Improving Interactive Programming Documentation for the HSF

This is a proposal by the HEP Software Foundation (HSF) to support its participation in the 2023 Google Season of Docs program. CERN-HSF and the mentors involved in this proposal have been involved in GSoC and GSoD programs for many years. CERN-HSF projects have benefited significantly and numerous participants have gone on to continue working with us as they have advanced in their open-source career.

The HEP Software Foundation Organization

High Energy Physics (HEP) is an exciting field where large collaborations of scientists collect and analyze petabytes of data from high-energy physics experiments, such as those at the Large Hadron Collider, hosted at the CERN laboratory in Geneva, Switzerland. Some of the questions that we collectively ask are:

- what are the fundamental blocks that make up our Universe?
- what is the nature of dark matter and dark energy?
- what is the nature of the asymmetry between matter and antimatter?
- what was early Universe like?

To answer these questions, particle physicists build software to simulate and analyze what happens in particle physics detectors. The scale of the needed research software and the computing facilities needed to use it are vast compared to most other scientific fields. Plans for exabyte scale data sets means that particle physics researchers must push technologies to enable their science.

The HEP Software Foundation (HSF) brings together researchers participating in the GSoC and GSoD programs. Particle physics researchers started in GSoC in 2011, subsequently joining together to use HSF as an umbrella organization in 2017.

Season of Docs Project

Particle physics relies heavily on open-source codes that compute key quantities for both theory and experiment. Often such codes have been developed for many years, with early documentation rendered incorrect or incomplete by later developments. Key user-facing information can be very difficult to find (e.g. in automatically generated code-interface documentation). It can be difficult for new physics-oriented users and contributors to get involved: the docs that exist are more for the developers and often assume too much technical prowess.

We seek to advance the documentation in the area of interactive analysis, both in the context of C++ and support for C++/Python interoperability. Our intent is to build the sort of documentation

that enables user engagement while being easy to update as our codes continue to evolve and improve.

Interactive Analysis

HEP researchers have developed several unique software technologies in the area of data analysis. Over the last decade we developed an interactive, interpretative C++ interpreter (aka REPL) as part of the ROOT data analysis project. We invested a significant effort to replace CINT, the C++ interpreter used until ROOT5, with a newly implemented REPL based on LLVM – Cling. Cling is a core component of ROOT and has been in production since 2014. Cling is also a standalone tool, which has a growing community outside of our field. It is recognized for enabling interactivity, dynamic interoperability and rapid prototyping capabilities for C++ developers. For example, if you are typing C++ in a Jupyter notebook you are using the xeus-cling Jupyter kernel.

We are in the midst of an important project to address one of the major challenges to ensure Cling's sustainability and to foster that growing community: moving most parts of Cling into LLVM. Since LLVM version 13 we have released a version of Cling called Clang-Repl within LLVM itself. We subsequently focused on the language interoperability capabilities of Cling. One user facing application of our libInterOp, together with Clang-Repl, is Xeus-Clang-Repl, which is a replacement for xeus-cling using these new codes. As we advance the implementation and generalize its usage we aim for improving the overall documentation experience in the area of interactive programming in various environments.

Scope

This project will audit the existing documentation for the Clang-Repl (interactive C++), Xeus-Clang-Repl (notebook-based C++ and Python platform) and libInterOp (bridging automatically C++ and Python). We aim to Identify gaps in the information or presentation from the point-of-view of new, science-oriented users.

The anticipated scope of the work is:

- Improve user and developer documentation about xeus-clang-repl
- Write several tutorials demonstrating the current capabilities of clang-repl.
- Prepare a blog post (or posts) about clang-repl and xeus-clang-repl.
- Improve user and developer documentation about InterOp library API and usage.
- Develop a set of blog posts on how to use the InterOp library API and usage.
- Develop a set of blog posts on how to use the InterOp library together with higher-level tool kits such as Cppyy.

Work that is out-of-scope for this project:

- Candidates are not expected to have detailed physics or interactive compilation knowledge. Technical writers will focus on explaining the technical systems, working closely with our research teams to ensure the end results have the appropriate mix of scientific reasoning and technical detail.

- Candidates are not required to have past experience with a particular set of documentation tools.

Success metrics

The project success metrics include: publishing at least 3 blog posts on blog.llvm.org or other fora as agreed with project mentors; publishing at least 2 LLVM help document in clang.llvm.org/docs/; publish at least 2 tutorials about more advanced features at compiler-research.org/tutorials/;.

Timeline

The project can potentially involve at least two technical writers. As the area is specialized, we expect a month of orientation (June), during which the documentation audit will be performed for all project components. The project timeline can be adopted for any scheduling constraints of writers. Primary emphasis is initially on tutorial and notebook development, and subsequently on developer documentation.

Here we summarize a proposed high-level schedule of all project components.

Dates	**Activities and Goals**
June	Orientation & documentation audit. 1 week for meeting the team; 1 week for getting into the relevant technologies; 1 week for environment setup; 1 week for doc. audit.
July	Demonstrate interactive C++ use cases in the LLVM documentation by developing basic and advanced documentation for Clang-Repl within the LLVM documentation system (3 weeks). 1 week buffer period.
August	Develop advanced libInterOp and Xeus-Clang-Repl documentation and tutorials (4 weeks). 2 weeks for communicating with the legacy Xeus-Cling team
September	Continue work started in August. Write a blog post on a working notebook demonstrating a tutorial (1 week). Start to review and enhance the developers' documentation, examples and tutorials (3 weeks)
October	1 week buffer. 2 weeks integration and validation of the written documentation, 1 week audit and outlining further work (if applicable).

Budget

The budget is separated into separate items for technical writer support for each project. Budgets were derived based on past experience gained during the 2022 Season of Docs program.

Item	Amount ($)	Running total ($)	Notes/justifications
Technical writer for Clang-Repl and Xeus-Clang-Repl documentation	6500	6500	
Technical writer for InterOp library documentation	6500	13000	
Volunteer Stipends	1000	14000	Stipend for two student mentors ($500/each) to assist with tutorial development
Total		14000	

Previous experience with GSoC and GSoD

Previous participation in Season of Docs, Google Summer of Code or others: CERN-HSF has participated in GSoC since 2017, and has three previous projects with Season of Docs (2019, 2020, ane 2022). The project mentors have previous experience in GSoC and Season of Docs: a total of around 30 student projects and between 2 and 10 years of participation each. In addition, mentors work in numerous student programs including those sponsored by CERN and the US National Science Foundation (NSF), are experienced in multidisciplinary environments, and routinely handle remote working and supervision roles.

Previous experience with technical writers or documentation: Mentors have previous experience with semi-automated code documentation using Doxygen and Sphinx, developing wiki/MarkDown based self-guided tutorials, interactive tutorials, and LaTeX/PDF-based manuals. User experience with these materials will be an important component of guiding our mentorship in this project. It will help us focus efforts on the key aspects and enhance the overall success of the project.