The NCBI C++ Toolkit Book
The NCBI C++ Toolkit Book

The NCBI C++ Toolkit

Release Notes (April 30, 2006)

Created: April 30, 2006.
Last Update: May 2, 2006.

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

Download Location

ftp://ftp.ncbi.nih.gov/toolbox/ncbi_tools++/2006/Apr_30_2006/

Source Archive Contents

Source Code Archives

- ncbi_cxx--Apr_30_2006.tar.gz — for UNIX'es (see the list of UNIX flavors below) and MacOSX/GCC
- ncbi_cxx--Apr_30_2006.gtar.gz for UNIX'es (see the list of UNIX flavors below) and MacOSX/GCC
- ncbi_cxx--Apr_30_2006.exe — for MS-Windows (32- and 64-bit) / MSVC++ (7.1, 8.0) — self-extracting
- ncbi_cxx--Apr_30_2006.zip — for MS-Windows (32- and 64-bit) / MSVC++ (7.1, 8.0)
- ncbi_cxx_mac_xcode--Apr_30_2006.gtar.gz — for MacOSX 10.4 / Xcode 1.5-2.2.1

The sources correspond to the NCBI production tree sources from patch "CATCHUP_APR_2006", which in turn roughly corresponds to the development tree sources from the very beginning of April, 2006.

There are also two sub-directories, containing easily buildable source distributives of the NCBI C Toolkit (for MS Windows and UNIX) and selected 3rd-party packages (for MS Windows only). These are the versions that the NCBI C++ Toolkit should build with. For build instructions, see README files there:

- NCBI_C_Toolkit
- ThirdParty

New Development

PLATFORMS AND CONFIGURATIONS

A Newly (re)supported:

1 MS Visual Studio 2005 (both 32- and 64-bit).

2 Mac OS X 10.4 (Tiger).

3 GCC/Cygwin. **NOTE:** As of now, it supports static builds only, DLLs are not buildable.

4 GCC with GNU ld on Solaris 10 (as long as the patch from http://sourceware.org/bugzilla/show_bug.cgi?id=1031 has been applied)

B Discontinued:

1 CodeWarrior on MacOSX

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

CORELIB — Portability and Application Framework

1. CDirEntry::SplitPathEx() — new function, to split a path string into its basic components (not OS-specific version).
2. CDir::SetCwd() — new function, to change the current working directory.
3. CFileUtil::GetFileSystemInfo() — new function.
4. CMemoryFile — new method.added constructor parameters for automatic creating/ extention of the mapped file. Added new Extend() method.
5. NStr::TStringToNumFlags — replaced by the integer-based counterparts. Replace NStr::fStringToNumDefault by 0 if you have compile errors.
6. NStr::StringToDouble() — added TStringToNumFlags flags support.
7. CStopWatch::Stop() — new function, to suspend a "timer".
8. SetCpuTimeLimit() — added new parameter - maximum process termination time (default 5 seconds)
9. CProcess::Kill() — added linger_timeout parameter to more control of process termination.
10. Sleep*() — try to utilize unslept part of the time if interrupted by a signal.
11. CExec::RunSilent() — new function to run console applications on MS Windows without terminal window. On UNIX, it is equivalent to SpawnL().
12. IBlobStorage:
    - changed its name from bormer INetScheduleStorage
    - moved to the corelib (from connect/services)
    - got equipped with a class factory CBlobStorageFactory
13. CSafeStatic[Ptr|Ref] — added life span parameter for safe static objects. By default protect only initialization of an object, but do not control its lifetime.
14. Added groups and error handler for application arguments.
15. IReader/IWriter abstraction as well as stream implementation on top of IReader/ IWriter-based stream buffer moved from UTIL to here.
16. CPluginManager — new methods, to tune the DLL search paths: ResetDllSearchPath (), SetDllStdSearchPath() and GetDllStdSearchPath().
17. Macros NCBI_RESTRICT and NCBI_FORCEINLINE — now available on all platforms.

CONNECT — Data streaming, Networking, and Dispatching

1. CPipe::Poll() — new function to check for the immediate availability of data in the pipe.
2. SOCK_HostPortToString() and SOCK_StringToHostPort() — new functions equivalent to the now obsolete HostPortToString() and StringToHostPort().
3. CONNUTIL_GetUsername() and ConnNetInfo_SetupStandardArgs() — new.

UTIL — Miscellaneous Low-Level APIs

1. CCache — new class template, implements a generic cache of objects..
2. CRegexp — added new methods Escape(), IsMatch().
3. CBlockingQueue<>, CPoolOfThreads<> — reworked to support increased user control (via handles) and optionally run callbacks upon status changes.

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

4. CBlockingQueue<>::Put, CPoolOfThreads<>::[x_]Accept[Urgent]Request — Accept an optional timeout in which space may become available, defaulting to zero for consistency with existing behavior.

SERIAL — Data Serialization (ASN.1, XML)

1. Added support of BIT STRING type data in serialization streams.
2. Added serial stream manipulators with parameters: MSerial_Format, MSerial_VerifyData, MSerialXml_DefaultStringEncoding.
3. Added input stream object iterators: CIStreamObjectIterator and CIStreamStdIterator
4. Added an option to write 64 bits integer data in ASNTOOL-compatible way.
5. ASN.1-based-object converter (for integrating C Toolkit code) — accommodate the C Toolkit's nonstandard treatment of Int8 fields, and allow text mode as an option in case other binary-representation issues somehow crop up.
6. RPC clients (for ID1, Entrez2, etc.) automatically retry in a wider range of circumstances.

CGI — CGI and Fast-CGI Application Framework

1. CCgiSession — new class, to pass user data between CGI calls. The default storage for the data is uses NetCache, but that can be easily changed by implementing ICgiSessionStorage interface in any other way.
2. CCgiException — added HTTP status code and message.
   CCgiApplication — generates HTTP status header if it is set in CCgiException.

HTML — HTML Generation Library

1. CHTMLPage — added possibility to cache template files and template libraries.

BDB — Yet Another C++ API Based On BerkeleyDB

1. CBDB_Env — added method to control reverse splitting of pages (RevSplitOff()).

Local data storage

1. Added read-only open mode.

DBAPI — SQL Database Connectivity

1. IConnValidator — new interface, to detect the status of database connection.
2. CTrivialConnValidator and CConnValidatorCoR implementations of IConnValidator.
3. I_DeriverContext::ConnectValidated() — new method, to check for the overall validity of the database connection after establishing the connection.
4. Close() — method added to all context/command-aware classes.
5. Treat database messages with severity == 10 && msgnumber == 0 as having informational severity. Replaced fatal severity of SQL Server error messages with error severity.
6. Added new sample application dbapi_conn_policy.

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

ALGO/ALIGN/SPLIGN — Spliced Alignment Algorithms

1. CSplign -
   Min singleton identity parameter re-introduced to control the identity cut-off for compartments unique per subject per strand. Parts of the code have been optimized including data loading and core dynamic programming
2. CHitFilter -
   Maximal Greedy Reconciliation algorithm have been introduced to enforce non-redundancy of pairwise alignments across multiple sequences.

ALNMGR — Bio-sequence Alignment Manager

1. CAlnMix::fRemoveLeadTrailGaps — new flag, to remove end gaps.

BLAST

1. * Variable word size feature has been removed.
2. * Removed (no longer needed) per-search-type constructors for CRemoteBlast.
3. Optimizations to lookup table width choice.
4. Introduced new varieties of composition based statistics.
5. Enable the use of composition based statistics for tblastn.
6. Added interruptible API to CBl2Seq.
7. * Add support for user-specified query masked locations in CRemoteBlast.
8. Optimizations to RPS-BLAST word finder.
9. Set number of subjects per database chunk adaptively.
10. Changed how the B, Z, X and U residues are scored within the composition adjustment library. Most significantly, the method of computing scores involving X has changed. The score of aligning X to a character is the expected score of aligning that character to any true amino acid, subject to the restriction that the score be at most -1. The score of aligning X to itself is the expected score of aligning any two true amino acids, again subject to the restriction that the score be at most -1. U is treated as equivalent to X. The score of aligning a true amino acid to the rare B or Z characters is computed by a log odds formulation that uses the sum of the target or background frequencies of the two characters each ambiguity character represents as the target or background frequency of the ambiguity character.

BIO-OBJECTS — Bio-Object Specific Utility Functions (Not Involving OM++)

1. * CSeq_id — ParseFastaIds() treats untagged numeric IDs as local IDs rather than GIs, which must be tagged gi|... in this context.
2. * CSeq_id — use (and accept) the FASTA ID tag pgp|... for "pre-grant patents" (applications).
3. CSeq_id — disregard extra trailing vertical bars in FASTA-style IDs.
4. CSeq_id — IdentifyAccession() supports several new prefixes (DX through EB).
5. Support the new amino acid codes J/Xle and O/Pyl.

OM++ — Object Manager — For Retrieving and Processing Bio-Objects

1. CSeqLocMapper — if intervals get truncated while mapping, it is now idicated by setting fuzz to lim tl/tr.

2 CSeqLocMapper — added flag to check strands before mapping (allows to filter out locations which are not on the source strand).

3 Added methods to collect only the annotations of selected types rather than all annotation objects.

OM++ LOADERS/READERS — Data Retrieval Libraries for OM++

1 Modified Phrap reader to allow aligned segment to be shorter than base segment, skip WR, CT and RT tags in some cases. Generate warnings when skipping tags.

OM++ DEMO program (objmgr_demo)

1 Added options to use LDS and BLAST data loaders.

BUILD FRAMEWORK (UNIX)

1 * The minimum supported Berkeley DB version is now 4.3. (4.4 is also fully supported.)

2 Configure now supports —with-distcc, which behaves similarly to —with-ccache apart from being off by default.

3 compilers/GCC.sh now supports looking for a specific GCC version, which the user can supply as an optional first argument.

4 Configure now supports a —with-gbench option that ensures correct compilation options (—with-mt —with-dll) and errors out if any of the third-party libraries Genome Workbench requires are unavailable.

PTB — Project Tree Builder for MSVC++ .NET

1 Added DATASPEC-ALL project, which includes all Toolkit projects with automatically generated sources.

APPLICATIONS

1 NetCache:

- Server-side: added session management, such as automatic shutdown when the last client is logged off.
- Client-side: added an implementation of ICache interface with NetCache.

2 NetSchedule:

- Optimization of use of BDB in order to reduce number of collisions and deadlocks when running transactions
- Reduced memory consumption
- Changed queue truncation algorithm not to overflow in-memory transaction log
- Fixed scheduling bug related to job affinity
- Added client registration command

GRID (DISTRIBUTED COMPUTING) FRAMEWORK

CONNECT/SERVICES — Components for the Network Grid Framework

1 CNetScheduleNSSStorage_NetCache — renamed to CBlobStorage_NetCache and moved to a separate library.

1. Idle task facility — to allow worker node to perform some task when it is not executing any "real" jobs.
2. "remote_app" worker node — to launch external applications or scripts via network using NetSchedule/NetCache services.
3. "max_failed_jobs" worker node configuration parameter — to specify the maximum number of failed jobs after which the worker node will shutdown itself.
4. "use_embedded_storage" configuration parameter for the worker node (and its client too) — to specify if the worker node should try to use a NetSchedule's internal storage instead of NetCache blobs for passing its input/output data. This reduces the network load and increases the data exchange.
5. "reuse_job_object" worker node configuration parameter — to have only one instance of the IWorkerNodeJob interface per job's thread (instead of creating a new object for each job).
6. "use_permanent_connection" worker node configuration parameter — to keep a permanent connection to NetSchedule server (rather than to establish a new connection for each job exchange).

Documentation

Document Location

The documentation is available online at as a book titled "The NCBI C++ Toolkit". This is an online searchable book.

The C++ Toolkit book also provides PDF version of the chapters; although these are not up to date in this release. The PDF version can be accessed by a link that appears on each page.

The older HTML documentation has been deprecated and is no longer being updated, and "The NCBI C++ Toolkit" online book at the previously listed URLs is the official documentation.

Document Content

Documentation has been grouped into chapters and sections that provide a more logical coherence and flow. New sections and paragraphs continue to be added to update and clarify the older documentation or provide new documentation. The chapter titled "Introduction to the C++ Toolkit" gives an overview of the C++ Toolkit. This chapter contains links to other chapters containing more details on a specific topic and is a good starting point for the new comer.

The DOXYGEN source browser is used to complement the traditional docs with structured DOXYGEN-generated "Library Reference" ones based on the in-source comments. You can access the DOXYGEN source browser for the NCBI code from:

http://www.ncbi.nlm.nih.gov/IEB/ToolBox/CPP_DOC/doxyhtml/

The above link is also available under the "Browsers" that appears on each page.

You can also access the CVS code repository via a Web interface. These links also appear in the sidebar box on each page.

A C/C++ Symbol Search query appears on each page of the online Toolkit documentation. You can use this to perform a symbol search on the public or in-house versions of LXR,

The NCBI C++ Toolkit Book
The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

Doxygen and Library. The Library search runs a CGI script that lists the library name where the symbol (such as function) is defined in.

Building on the MacOS

We now build all the libraries and most of the applications including the Genome Workbench (gbench), some test and a few demo applications. We also build the FLTK library's GUI editor, fluid.

All apps are built as application bundles except gbench_plugin_scan and datatool which are built as command line apps.

GCC

Uses a regular Unix build pattern: run configure, then make.

Xcode

When building the toolkit with Xcode, the latest version of Xcode (at least 1.5) is required for the trouble-free build. Build procedure is as follows: open, build and run a project file in compilers/xCode. This is a GUI tool to generate a new NCBI C++ Toolkit Xcode project. You'll have an option to specify third-party installation directories and choose which packages (libs, applications and tests) to include into the final project. The option to automatically download and install all the third-party libraries is also available.

Xcode build fully supports all the latest Apple innovations: distributed builds, optional CPU specific optimization, pre-compiled headers, fix & continue, code cleanup and Zero Link (with few exceptions).

Xcode build has a Shell Script build phase for each Target dependent on generated ASN files. These shell scripts use datatool to regenerate source files each time the ASN specification files do change.

Xcode builds all libraries as a Mach-O dynamically linked shared ones (.dylib) and all Genome Workbench plugins as Mach-O bundles (also .dylib extension). Note, that Xcode will place Genome Workbench plugins inside Genome Workbench application bundle (Genome Workbench.app/Contents/MacOS/plugins).

Platforms (OS's, compilers used inside NCBI)

This release was successfully tested on at least the following platforms — but may also work on other platforms. Since the previous release, some platforms were dropped from this list, just because we do not use them here anymore, and some were added (these new platforms are highlighted using bold font). Also, it can happen that some projects would not work (or even compile) in the absence of 3rd-party packages, or with older or newer versions of such packages — in these cases, just skipping such projects (e.g. using flag "-k" for make on UNIX), can get you through.

Table 2. Unix OS's and Supported Compilers

Operating System	Architecture	Compilers
Linux-2.**6.x** (w/ LIBC 2.3.2)	x86-32	GCC 3.4.0 ICC 8.0 (build 20040520Z, package ID l_cc_pc_8.0.066_pe067.1) (GCC 3.0.4, 2.95.3, **4.1.0** - nominal support)
Linux-2.6.x (w/ LIBC 2.3.3)	x86-64	GCC 4.0.1 ICC 9.0 (build 20051201)
Solaris-8	SPARC	C++ 5.3 (WorkShop 6u2) patch 111685-23 (64-, 32-bit) (GCC 3.4.3 - nominal support)
Solaris-10	SPARC	Sun C++ 5.5 (Studio8) patch 113817-15 GCC 4.0.1 (32-bit mode only)
Solaris-9	x86-32	C++ 5.3 (WorkShop 6u2) patch 111686-13 GCC 3.4.3
IRIX64-6.5	SGI-Mips	MIPSpro 7.3.1.3m (64-bit, 32-bit)
FreeBSD-4.10	x86-32	GCC 3.4.2
Digital Tru64 Unix 5.1 (aka OSF1)	ALPHA	GCC 3.3.2

Table 3. MS Windows and Supported Compilers

Operating System	Compilers
MS Windows-32	MS Visual Studio .NET 2003 (C++ 7.1). See documentation for building the Toolkit with MS Visual C++ .NET NOTE: We also ship an easily buildable archive of 3rd-party packages (including NCBI C Toolkit) for this platform.
MS Windows-32	**MS Visual Studio .NET 2005 (C++ 8.0)**
MS Windows-64	**MS Visual Studio .NET 2005 (C++ 8.0)**
Cygwin 1.5.18 on MS Windows-32	**GCC 3.4.4**

Table 4. Mac OS X, and Supported Compilers

Operating System	Compilers
Darwin on MacOS X 10.4 ("Tiger")	GCC **4.0.1**
Darwin on MacOS X 10.4 ("Tiger")	Xcode 1.5 - 2.2.1

Caveats and Hints

GCC 2.95

1. Poor MT-safety record.
2. Relatively incomplete/incorrect (comparing to modern compilers) STL implementation.
3. It is going to be deprecated in NCBI as soon as we have any significant trouble with its maintenance.

GCC 3.0.4

1. Destructor of constructed class member is not called when exception is thrown from a method called from class constructor body (fixed in GCC 3.3).

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

2. STL stream uses locale in thread unsafe way which may result to segmentation fault when run in multithread mode (fixed in GCC 3.3).
3. Long-file support for C++ streams is disabled/broken (first broken in 3.0, fixed in 3.4).

GCC 3.3

Other than the feature described below, GCC 3.3.2 had been very good to us; it had a lot of very ugly bugs finally fixed.

1. Painfully slow linking in debug mode on Linux with GCC-3.3 compiler. — Starting with binutils 2.12 linker tries to merge constant/debug strings marked for merging in object files. But it seems it does this job very inefficiently - I've seen messages about it in internet. GCC starting with version 3.2 marks section of string constants ready for merging, and also has an option to disable this flag in object files (-fno-merge-constants). Adding this flag to compilation stage allows to avoid slow linking. GCC 3.3 also sets merge flag for debug sections and unfortunately there is no option to disable this flag. As a result, linking of debug executables significatly slower than with gcc 3.0.4. The slowdown rate depends on size of debug strings section and it's non-linear, so bigger projects will suffer more of this bug (N^2). Binutils 2.15 fixes this. The link time still 2 times slower than without symbol merge, but the resultant executable is about two times smaller in size, and no compiler patching is necessary. We are still testing it in-house. We had to patch GCC 3.3 in-house with the fix described at http://lists.boost.org/MailArchives/boost/msg53004.php.
2. Long-file support still broken.

GCC 3.4.x, 4.0.x

1. The "Painfully slow linking..." (see GCC3.3, [1] above) was an issue, and we had to patch it in-house to speed up, a la GCC 3.3 — until we finally upgraded to binutils 2.15.
2. At least on Linux, ifstream::readsome() does not always work for large files, as it calls an ioctl that doesn't work properly for large files (we didn't test whether 4.0.x fixed this).
3. At least on Linux, GCC 3.4.[0,1] optimizer (very rarely) generates incorrect code when comparing enumerated values in else-ifs. (Fixed in 3.4.2)
4. GCC 3.4.3, 3.4.4 and 4.0 have a bug in the C++ stream library that affects some parts of our code, notably CGI framework. (Fixed in 4.0.1).

Last Updated

This section last updated on May 5, 2006.