

GSoC Proposal

*Project Idea: Tutorial Development with
Clang-Repl
2 April, 2023.*

Mentor: Vassil Vassilev

Krishna Narayanan

Table Of Contents

Curriculum Vitae	3
Details	3
Background	3
Academic performance	3
Projects	4
Co-curricular activities	4
Which features will be added to the project?.....	5
Why am I suitable for this project ?	5
Project Synopsis	7
Abstract	7
Pre-commencement	8
During GSoC	8
Phase I (Community Bonding)	8
Phase II (Coding)	8
After GSoC	8
Implementation plan	9
Timeline	10
Proof of concept	12
Checklist	12
Conclusion	13
Appendix	13

Curriculum Vitae

1.1 Details

- **Name:** Krishna Narayanan
- **Email:** krishnanarayanan132002@gmail.com
- **Mobile:** +91 9082411935
- **Timezone:** +05:30 hrs
- **Institute:** Veermata Jijabai Technological Institute (VJTI)
- **Course:** Bachelor of Technology
- **Specialization:** Electronics and Telecommunications engineering
- **Github:**<https://github.com/Krishna-13-cyber>
- **LinkedIn:**<https://www.linkedin.com/in/krishna-narayanan-295b57209/>

1.2 Background

I am an undergraduate student pursuing B.Tech. from Veermata Jijabai Technological Institute, Mumbai, India. My areas of interest are compiler development, Robotics, Embedded Systems in specific. I am a passionate and vivid learner interested in contributing to the open source community. I am actively involved in the robotics club of my institute. I have been coding for 5 years, having good experience in C, C++, python.

1.3 Academic Performance

I am currently in the third year of my bachelors, and have scored an overall CGPA of 8 (out of 10). I had scored **94%** in High School.

I had scored **99.6%** in the State Common Entrance Test. I have also qualified other entrance tests with commendable performance.

1.4 Projects

I have worked on the projects of embedded system and ROS in recent times which can be viewed from my Github profile: <https://github.com/Krishna-13-cyber>

Some of the **main project ideas** I have worked on were:


- I have worked on a **transpiler(source to source transformer)** <https://github.com/VedantParanjape/simpPRU> which transpiles a custom language to PRU C (supported by PRU's of beagle) during my GSoC 2022.

- I am working on a C++ partial evaluator / interpreter tool based on boost libraries qi parsers. <https://github.com/SAtacker/quick-ftxui>
- <https://github.com/Krishna-13-cyber/BluetoothJoystick> which aims to make a joystick with the feature of bluetooth connectivity in esp-32 microcontroller.
- <https://github.com/Krishna-13-cyber/Micromouse-Eklavya> which we had designed a bot from scratch, in which we had used a left hand algorithm for solving the maze in the shortest time. We got a special mention for this competition, coming in the **top 5** positions in the [International Micromouse Challenge](#) hosted by Techfest IIT, Bombay.
- I have been working on a project [Quick-FTXUI](#) which generates C++ terminal which is based out of FTXUI.

To give a background of the other tasks I have done:

- PCB-Designing, <https://github.com/Krishna-13-cyber/PCB-Designing-LSA>
- Learning **Compiler Framework and debugger.**

1.5 Co-curricular activities

Line following and self balancing robot workshop  Wall-E 2.0 : I have mentored students in a workshop conducted by the Robotics committee of our institute. I taught the concept and displayed the demo of LSA.

Pianist: I have been trained for 8 years and like to play during my leisure time.

Orator: I have represented my High School in many inter school competitions.

Open Source Community: I generally contribute and refer to open source communities to discuss and resolve issues.

2. Which features will be added to the project?

- Write several tutorials demonstrating the current capabilities of clang-repl. (workings and limitations if any)
- Investigate the requirements for adding **clang-repl** as a backend to **xeus-cling**.
- Implement the **xeus kernel protocol** for clang-repl.
- Prepare a blog post about xeus-clang-repl.
- Present the work at the relevant meetings and conferences.

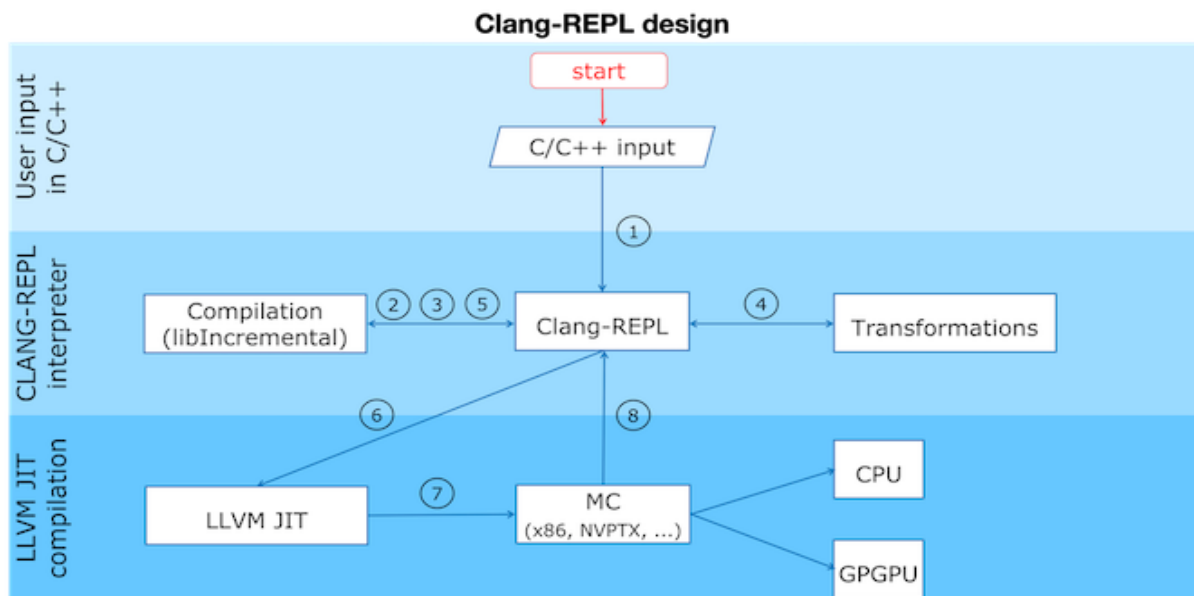
3. Why am I suitable for this project ?

- This project requires good knowledge and background of **compiler development**.
- I have been exploring the **compiler developer environment** for quite some time, having good knowledge about the compilers.
- I have been contributing to **open source** actively in the **compiler development**: <https://gcc.gnu.org/pipermail/gcc-patches/2022-March/591595.html> , <https://gcc.gnu.org/pipermail/gcc-patches/2022-February/590817.html>.
- I have a **decent experience** in C, C++ which I can enhance over the course of the project.
- I have opened a patch regarding **static_assert message** redundancy and am currently working on it. <https://reviews.llvm.org/D146376/new/>
- I am an open source enthusiast, passionate about technologies and have always dedicated myself to the work I do with utmost perfection. I have **no major commitment** other than **GSoC** during the summer break and would give the best of my potential to complete the project idea in the given timeframe.

4. Project Synopsis

4.1 Abstract

This project aims to add support for Xeus with the newly added clang-repl which is inspired from cling. The need for clang-repl is that it presents opportunities for rigorous open source development. However even though it is inspired by cling, not all of clang-repl and cling are same i.e. they are similar but work needs to be done to add xeus protocol support for clang-repl.



4.2 Pre-commencement

- To go through the concepts which require a change in the compiler codebase in **clang-repl** with respect to the requirement.
- I will go through the resources of clang-repl,cling,xeus,xeus-cling and try running a few examples after building them from source.
- Getting familiar with the code base and asking the mentors if I encounter any doubts.
- Discussion related to different types of examples, tests to be run in clang-repl.
- Plans how to integrate clang-repl as a backend to xeus-cling.
- Plans how to implement xeus kernel protocol for clang-repl.

4.3 During GSoC

4.3.1 Phase I (Community Bonding)

- Firstly, I will understand the guidelines and the conventions of coding standards at LLVM.
- Setup Phabricator for patch submissions and review.
- List out all the features to be implemented, demonstrated and tested.
- Setup all the cling, clang-repl, xeus and xeus-cling for testing and development.

4.3.2 Phase II (Coding)

- Understand and start implementing the tutorials required for the **clang-repl** with reference to current docs and setup installation steps and a **basic startup guide**.
- Write code in an efficient way, to boost readability and maintainability.
- Understand the xeus-cling and its integration of xeus for kernel protocol with cling interpreter.
- Feasibility of clang repl with **xeus-cling** and **protocol for xeus**.
- Write **test cases** for the developed codes and try their functionality.
- Create a detailed documentation of the code with explanation and **benchmark** the results of clang-repl.

4.4 After GSoC

- After GSoC, I plan to maintain the project of **clang-repl development** changes along with the other mentors. I would resolve the issues and merge pull requests on a regular basis.
- I would still continue to contribute to the clang-repl. I am willing to contribute to LLVM, as it gears up my interest to work in this domain. I will keep on brainstorming to think about the add-ons and features which could be added.

5. Implementation plan for GSoC

- *Clang Tutorial Development*
- The clang tutorial development starts from **basic examples** as below. The working of clang-repl as an interpreter.

```
krishna@krishna:~/llvm-project/ninja_build/bin$ ./clang-repl
clang-repl> int i = 12;
clang-repl> printf("%d\n", i);
In file included from <<< inputs >>>:1:
input_line_1:1:1: error: use of undeclared identifier 'printf'
printf("%d\n", i);
^
error: Parsing failed.
clang-repl> #include<stdio.h>
clang-repl> printf("%d\n", i);
12
clang-repl> i++;
clang-repl> printf("%d\n", i);
13
clang-repl>
```

- Using the appropriate flags as per requirement for the use case. **Basic demonstration** regarding these different cases:

```
krishna@krishna:~/llvm-project/ninja_build/bin$ ./clang-repl --help
USAGE: clang-repl [options] [code to run]

OPTIONS:
Color Options:
--color                               - Use colors in output (default=autodetect)

General options:
--Xcc=<string>                        - Argument to pass to the CompilerInvocation
--abort-on-max-devirt-iterations-reached - Abort when the max iterations for devirtualization CGSCC repeat pass is reached
--allow-ginsert-as-artifact           - Allow G_INSERT to be considered an artifact. Hack around AMDGPU test infinite loops.
--atomic-counter-update-promoted      - Do counter update using atomic fetch add for promoted counters only
--atomic-first-counter                 - Use atomic fetch add for first counter in a function (usually the entry counter)
--bounds-checking-single-trap         - Use one trap block per function
--cfg-hide-cold-paths=<number>        - Hide blocks with relative frequency below the given value
--cfg-hide-deoptimize-paths           -
--cfg-hide-unreachable-paths         -
--cost-kind=<value>                   - Target cost kind
--throughput                          - Reciprocal throughput
--latency                              - Instruction latency
--code-size                            - Code size
--size-latency                         - Code size and latency
--debug-info-correlate                 - Use debug info to correlate profiles.
--debugify-func-init=<ulong>           - Set max number of processed functions per pass.
--debugify-level=<value>               - Kind of debug info to add
--locations                            - Locations only
--locations-variables                  - Locations and Variables
--debugify-quiet                       - Suppress verbose debugify output
--disable-auto-upgrade-debug-info     - Disable autoupgrade of debug info
--disable-ir-ppt-opt                   - Disables irptopt/irpoint roundtrip optimization
--do-counter-promotion                 - Do counter register promotion
--dot-cfg-ssa=<file name for generated dot file> - file name for generated dot file
--enable-cse-in-irtranslator           - Should enable CSE in Irtranslator
--enable-cse-in-legalizer              - Should enable CSE in Legalizer
--enable-gvn-hoist                     - Enable the GVN hoisting pass (default = off)
--enable-gvn-memdep                    - Enable the GVN sinking pass (default = off)
--enable-gvn-sink                      -
--enable-load-in-loop-pre              -
--enable-load-pre                      -
--enable-loop-simplify-cfg-term-folding -
--enable-name-compression              - Enable name/filename string compression
```

- *Investigate requirements for clang-repl as backend xeus-cling*
- The **xeus-cling** takes cling as its interpreter. I will be working on clang-repl for establishing the xeus setup integrated with clang-repl.
- The interpreter has to be recognised i.e **clang-repl** as the **primary interpreter** with support in the **backend**.
- **Xeus::xkernel,xeus::load_configuration**, will be our **initial targets** on which I will work and gradually add support further. Xeus-cling takes cling interpreter where we will be using directly from clang interpreter.

- *Implement the xeus kernel protocol for clang-repl.*
- The **xeus kernel** takes the initiative for kernel protocols, where developers get to start their kernel for their use cases. The following below is the start of a new kernel for client and user interaction.

```

`execute_request_impl`: See execute_request_
Code execution request from the client.
`complete_request_impl`: See complete_request_
Code completion request from the client.
`inspect_request_impl`: See inspect_request_
Code inspection request (using a question mark on a type for example).
`is_complete_request_impl`: See is_complete_request_
Called before code execution (terminal mode) in order to check if the code is complete
and can be executed as it is (e.g. when typing a `for` loop on multiple lines in Python, code will be
complete when the `for` loop has been closed).
`kernel_info_request_impl`: See kernel_info_request_
Information request about the kernel: language name (for code highlighting),
language version, terminal banner etc.
`shutdown_request_impl`:
Shutdown request from the client, this allows you to do some extra work before the kernel
is shut down (e.g. free allocated memory).

```

- The **xeus kernel** has to be integrated with the clang-repl for getting the communication which are present in xeus which initiates kernel protocols. For integrating we can refer cling integration and snippets can be referred from cling to get **clang-repl** in working with **xeus-kernel**.

```

namespace xcpp
{
    class XEUS_CLING_API interpreter : public xeus::xinterpreter
    {
    public:

```

- The **xcpp** which has been present in the xeus-cling, in the similar fashion this is the resource which provides us the kernel modules which can be integrated for clang-repl.
- *Documentation and testing*

Documentation is the guide for conveying the work which has been done for the user to gauge. I will document the changes done in the codebase with respect to Clang-repl.

- *Submit final work and final mentor evaluation*

Submission of the work done which has been documented and evaluated by mentors. Patching the required changes after suggestion from mentors.

6. Timeline

Community Bonding (May 4, 2023 - May 28, 2023)	
May 04 - May 11 (07 days)	Getting familiar with work culture of the organization
May 11 - May 18 (07 days)	Discussion on resources and tools relevant to the project
May 19 - May 26 (07 days)	I have my semester exams during this period but I will manage time and go through the clang-repl resources.
May 27 - May 29 (2 days)	Trying small patches and Discussion with mentors.
Coding (May 29, 2023 - July 10, 2023)	
May 29 - June 5 (07 days)	Initial tutorial development with running basic examples and test cases. (Video depiction and snippets)
June 6 - June 13 (07 days)	Testing and debugging Clang-REPL if there are any bugs or false positives with the edge cases which have been run on Cling (Video depiction and snippets).
June 14 - June 21 (07 days)	A detailed overview of the difference and similarity between CINT Cling and Clang-REPL . (Video depiction and snippets)
June 22 - June 30 (07 days)	Regarding the investigation part, I will go through the xeus-cling where xcpp is the main target which we can opt for and in the similar way generate a namespace

	for the clang-repl interface.
July 01 - July 10 (07 days)	<p>1.I will be trying out taking the interpreter which resides in xeus::cling is being taken for cling support,Interpreter.h. In our case, we will be using Clang dependency for working with Clang-REPL as the interpreter.</p> <p>2.Create a detailed blog regarding.</p>
Evaluation 1 (July 10 - July 14, 2023)	
July 10 - July 14 (04 days)	Evaluation from mentors, changes done as suggested by mentors.
Coding (July 15, 2023 - August 21,2023)	
July 15 - July 22 (07 days)	<p>Implement the xeus kernel protocol for clang-repl.</p> <p>1.We will be using the namespace nlohmann to use JSON library.</p> <p>2.Implement the class XEUS_CLANG-REPL_API interpreter for getting all the functions which has to be used.</p>
July 23 - July 31 (07 days)	<p>Implement the xeus kernel protocol for clang-repl.</p> <p>3.Implement xeus::xkernel kernel for invoking it from the xeus.</p> <p>4.Establish protocol by using xeus protocols following the convention of cling interpreter.</p>
August 1 - August 7 (07 days)	Fix bugs which we developed in the previous phases/weeks.
August 8 - August 15 (07 days)	Blogs and posts regarding the tutorials in clang-repl and jupyter .

August 16 - August 20 (02 days)	Ask for the review from the mentors and fix bugs if any. Preparing for the completion of the project and final documentation.
Evaluation 2 (Aug 21 - Aug 28, 2023)	
Aug 21- Aug 23 (03 days)	Merging Phabricator PR's if any
Aug 24- Aug 28 (04 days)	Will try to implement more features after suggestions from mentors.
Submission of GSoC Project	

Many thanks for considering my proposal and taking time out for the effort being taken. I hope to work with LLVM in the near future with lots of exciting and interesting developments to come up 😊!