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

The NCBI C++ Toolkit

Release Notes (December 31, 2005)

Created: December 31, 2005.
Last Update: February 1, 2006.

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

Download Location

ftp://ftp.ncbi.nih.gov/toolbox/ncbi_tools++/2005/Dec_31_2005/

Source Archive Contents

Source Code Archives

- ncbi_cxx--Dec_31_2005.tar.gz — for UNIX'es (see the list of UNIX flavors below) and MacOSX/GCC
- ncbi_cxx--Dec_31_2005.gtar.gz — for UNIX'es (see the list of UNIX flavors below) and MacOSX/GCC
- ncbi_cxx--Dec_31_2005.exe — for MS-Windows / MSVC++ 7.1 (self-extracting)
- ncbi_cxx--Dec_31_2005.zip — for MS-Windows / MSVC++ 7.1
- ncbi_cxx_mac_cw--Dec_31_2005.tgz — for MacOSX 10.4 / CodeWarrior DevStudio for MacOS 9.5
- ncbi_cxx_mac_xcode--Dec_31_2005.gtar.gz — for MacOSX 10.4 / Xcode 1.5-2.2.1

The sources correspond to the NCBI production tree sources from patch "CATCHUP_DEC_2005", which in turn roughly corresponds to the development tree sources from the very beginning of December, 2005.

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

CORELIB — Portability and Application Framework

A. File system

1. CDirEntry — added EGetEntriesFlags type and 'flag' versions of GetEntries[Ptr]() methods. Marked all 'enum' versions as obsolete.
2. CDirEntry::GetObjectType() implemented for all CDirEntry based classes.
3. Added class CFileUtil, now with GetDiskSpace() and GetFreeDiskSpace() methods.

B. Strings

1. NStr::JavaScriptEncode() — new method.
2. NStr::StringToXxx() — the obsolete versions (ones using 'enum') are removed for good.
3. NStr::XxxToString() — added new radix base parameter.
4. CStringUTF8 class — enhanced to also support ISO8859-1 (Latin1) and Windows CP-1252 character encodings, and to guess the encoding of a text.

C. Time

1. CTime — now using bit fields and fixed-size types for class members, to save memory.
2. CStopWatch —added new constructor CStopWatch(EStart). CStopWatch(bool) declared as deprecated.
3. CFastLocalTime::GetLocalTimezone() — new method, to obtain the difference between UTC and current local time.

D. Application parametrization

1. CParam<> — new class to handle application run-time parameters.
2. CTreeNode — redesigned to support both simple values and key-value pairs, added search methods, removed CPairTreeNode.
3. Configuration file syntax now permits the use of CR, LF and TAB separators in .Include, .SubNode etc. values.

E. Miscellaneous

1. Functions to convert "streampos" to/from "Int8", where possible.
2. ArraySize — template (sometimes a macro) to get a number of elements in a static or stack array.

CONNECT — Data streaming, Networking, and Dispatching

1. HTTP connectors have got "flushable" feature (controlled by constructor's fHCC_Flushable flag): "flushable" connectors allow forced CONN_Flush() to actually flush the accumulated contents down to server (previous implementation defined this as no operation, until an explicit READ or WAIT-ON-READ conditions). This flag has also been added to SERVICE connectors (to access the cases when HTTP is the underlying implementation).
2. CRC32 (pure C version) added; declared in <connect/ncbi_connutil.h>.
3. CSocket::ReadLine() — new method that mimics behavior of its C counterpart, and is aimed to speed up reading by reducing the number of reallocations when filling up a string.

UTIL — Miscellaneous Low-Level APIs

1. Implemented ZIP-style CRC32.
2. Ensure that functions to compute erf() and erfc() are available, using system-supplied implementations where possible.

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

SERIAL — Data Serialization (ASN.1, XML)

1. Enhanced character encoding support in the XML serialization.
2. Corrected stream position calculations to support large (greater than 2GB) files on the capable platforms.
3. Reviewed and classified fail flags in serial object streams to allow for a more detailed reporting of an input stream state after the read operations. In particular, it's now easier to tell whether EOF occured in the middle of an object or between the objects.
4. Added support of ASN.1 BIT STRING data type.
5. CClassTypeInfoBase::GetRegistered{Module,Class}Names — new methods, to get the list of registered module and class names.
6. Implemented serialization of set and map with custom comparator.

CGI — CGI and Fast-CGI Application Framework

1. CCgiUserAgent — new class, to parse user agent strings.
2. CCgiArgs_Parser, CCgiArgs — new classes to handle (parse, hold and compose) CGI arguments.
3. CUrl — new class, to handle URLs.

HTML — HTML Generation Library

1. CHTMLHelper — added HTMLJavaScriptEncode() method.
2. CHTMLOpenElement — added parsing mapping tags <@...@> for attributes.
3. CHTML_Table::GetCurrent[Row|Col] return 0 instead of -1 if current row/column in the table is not defined. By default, the current row/col in the table always is (0,0).

BerkeleyDB API (bdb) — Much Enriched C++ API Based On BerkeleyDB

1. Implemeneted the reading of BLOBs in cursors.
2. BDB Environment - added methods to customize locking. Added option to place transaction logs in memory.
3. Added a bit vector storage file.

DBAPI — SQL Database Connectivity

1. DBAPI driver library names have changed from 'dbapi_driver_*' to 'ncbi_xdbapi_'.
2. CDB_DateTime, CDB_SmallDateTime — now 'null' if initializated with empty CTime.
3. Handle SYBUNIQUE data type as eDB_VarBinary in case of the FTDS driver.
4. I_DriverContext - added methods Set/GetApplicationName, Set/GetHostName.

PYTHON database module based on DBAPI

1. Now use TDS protocol version 12.5 when connecting to Sybase servers.
2. Fixed the reading of LOB (Text/Image) data types.

ALGO/ALIGN/SPLIGN — Spliced Alignment Algorithms

1. Splign code — now uses Object Manager data loaders and LDS to load and look up sequence data. Handling of sequence data has been optimized which led to a dramatic improvement in the overall batch performance.

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

2. CSplign::SetMaxGenomicExtent()— new method, to adjust the maximum span of genomic sequence beyond the hit boundaries where SPLIGN should look for terminal exons.
3. Compartmentization code has been moved to the 'xalgoalignutil' library. See CCompartmentFinder interface for details.
4. SPLIGN application — now supports traditional pairwise text output.

ALNMGR — Bio-sequence Alignment Manager

1. CSeq-align:
   - CreateDensegFromStdseg now copies the Seq-align's scores too.
2. CAlnMixMerger interface:
   - Derive from CTaskProgressReporter
   - Delegated truncation to CDiagRangeCollection.
   - Added progress feedback.
3. CAlnMix interface:
   - Derive from CTaskProgressReporter
   - Added progress feedback.
4. CAlnMrgApp:
   - Added optional progress feedback.
5. CDiagRangeCollection:
   - Implemented initial version of a diagonal alignment range collection.

BLAST

1. Added search classes to perform preliminary and traceback stages of the BLAST search separately.
2. Added location-transparent, uniform search interface with support for limited functionality (algo/blast/api/uniform_search.hpp).
3. Renamed headers for PSSM engine (algo/blast/api/blast_psi.hpp -> algo/blast/api/pssm_engine.hpp)
4. Added query sequence retrieval interface to allow sequence data retrieval with and without the NCBI C++ Object Manager.
5. Introduced usage of precomputed statistical parameters for blastn:
   - Added support for blastn reward/penalty values of 1/-5, 3/-4, and 3/-2.
   - Added adjustment of odd blastn scores when match reward = 2
6. Simplified and made organization of filtering locations data (BlastMaskLoc::seqloc_array) more consistent throughout the code.
7. Gapped alignment changes in CORE BLAST:
   - Remove ability to decline alignments
   - Streamlined memory usage in core dynamic programming routines.
   - Capped at 1000 the number of diagonals examined by the greedy aligner.
8. Changed convention for unset gap parameters — from zero to negative number.
9. CPsiBl2Seq — new class, to perform PSSM to protein sequence comparisons.

10 New library dependencies:

- xblast now depends on composition_adjustment
- xblastformat now depends on xblast

11 Created a new library in algo/blast/composition_adjustment. This library contains routines for adjusting the scoring system of blastp and tblastn searches to reflect the composition of the sequences being aligned. The composition_adjustment library is used in the blastpgp executable compiled from the C toolkit. The library is included because it will be used in the future to provide composition adjustment to BLAST executables compiled from the C++ Toolkit. One goal is to implement as much of this functionality as possible in a library that is shared between the C and C++ Toolkits.

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

1 CSeq_id — allow (but do not require) FASTA-style dbSNP IDs to have "extra" vertical bars per historical practice.

2 CSeq_id — support new prefix "DV" in IdentifyAccession() method.

BIO-TOOLS

1 CFastaOstream — improve output when instantiating gaps, and streamline internally.

LDS - Local data storage

1 Library lds_admin merged with lds.lib

2 Improved indexing of large files. Use 64-bit file offsets.

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

1 Implemented new feature editing API.

2 Added CBioseq_Handle::IsProtein() & IsNucleotide().

3 CSeq_loc_Mapper — adjust segment length when mapping between nucleotide and protein.

4 CAlign_CII — now reports (as warning) and skips alignments of zero length.

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

GenBank data loader:

1 Changed default reader to ID2.

2 HaveCache(), PurgeCache()— new methods, for cache control.

OM++ DEMO program (objmgr_demo)

1 Added options to use LDS and BLAST data loaders.

BUILD FRAMEWORK (UNIX)

1 Support running the test suite under auxiliary checkers such as 'valgrind'.

2 Add checks for wxWidgets 2.6 and Freetype (version 2), and make small improvements to checks for other third-party libraries.

3 Add an NCBI_DEPRECATED macro that can be used to mark deprecated functions.

4 Allow projects to specify "negative requirements", which must be unsatisfied (but known to 'configure') for them to be buildable.

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

5 Relink applications and libraries when their makefiles change (but recompile them only by explicit request, or if actual sources change).

PTB — Project Tree Builder for MSVC++ .NET

1 Pay more attention to project requirements when analyzing project tree — "unrecognized" requirements now result in the exclusion of the project.

2 Added support of negation of requirements in source makefile.

3 Implemented processing of local (for a specific makefile) macros.

APPLICATIONS

1 WindowMasker:

WindowMasker is a program that identifies and masks out highly repetitive DNA sequences and DNA sequences with low complexity in a genome using only the sequence of the genome itself.

WindowMasker is described in: Morgulis A, Gertz EM, Schaffer AA, Agarwala R; WindowMasker: Window based masker for sequence genomes. Bioinformatics, to appear. Advance access: Nov. 15, 2005. (Please cite this paper in any publication that uses WindowMasker.)

2 DustMasker:

DustMasker is a program that identifies and masks out low complexity parts of a genome using a new and improved DUST algorithm. The main advantages of the new algorithm are symmetry with respect to taking reverse complements, context insensitivity, and much better performance.

The new DUST algorithm is described in: Morgulis A, Gertz EM, Schaffer AA, Agarwala R; A Fast and Symmetric DUST Implementation to Mask Low-Complexity DNA Sequences. Journal of Computational Biology, to appear. (Please cite this paper in any publication that uses DustMasker.)

GRID (DISTRIBUTED COMPUTING) FRAMEWORK

1 NetSchedule - number of performance optimizations and bug fixes.

2 NetCache

- Implemented new transmission protocol with error checking.
- Improved error processing. Fixed bugs in communication protocol.
- Implemented no-wait-if-locked mode in GetBlob.
- Implemented BLOB locking detection. Improved overflow file management.

3 CWorkerNodeJobContext::PutProgressMessage() — added an optional parameter to allow sending progress messages regardless of the rate control.

4 CRemoteCgiApp::PutProgressMessage() — added an optional parameter to allow sending progress messages regardless of the rate control.

5 CGridCgiApplication — new optional parameter "expect_complete" in the configuration file, to control for how long the CGI should wait before showing the "wait" HTML page. If during this time the result fromWorker Node is received, then the "real" result HTML page is showed.

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

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, 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. Any of the applications can be built as command line apps by tweaking the build scripts or the CodeWarrior projects.

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

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

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).

CodeWarrior

To build the toolkit with CodeWarrior use an AppleScript editor to open and run the script files makeLibs.met and makeApps.met. You must use a script editor capable of opening a script file larger than 32K, such as Apple's Script Editor v2.0, or Smile. Script Editor v1.9 will not work since makeLibs.met just got too big. The command-line tool osascript also works.

On running the scripts you will be prompted as to which targets you want to build. Or if you always build the same targets they can be specified by including an empty file or folder in the compilers:mac_prj folder with the name 'Build' followed by the keywords of the targets you want built. The keywords are: Debug and Final. For example, to build only the Debug targets use: "Build Debug", to build both debug and release (final) versions: "Build".

If you install the C++ Toolkit under a different name than "ncbi_cxx" or in a different location than your home directory, you can edit the script's properties, pRootFolderName and pRootFolderPath, to override these defaults. Note: these paths, and those mentioned below, must be entered in a Mac format (e.g. disk:Users:username :) not in Unix format (e.g. /Users/username/). The disk name (and its following colon) may be omitted.

Certain third party libraries (see Table 1) are required to build some parts of the C++ toolkit. The scripts will try and find them if they are in your home directory, or you can specify where they were installed using properties at the beginning of the script.

Table 1. Third Party Libraries

Library	Property	Example
FLTK 1.1.6 (w/ NCBI patches #5)	pFLTKRootFolder	" home:mhome:fltk-1.1.6-ncbi5"
BerkeleyDB 4.3.21	pBdbRootFolder	"Users:myhome:mylibs:db-4.3.21" (or "... db-4.3.21")
SQLite 2.8.13	gSqliteFolder	

You do not have to build the FLTK or BDB libraries separately. This is done by the scripts and CodeWarrior along with the Toolkit libraries. Just unpack the source bundles in your home directory or where ever you have specified in the appropriate properties. The root folders for FLTK and BDB do not have to have any particular names. If there is more than one version

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

the scripts will grab whichever is last alphabetically (e.g. fltk-1.1.4r2 will get used instead of fltk-1.1.3).

The scripts normally halt on any CodeWarrior compilation errors. If you want them to continue and save errors, set the next script property, pSaveContinueOnErrors, to true. Compilation errors for a project will be saved in a file in the same folder as the project being built, with a name in the following format: projectName-targetNumber.errs (e.g. xncbi-2.errs).

The Genome Workbench's configuration file(s) is stored in the user's Library:Application Support:gbench folder.

CFM builds are not supported. OS 8 or 9 are not supported. We know 10.4 works. We think 10.1 still works, 10.2 might work, and 10.3 most probably works.

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-2.4.23 (w/ LIBC 2.3.2)	INTEL	GCC 3.4.0 ICC 8.0 (GCC 3.0.4, 2.95.3 - nominal support)
Linux-2.6.11.10 (w/ LIBC 2.3.3)	INTEL-64	GCC 4.0.1 ICC 9.0
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**
Solaris-9	INTEL	C++ 5.3 (WorkShop 6u2) patch 11168**6**-13
IRIX64-6.5	SGI-Mips	MIPSpro 7.3.1.3m (64-bit, 32-bit)
FreeBSD-4.10	INTEL	GCC 3.4.2
Tru64 (OSF1) V5.1	ALPHA	GCC 3.3.2

Table 3. MS Windows and Supported Compilers

Operating System	Compilers
MS Windows	MSVC++ 7.1. See documentation for building the Toolkit with MS Visual C++ .NET .NOTE: We also have 3rd-party packages archive for this platform, easily built with MSVC++ .NET (7.1).

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

Table 4. Mac OS X, and Supported Compilers

Operating System	Compilers
MacOS 10. 4 (darwin7.9.0)	GCC 3.3
MacOS 10. 4	CodeWarrior 9.**5**
MacOS 10. 4	Xcode 1.5 - 2.2.**1**

Caveats and Hints

MacOS 10.X / CodeWarrior 9.2

1. Not all of the test or demo applications are built.
2. The source code for the latest release of FLTK (1.1.x), BerkeleyDB (4.3.x) and SQLite (2.x) should be present. See the installation instructions for details.

MacOS 10.2/GCC 3.3

At least the GCC 3.3 update for Dec. 2002 Developers Tools required from Apple.

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).
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.

The NCBI C++ Toolkit Book

The NCBI C++ Toolkit Book

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 January 25, 2006.