

The GNU Transactional Memory Library

Copyright © 2011-2025 Free Software Foundation, Inc.

Permission is granted to copy, distribute and/or modify this document under the terms of the GNU Free Documentation License, Version 1.2 or any later version published by the Free Software Foundation; with no Invariant Sections, no Front-Cover Texts, and no Back-Cover Texts. A copy of the license is included in the section entitled “GNU Free Documentation License”.

Short Contents

1	Enabling libitm	1
2	C/C++ Language Constructs for TM	3
3	The libitm ABI	5
4	Internals	11
	GNU Free Documentation License	17
	Library Index	25

3.5.18	[New] Transactional indirect calls	9
3.5.19	[New] Transactional dynamic memory management.....	9
3.6	[No changes] Future Enhancements to the ABI.....	10
3.7	Sample code.....	10
3.8	[New] Memory model.....	10
4	Internals	11
4.1	TM methods and method groups	11
4.1.1	TM method life cycle.....	11
4.1.2	Selecting the default method.....	11
4.2	Nesting: flat vs. closed	11
4.3	Locking conventions	12
4.3.1	State-to-lock mapping	12
4.3.2	Lock acquisition order	13
4.3.3	Serial lock implementation.....	14
4.3.4	Reentrancy	14
4.3.5	Privatization safety	15
4.3.6	Progress guarantees.....	15
	GNU Free Documentation License	17
	ADDENDUM: How to use this License for your documents.....	24
	Library Index	25

1 Enabling libitm

To activate support for TM in C/C++, the compile-time flag `-fgnu-tm` must be specified. This enables TM language-level constructs such as transaction statements (e.g., `--transaction_atomic`, see Chapter 2 [C/C++ Language Constructs for TM], page 3, for details).

2 C/C++ Language Constructs for TM

Transactions are supported in C++ and C in the form of transaction statements, transaction expressions, and function transactions. In the following example, both **a** and **b** will be read and the difference will be written to **c**, all atomically and isolated from other transactions:

```
__transaction_atomic { c = a - b; }
```

Therefore, another thread can use the following code to concurrently update **b** without ever causing **c** to hold a negative value (and without having to use other synchronization constructs such as locks or C++11 atomics):

```
__transaction_atomic { if (a > b) b++; }
```

GCC follows the Draft Specification of Transactional Language Constructs for C++ (v1.1) (<https://sites.google.com/site/tmforcplusplus/>) in its implementation of transactions.

The precise semantics of transactions are defined in terms of the C++11/C11 memory model (see the specification). Roughly, transactions provide synchronization guarantees that are similar to what would be guaranteed when using a single global lock as a guard for all transactions. Note that like other synchronization constructs in C/C++, transactions rely on a data-race-free program (e.g., a nontransactional write that is concurrent with a transactional read to the same memory location is a data race).

libitm also provides transactional clones of C++ memory management functions such as global operator new and delete. They are part of libitm for historic reasons but do not need to be part of this ABI.

3.6 [No changes] Future Enhancements to the ABI

3.7 Sample code

The code examples might not be correct w.r.t. the current version of the ABI, especially everything related to exception handling.

3.8 [New] Memory model

The ABI should define a memory model and the ordering that is guaranteed for data transfers and commit/undo actions, or at least refer to another memory model that needs to be preserved. Without that, the compiler cannot ensure the memory model specified on the level of the programming language (e.g., by the C++ TM specification).

For example, if a transactional load is ordered before another load/store, then the TM runtime must also ensure this ordering when accessing shared state. If not, this might break the kind of publication safety used in the C++ TM specification. Likewise, the TM runtime must ensure privatization safety.

Because enforcing stronger progress guarantees in the TM has a higher runtime overhead, we focus on deadlock-freedom right now and assume that the threads will get scheduled eventually by the OS (but don't consider threads with different priorities). We should support starvation-freedom for serial transactions in the future. Everything beyond that is highly related to proper contention management across all of the TM (including with TM method to choose), and is future work.

TODO Handling thread priorities: We want to avoid priority inversion but it's unclear how often that actually matters in practice. Workloads that have threads with different priorities will likely also require lower latency or higher throughput for high-priority threads. Therefore, it probably makes not that much sense (except for eventual progress guarantees) to use priority inheritance until the TM has priority-aware contention management.

ADDENDUM: How to use this License for your documents

To use this License in a document you have written, include a copy of the License in the document and put the following copyright and license notices just after the title page:

```
Copyright (C)  year  your name.
Permission is granted to copy, distribute and/or modify this document
under the terms of the GNU Free Documentation License, Version 1.3
or any later version published by the Free Software Foundation;
with no Invariant Sections, no Front-Cover Texts, and no Back-Cover
Texts. A copy of the license is included in the section entitled ``GNU
Free Documentation License''.
```

If you have Invariant Sections, Front-Cover Texts and Back-Cover Texts, replace the “with...Texts.” line with this:

```
with the Invariant Sections being list their titles, with
the Front-Cover Texts being list, and with the Back-Cover Texts
being list.
```

If you have Invariant Sections without Cover Texts, or some other combination of the three, merge those two alternatives to suit the situation.

If your document contains nontrivial examples of program code, we recommend releasing these examples in parallel under your choice of free software license, such as the GNU General Public License, to permit their use in free software.

Library Index

FDL, GNU Free Documentation License..... 17