150.

Spicer asked people to vote against this. He said that, while it solves a problem, it also breaks a lot of code. He mentioned several specific problems with the proposal. Gibbons rebutted several of Spicer’s points. He acknowledged that the proposal breaks some code, but it’s arguable about how much.

After some discussion, the committees took a short break. Upon returning, Spicer said he and others agreed over the break that the changes to the WP listed in N0968 = 96-0150 do not match the intent as discussed and approved during the straw votes. Koenig said it's a close call.

Motion passed X3J16: 19 yes, 11 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

25) Motion (to extend Koenig lookup to function names outside templates) by Gibbons/Corfield:

    Move we amend the WP by replacing the second sentence of 13.3.1.1.1 [over.call.func] paragraph 3 with:

        The name is looked up in the context of the function call using the normal rules for name lookup (_basic.lookup.unqual_), and also in the namespaces of any associated types for the arguments using the lookup rules described in _over.match.oper_.

Much discussion. Unruh, Spicer and Wilkinson said the WG considered this proposal rather hastily, and the decision was not unanimous. Stroustrup argued that this proposal makes lookup rules more consistent, and that's no small thing.

Motion passed X3J16: 20 yes, 10 no.
Motion passed WG21: 4 yes, 1 no, 2 abstain.

==== Library I (Dawes) ====

26) Motion (to resolve issues from the Clause 17 [Library Introduction] Issues List) by Dawes/Rumsby:

    Move we amend the WP as described in N0907R1 = 96-0089R1 by adopting the proposed resolutions for issues 001, 017, 018, 019 option 2, and 020. In addition, close issue 016 without taking any action.

Motion passed X3J16: 28 yes, 2 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

27) Motion (to resolve several issues from the Clause 18 [Language Support] Issues List) by Dawes/Rumsby:

    Move we:

    -- amend the WP as described in N0935R1 = 96-0117R1 by adopting the proposed resolutions for issues 020, 022, 023, 025, 027, 028, and 029.

    -- amend the WP as described in N0935R1 = 96-0117R1 by adopting the proposed resolution for issue 021 option 3, except change "extern" to "extern const".

    -- close issues 015 through 019, 024, and 026 without taking any action.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 1 abstain.

28) Motion (to resolve an issue from the Clause 19 [Diagnostics Library] Issues List) by Dawes/Rumsby:

    Move we close without taking any action issue 19-001.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 1 abstain.

29) Motion (to resolve several issues from the Clause 20 [Utilities] Issues List) by Dawes/Rumsby:

Move we amend the WP as described in N0937R1 = 96-0119R1 by adopting the proposed resolutions for issues 007, 024, 026 through 029, 031, and 034 through 038. In addition, close without taking any action issues 025, 030, and 033.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 1 abstain.

30) Motion (to resolve several issues from the Clause 21 [Strings] Issues List) by Dawes/Rumsby:

Move we amend the WP as described in N0948R1 = 96-0130R1 by adopting the proposed resolutions for issues 062, 093, 094, 106 through 110, 112. In addition, close without taking any action issues 085, 092, 096, 097 through 105.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 1 abstain.

30b)Motion (to add relational operators to standard library classes) by Dawes/Rumsby:

Move we amend the WP as proposed in N0967R1 = 96-0149R1.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

==== Library II (Plum) ====

31) Motion (to resolve certain open locales issues) by Plum/le Mouel:

Move we close, without further action, the following open locales issues (from N0920 = 96-0112):
22-004 (description of _byname facets too vague)
22-046 (locale facets for money/time/messages need semantics)
22-071 (messages_base isn't really needed)

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 1 abstain.

32) Motion (to remove stream operators for class locale) by Plum/le Mouel:

Move we amend the WP by removing the shift operators for class locale from subclause 22.1 [lib.locales] and from subclause 22.1.2 [lib.locale.global.templates]. Specifically remove these operators:

```
template <class charT, class Traits>
    basic_ostream<charT,Traits>&
    operator<<(basic_ostream<charT,Traits>& s, const locale& loc);
template <class charT, class Traits>
    basic_istream<charT,Traits>&
    operator>>(basic_istream<charT,Traits>& s, locale& loc);
```

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 1 abstain.

==== Library II (Schwarz) ====

33) Motion (to distinguish between operations on char and char_type) by le Mouel/Koenig:

Move we amend the WP to allow insertion of chars into streams

instantiated on some other character types as described in N0918 = 96-0100 proposal 2 (with some changes) and proposal 4. In particular:

-- remove insertors on the character type from basic_ostream
-- remove extractors on the character type from basic_istream
-- add global member template functions to insert char and other characters and to extract other characters
-- use the ctype facet member function widen to perform the conversion between char and other characters
-- allow insertion of signed char and unsigned char into streams instantiated on char
-- allow extraction of signed char and unsigned char from streams instantiated on char

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

34) Motion (to close various iostream issues) by le Mouel/Dawes:

Move we amend the WP as described in N0964 = 96-0146.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

35) Motion (to revise conversions in codecvt facet) by le Mouel/Becker:

Move we amend the WP as described in N0955R1 = 96-0137R1.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 1 abstain.

36) Motion (to take actions on various issues as recommend by the meeting of the Library WG in May) by le Mouel/Plum:

Move we amend the WP as described generally in N0914 = 96-0096 and specifically in N0944 = 96-0126, N0945 = 96-0127, N0958 = 96-0140, N0930 = 96-0112, N0954R1 = 96-0136R1, and N0946 = 96-0128 part 2 with the exception of issue 27-919.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

37) Motion (to clarify the role of pos_type and off_type traits) by le Mouel/Becker:

Move we amend the WP as described in N0957R1 = 96-0139R1.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

38) Motion (to revise description of stringbuf) by le Mouel/Becker:

Move we amend the WP as described in N0963 = 96-0145.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

39) Motion (to fix problems in string i/o operations) by le Mouel/Dawes:

Move we amend the WP as described in N0965 = 96-0147.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

==== Library III (Becker) ====

40) Motion (to adopt various changes to clauses 23 through 26) by Becker/Dawes:

Move we amend the WP as described in N0977 = 96-0159.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 1 abstain.

41) Motion (to close various issues without action) by Becker/Dawes:

Move we close the following library issues without taking any action: 23-041, -044, -045, -048, -050, and -061 in N0916 = 96-0098; issues 24-024, -028, -039, and -041 in N0915 = 96-0097; issues 25-004 and -014 in N0793 = 95-0193; issues 26-013, -015, -019, -020, -021, -024, -025, -026, -027, -031, and -033 in N0934 = 96-0116. Amend the WP by removing box 90, in accordance with closing issue 25-004.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

42) Motion (to clean up empty sections in Clause 23) by Austern/Becker:

Move we amend the WP as described in N0966R1 = 96-0148R1, thus closing issues 23-028 and 23-057.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 6 yes, 0 no, 1 abstain.

43) Motion (to resolve various streambuf and stream iterator issues) by Becker/Dawes:

Move we amend the WP as described in N0974 = 96-0156, thus closing issue 24-029.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

44) Motion (to correct the description of copy) by Becker/Dawes:

Move we amend the WP as described in the Proposed Resolution to issue 25-006 in N0793 = 95-0193, thus closing issue 25-006.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

45) Motion (to improve the usefulness of valarray) by le Mouel/Rumsby:

Move we amend the WP as described in N0962 = 96-0144, thus closing issue 26-009.

Motion passed X3J16: 30 yes, 0 no.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

==== Convener (Harbison) ====

46) Motion (to respond to the SC24/WG6 liason statement) by Harbison/Bruck:

WG21 directs the convener to transmit document SC22/WG21/N0961R1 = 96-0143R1 as the response to the SC24/WG6 liason statement SC24/N1577 = SC22/WG21/N0960 = X3J16/96-0142).

Motion passed WG21: 7 yes, 0 no, 0 abstain.

46b) Motion (to endorse the Convener's Report and the schedule change) by Harbison/Rumsby:

WG21 endorses the 1996 Convener's Report to SC22 (N0959R1 = 96-0141R1), including the change to the schedule and the change of convener.

Motion passed WG21: 7 yes, 0 no, 0 abstain.

47) Motion (to nominate Keld Simonsen as WG21 WWW maintainer) by Harbison/Bruck:

WG21 authorizes Keld Simonsen to maintain WG21's web-site within the guidelines set down by SC22, and directs the Convener to report this to SC22.

Motion passed WG21: 7 yes, 0 no, 0 abstain.

48) Motion (to distribute C++ documents within the C committee) by Plum/Bruns:

WG21+X3J16 authorizes ISO/IEC JTC1/SC22/WG14 - C and the US Technical Committee X3J11 to make available electronic copies of the WG21+X3J16 working draft and working documents to members of WG14 and X3J11 only. WG21+X3J16 directs its WG14 liaison to request that WG14+X3J11 permit WG21+X3J16 to distribute C documents.

Motion passed X3J16: 30 yes, 0 no, 0 abstain.
Motion passed WG21: 6 yes, 0 no, 1 abstain.

49) Motion (to submit the WP for a second CD Ballot) by Plum/Lajoie:

WG21+X3J16 agree that the WP, after changes agreed to at this meeting, incorporates all resolutions to national body comments submitted during CD ballot. WG21+X3J16 state their intention to forward the WP to SC22 for a second CD ballot after a satisfactory review of the WP at their next regularly scheduled meeting.

Ball expressed concern that the synopsis of the motion (in parentheses) was misleading. Anderson was concerned that there were still technical problems with the draft that we would need to fix in November. Harbison said it was hard to phrase this proposal more precisely, so he just asked people to go along with it. Plum said we all agree to stop outright invention, and we all agree to allow editorial correction. So the gray area is somewhere in between.

Motion passed X3J16: 28 yes, 0 no, 3 abstain.
Motion passed WG21: 7 yes, 0 no, 0 abstain.

The committees thanked:
-- Jonsson and Ericsson for the hosting meeting.
-- Harbison for his work as WG21 convener.
-- Rumsby for all his work this week, particular on drafting the motions.
-- Jim Dennert from SGI for help on the template proposal.
Applause.

11.2 Review action items, decisions made, and documents approved

11.3 Issues delayed until Friday

Gibbons explained that, at the last meeting, the committees approved Lajoie's rewrite of clause 14 with the understanding that the rewrite was purely editorial. That is, if the rewrite included any substantive changes appeared, they would be removed editorially. Gibbons reported that there was one such substantive change regarding syntax checking for

templates, and it will be edited out.

12 Plans for the future

12.1 Next meeting

No discussion.

12.2 Mailings

Lajoie said she must receive documents by July 26 for them to make the post-meeting mailing.

12.3 Following meetings

Harbison listed the dates and locations for the upcoming meetings:
-- 10-15 November 1996, Kona, Hawaii, USA hosted by Plum Hall
-- 9-14 March 1997, Nashua, New Hampshire, USA hosted by Digital
-- 13-18 July 1997, Guilford, UK, hosted by Programming Research
-- 9-14 November 1997 (location and host to be determined)
-- 8-13 March 1998, Sophia Anipolis, France hosted by AFNOR
-- 12-17 July 1998 (location and host to be determined)

13 Adjournment

Motion to adjourn passed WG21+X3J16: lots yes, 0 no.

The meeting adjourned at 12:10 on Friday.

Appendix A - Attendance

Name	Affiliation	M	Tu	W	Th	F
Dawes, Beman	.	V	V	V	V	V
Hesse, Joseph	.	A	A	A	A	A
Myers, Nathan	.	A	A	A	A	A
O'Riordan, Martin	.	A	A	A	A	A
Frennemo, Stefan	ABB Industrial Systems	A	A	A	A	A
Koenig, Andrew	AT&T Research	V	V	V	V	V
Stroustrup, Bjarne	AT&T Research	A	A	A	A	A
Becker, Pete	Borland	V	V	V	V	V
Tooke, Simon	Canada	A		A	A	A
Charney, Reg	Charney & Day	V		V	V	V
Whitman, Sandra	Digital	V	V	V	V	V
Simonsen, Keld	DKUUG			A	A	A
Bruck, Dag	Dynasim AB	V	V	V	V	V
Adamczyk, Steve	Edison Design Group	V	V	V	V	V
Anderson, Mike	Edison Design Group	A	A	A	A	A
Spicer, John	Edison Design Group	A	A	A	A	A
Jonsson, Fredrik	Ericsson	V	V	V	V	V
Gibbons, Bill	Hewlett Packard	V	V	V	V	V
Lajoie, Josee	IBM	V	V	V	V	V
Soop, Karl	IBM Sweden	A	A		A	
Nelson, Clark	Intel	V	V	V	V	V
Suto, Gyuszi	Intel	A	A	A	A	
Schwarz, Jerry	Intrinsa	A	A	A	A	A
Andersson, Per	Ipso Object Software	V	V	V	V	V
Stuessel, Marc	IST GmbH	V	V	V	V	V
Kamimura, Tsutomu	Japan	A	A	A	A	
Koshida, Ichiro	Japan	A	A	A	A	A
Umekawa, Ryuichi	Japan	A	A	A	A	A
Munch, Max	Lex Hack & Associates	A	A	A	A	A
Bruns, John	NationsBanc-CRT	V	V	V	V	V
Corfield, Sean	Object Consultancy Services	V	V	V	V	V

Benito, John	Perennial	V	V	V	V	V
Plum, Tom	Plum Hall	V	V	V	V	V
Southworth, Mark	Programming Research	V	V	V	V	V
Wilcox, Thomas R.	Rational Software	V	V	V	V	V
Glassborow, Francis	Richfords	V	V	V	V	V
Le Mouel, Philippe	Rogue Wave Software	A	A	A	A	V
Smithey, Randy	Rogue Wave Software	V	V	V	V	
Vandevoorde, David	RPI	A	A	A	A	A
Saks, Dan	Saks & Associates	V	V	V	V	V
Rouse, Jack	SAS Institute	V	V	V	V	V
Schilling, Jonathan	SCO	V	V	V	V	V
Wengler, Christian	SET Software Consulting	V	V	V	V	V
Hartinger, Roland	Siemens Nixdorf	V	V	V	V	V
Unruh, Erwin	Siemens Nixdorf	A	A	A	A	A
Austern, Matthew	Silicon Graphics	V	V	V	V	V
Wilkinson, John	Silicon Graphics	A	A	A	A	A
Miller, William M.	Software Emancipation Tech	V	V	V	V	V
Ball, Mike	Sun Microsystems	V	V	V	V	V
Clamage, Steve	Sun Microsystems	A	A	A	A	A
Harbison, Sam	Texas Instruments	A	A	A	A	A
Rumsby, Steve	UK	A	A	A	A	A
Welch, Jim	Watcom	V	V	V	V	V
Plauger, P. J.	WG14	A	A	A	A	
Crowfoot, Norm	Xerox	V	V	V	V	V
Total Voting Attendance		31	30	31	31	31
Total Attendance		54	52	54	55	50