C++ ABI Summary

Revised 3 June 1999

Meetings

When		Where	Phone	Reservation	Passcode
3 June	11:00-13:00	completed			
10 June	10:00-12:00	SGI	?	?	?
17 June	10:00-12:00	SGI	?	?	?

Participants

Company	Name	Telephone	Fax	Email
	overall reflector			cxx-abi@corp.sgi.com
SGI	Jim Dehnert	(650) 933-4272	(650) 932-4272	dehnert@sgi.com
	Matt Austern	(650) 933-4196	(650) 932-4196	austern@engr.sgi.com
	Shin-Ming Liu	(650) 933-4287	(650) 932-4287	shin@engr.sgi.com
	John Wilkinson	(650) 933-4298	(650) 932-4298	jfw@engr.sgi.com
	reflector			cxx-abi-sgi@engr.sgi.com
Cygnus	Jason Merrill	(408) 542-9665	(408) 542-9765	jason@cygnus.com
	Ian Carmichael	(416) 482-3946	(416) 482-6299	iancarm@cygnus.com
	Ulrich Drepper	(408) 765-4699	?	drepper@cygnus.com
	reflector			c++abi@cygnus.com
Hewlett- Packard	Cary Coutant	(408) 447-5759	?	cary@cup.hp.com
	Christophe de Dinechin	(408) 447-5491	?	ddd@cup.hp.com
	Sassan Hazeghi	(408) 447-5007	?	sassan@cup.hp.com
	reflector			cxx-abi-hp@cllmail.cup.hp.com
IBM	Mark Mendell	(416) 448-3485	?	mendell@ca.ibm.com
	Allan H. Kielstra	(416) 448-3558	?	kielstra@ca.ibm.com
Intel	Sunil Saxena	(408) 765-5272	(408) 653-8511	Sunil.Saxena@Intel.com
	Priti Shrivastav	(408) 765-4699	?	Priti.Shrivastav@Intel.com
	reflector			cxx-abi@unix-os.sc.intel.com
SCO	Jonathan Schilling	(908) 790-2364	(908) 790-2426	jls@sco.com
	George Vasick	(650) 786-5123	(650) 786-9551	george.vasick@eng.sun.com
	Michael Lam	(650) 786-3492	(650) 786-9551	michael.lam@eng.sun.com

Sun	Michael Ball	(650) 786-9109	(650) 786-9551	michael.ball@eng.sun.com
	Reza Monajjemi	(650) 786-6175	?	reza.monajjemi@eng.sun.com

Objectives

- Interoperable C++ compilation on IA-64: we want users to be able to build relocatable objects with different compilers and link them together, and if possible even to ship common DSOs. This objective implies agreement on:
    - Data representation
    - Object file representation
    - Library API
- ISO Standard C++: highest priority is functionality and performance of standard-compliant code. It should not be sacrificed for the benefit of language extensions or legacy implementations (though considering them as tie-breakers is fine).
- Some areas will be easier to agree on than others. Our priorities should be based on achieving as much interoperability as possible if we can't attain perfection. That is, it is better to end up with a few restrictions being required for interoperable code, than to have no interoperability at all. This suggests priorities as follows:
    1. Items requiring base ABI changes that might affect other languages, and will therefore become impossible soon. Examples include exception handling / stack unwind, or ELF changes (not extensions).
    2. Core features where differences will prevent virtually any C++ object code from porting. Examples include data layout and calling conventions.
    3. Limited usage features, where users can achieve portability by avoiding the feature. An example might be multi-threading.
    4. Peripheral features, where the requirements on users to achieve portability are clear and easy to implement. An example is non-explicit inlining, where compilers would presumably allow it to just be suppressed.
    5. Tool interfaces, which affect how users build code, rather than what they build. An example is the compilation command line.
- Mechanisms/methods which allow coexistence of incompatible implementations may be suitable in some cases. For instance, packaging vendor-specific compiler support runtimes in DSOs occupying distinct namespaces might allow multiple such DSOs to be loaded for mixed objects and avoid requiring that all vendors have the same support runtimes.

Action Item Status

#	Action	Who	Status	Opened	Closed
1	Distribute Sun C++ ABI	Mike Ball	open	990603	
2	Distribute Sun C++ ABI Rationale	Mike Ball	open	990603	
3	Distribute Taligent C++ ABI	Cary Coutant	open	990603	
4	Expedite IA-64 RT Arch doc release	Cary Coutant	open	990603	
5	Set up n-way NDA for eligible members	Priti Shrivastav	open	990603	
6	Organize/summarize object layout issues and alternatives	Matt Austern	open	990603	

Issue Status

In the following table, the ***class*** column attempts to classify the issue on the basis of what it likely affects. The identifiers used are:

call	Function call interface, i.e. call linkage
data	Data layout
lib	Runtime library support
lif	Library interface, i.e. API
g	Potential gABI impact
ps	Potential psABI impact
tools	May affect how program construction tools interact

#	Issue	Class	Status	Source	Opened	Closed
A	**Object Layout**					
A-1	Virtual function table	data	open	SGI	990520	
A-2	Virtual base classes	data	open	SGI	990520	
A-3	Multiple inheritance	data	open	SGI	990520	
A-4	Empty base classes	data	open	SGI	990520	
A-5	Empty parameters	data	open	SGI	990520	
A-6	RTTI (`type_info`) .o representation	data call ps	open	SGI	990520	
A-7	Vptr sharing with primary base class	data	open	HP	990603	
A-8	(Virtual) base class alignment	data	open	HP	990603	
A-9	Sorting fields as allowed by [class.mem]/12	data	open	HP	990603	
A-10	Parameter struct field promotion	call	open	HP	990603	
A-11	Representation of pointers to members	data	open	Sun	990603	
B	**Virtual Function Handling**					
B-1	Adjustment of "this" pointer (e.g. thunks)	data call	open	SGI	990520	
B-2	Covariant return types	call	open	SGI	990520	
B-3	Optimizing for unchanged vptr	call	open	HP	990603	
B-4	Function descriptors in vtable	data	open	HP	990603	
B-5	Where are vtables emitted?	data	open	HP	990603	
C	**Object Construction/Destruction**					
C-1	Interaction with .init/.fini	lif ps	open	SGI	990520	
C-2	Order of const/destr w.r.t. link	lif ps	open	HP	990603	
C-3	Order of const/destr w.r.t. DSOs	ps	open	HP	990603	
C-4	Calling vfuncs in constr/destr	call	open	Sun	990603	
C-5	Calling virtual destructors	call	open	Sun	990603	
C-6	Extra parameters to constr/destr	call	open	Cygnus	990603	
C-7	Extra parameters to constr/destr	call	open	Cygnus	990603	
D	**Exception Handling**					
D-1	Language-specific data area format	lib ps	open	SGI	990520	
D-2	Unwind personality routines	lib ps	open	SGI	990520	
D-3	Unwind process clarification	lib ps	open	SGI	990520	
D-4	Unwind routines nested?	lib ps	open	SGI	990520	
D-5	Interaction with other languages (e.g. Java)	lib ps	open	HP	990603	
D-6	Allow resumption in other languages?	lib ps	open	HP	990603	
D-7	Interaction with signals or asynch events	lib ps	open	HP	990603	

D-7	Interaction with threads packages	lib ps	open	SGI	990603
E	**Template Instantiation Model**				
E-1	When does instantiation occur?	tools	open	SGI	990520
E-2	Separate compilation model	tools	open	SGI	990520
E-3	Template repository	tools	open	HP	990603
F	**Name Mangling**				
F-1	Mangling convention	call	open	SGI	990520
F-2	Mangled name size	call g	open	SGI	990520
F-3	Distinguish template instantiation and specialization	call g	open	SGI	990520
G	**Miscellaneous**				
G-1	Basic command line options	tools	open	HP	990603
G-2	Detection of 1-def rule violations	call	open	Sun	990603
G-3	Inlined routine linkage	call	open	Sun	990603

Notes from 3 June 1999

- Introductions
- Objectives: see above
- Procedure
  - Meetings: 10-12 Thursdays at SGI for the near term.
  - Intel NDA: Generally unnecessary. Priti will set up n-way for eligible members for cases where needed. Cary expects RT architecture/software conventions document to be released in the next month or two, removing most of the issues.
  - Communication: Use of reflector encouraged for discussion. NDA communication will be handled with password-protected PDF once Intel sets up n-way.
  - Available documents: Parties with existing, relevant documents (includes Sun, HP) will send them to group.
  - Intellectual property: Participants don't expect problems with release of any of their IP. Microsoft has extensive patents in the area, but they are excessively broad (covering obvious ideas and prior art), so expectation is that they are not a problem. Nonetheless, we should be aware of them.
- Issue Identification: new issues reflected in status table.

Please send corrections to Jim Dehnert.