CoCoALib - Design #1783

Code & doc structure: one-big-file or many-small-files?

24 Feb 2024 18:32 - John Abbott

Status:	In Progress	**Start date:**	24 Feb 2024
Priority:	Normal	**Due date:**	
Assignee:		**% Done:**	10%
Category:	Tidying	**Estimated time:**	0.00 hour
Target version:	CoCoALib-0.99900	**Spent time:**	0.60 hour

Description

I'd like to discuss briefly the pros & cons of many-small-files and one-big-file. Currently the CoCoALib code is a mixture:

- NumTheory.H is one-big-file (but the impl files are several)
- SparsePolyOp-XXX is many-small-files (both headers & impls)

Can we establish guidelines/recommendations (for code & for doc)?

Related issues:

Related to CoCoALib - Support #1672: Documentation: a good model to follow?	**New**	**15 Mar 2022**
Related to CoCoALib - Support #1510: Documentation for SparsePolyOps?	**New**	**14 Oct 2020**
Related to CoCoALib - Support #1265: Unregistered TXT files in doc	**Closed**	**27 Mar 2019**

History

#1 - 24 Feb 2024 18:33 - John Abbott

- Status changed from New to In Progress

- % Done changed from 0 to 10

Thoughts about one-big-file vs many-small-files:

For source code:

- ASSUME many-small-files means many header files & many impl files -- could in principle have few header files and many impl files
- compilation of many-small-files is likely easier on a computer with little RAM
- compilation of many-small-files may be faster on a computer with many cores
- recompilation after adding a new function is likely faster with many-small-files
- changing an existing API (fn-name, or arg-list, or rtn-value) may involve editing many more files with many-small-files -- I would not expect much difference in recompilation time
- with many-small-files it may not be obvious where the impl is to be found (or where a new impl should be placed, e.g. if it operates on 2 objets of different type)
- with many-small-files the impl files will need to have more include directives (and it may take some effort to discover exactly which header files should be included)

It is not clear to me that the convention of structuring the header files
and impl files the same way is really a good idea; it is a natural idea,
and is quite possibly helpful for the developers. There is a connection
between fine-grained header files and "minimal recompilation" -- this surely
saves time for the developers.

For documentation:

- CONVENTION: we have typically had a 1--1 relationship between header files and doc files (no doubt with some exceptions)
- with many-small-files each doc file is shorter, and thus likely easier to digest -- though a well-structured big file might be just as easy
- with many-small-files it may not be obvious in which doc file to look for doc about a specific fn -- this could be resolved by having a good search mechanism (JAA thinks the CoCoA-5 manual works pretty well)
- with many-small-files there is possibly more need for xrefs (??)

It is not clear to me that the CONVENTION of structuring the doc files
the same way we do for the header/source files is really a good
convention. The convention is probably helpful for developers, but a
normal user probably does not care

#2 - 24 Feb 2024 18:34 - John Abbott

A brief phone discussion with Anna yesterday suggested a vague preference for many-small-files.
But no genuine reasons for the preference were stated.

#3 - 01 Mar 2024 09:36 - Anna Maria Bigatti

- Related to Support #1672: Documentation: a good model to follow? added

#4 - 01 Mar 2024 09:36 - Anna Maria Bigatti

- Related to Support #1510: Documentation for SparsePolyOps? added

#5 - 01 Mar 2024 09:37 - Anna Maria Bigatti

- Related to Support #1265: Unregistered TXT files in doc added