Enumeration
mathematical library

Mosfiqur Rahman
PROJECT MANAGER
Debdut Karmakar
DEVELOPER

Voice: (215)220-5007
Email: info@enumeration.ml
www.enumertion.ml

Project Introduction:

- The name of our project is "Enumeration Mathematical Library" acronym as EML.
- This will be a numerical Math library for C and C++ programmers. Firstly, we will start with Number Theory, and subsequently, explore other areas of mathematics.
- Our project will be an open source one, which will be published under the GNU General Public License.
- Our intention is to follow the open source business model. So, it will be really useful for the scientific computing community.
- Our first stable version will be named EML 0.1 alpha release.

Project Abstract:

BASIC CONCEPTS →

- EML will be a fast C/C++ library for mathematical computations.
- It will concentrate mainly in Number Theory supporting from,

- ✓ Arbitrary Large Integers
- ✓ Arbitrary Precision Numbers
- ✓ Basic Number Theory
- ✓ Power Series
- ✓ Polynomials
- ✓ P-adic fields
- ✓ Complex, Quaternion's and Higher Fields
- ✓ Matrix Algebra
- ✓ Algebraic Number Theory
- ✓ Combinatorial Number Theory, and so on.

Project Abstract [Continued]:
BASIC CONCEPTS [Continued]→

- EML will provide various low-level routines for fast arithmetic. Moreover, it will be extensively documented and tested.
- It will support x86-64 architectures, and also work for parallel programming.
- We are planning to write it in ANSI C, which will run in many platforms.

Project Abstract [Continued]:

USER PROFILE→

- EML will be primarily used in scientific computing and mathematical research.
- It will also be used in cryptography and security systems.
- Its support for parallel computing will make it suitable for computing clusters.
- Engineers will be able to solve complex mathematical problems with an extremely fast calculating experience.

Project Abstract [Continued]:

EXTERNAL INTERFACE→

- EML is a C++/C library that can be accessed through its relevant header files.

USER INTERFACE→

- In C /C++programs, EML library can be accessed by including the header files.
- As an example for C, to include arbitrary precision library from EML, #include <eml/arb.h>. Then while compiling, use the proper flags for EML like -leml in GCC.
- For C++, you can #include <eml/arb>. Then while compiling, use the proper flags for EML like -leml in GCC.

Project Abstract [Continued]:

Data Interface→

- EML is generally stand-alone and it won't require any data interface in general cases.
- However, in case of cluster computing, data interface is required for data transfer between different nodes. Here, OpenMP and MPI libraries will handle it.

Graphical Rendering→

- In general, graphical renders aren't fast and resource consuming. As current graphics packages and such libraries are not fast enough, these will be re-written.

Project Abstract [Continued]:

BASIC FEATURES→

- *Fast Library*: EML will be designed carefully to make as fast as possible, both for small and big operands. To achieve this goal, we will use-

- Full words as the basic arithmetic type
- Fast Algorithms
- Highly optimized C codes for the most common inner loops and so on.

- *Parallel Programming*: EML will also be available in parallel programming by using OpenMP and MPI libraries.

Project Deliverables:

- Our plan is to complete all preparation and prior works by the winter term, which includes-

✓ Breaking down everything into different steps

✓ Making a perfect schedule for keeping a work-balance

✓ Mastering all those we will need to accomplish this project and so on.

❑ By the spring term, we are planning to deliver a stable working library as our alpha release.

✓ We will make our source code available to public as an archive .zip or .tar of source code per open source ideology.

✓ We will also provide a dynamic compiled library as .so file. with complete documentation and system manual.

Special Resources:

- We need a computing cluster to implement the support for parallel programming properly.
- In fact, we are looking for a collection of previously implemented algorithms for reducing time constraint.
- However, we've already started working to fix this issue by making our own computing cluster using several Banana Pi M2s.

Expertise:

- This project requires expertise in mathematics and complex-algorithms.
- It also requires enough knowledge in parallel programming which is still needed to be mastered by us, and we are working on it.

Now, Any Questions?