Improving Analysis Tool Documentation for the HSF

This is a proposal by the HEP Software Foundation (HSF) to support its participation in the 2022 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 two areas of HEP research: interactive analysis based on C++; and reinterpretation of analysis results. Our intent is to build the sort of documentation that enables user engagement while being easy to update as our codes continue to evolve.

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 a version of Cling called Clang-Repl. As we advance the implementation and generalize its usage we aim for improving the overall documentation experience in the area of interactive C++.

Reinterpretation

"Reinterpretation" codes aim to reuse data to make new statements about fundamental physics. These codes are meant for wide use in the physics community, but for that to happen much better user-facing documentation needs to be created. We will focus on improvements to the Rivet and Gambit projects, two largely C++ physics applications for reinterpretation, which require a range of technical skills -- from command-line Unix to C++ programming -- in order to get the most from their capabilities.

Scope

This project will audit the existing documentation for the Clang-Repl (interactive C++), Rivet (Reinterpretation) and Gambit (Reinterpretation) codes. Identify gaps in the information or presentation from the point-of-view of new, physics-oriented users.

The scope of the reinterpretation component continues with:

- Create onboarding guides for Rivet and Gambit, incorporating elements of tutorial, explanations of software and physics (with input from supervisors!) rationales in the designs, and clear explanations of best practice in system use and writing of user-provided plugins.
- Improve the generated documentation for the Rivet and Gambit codes (e.g. existing Doxygen and Sphinx) to be more user-oriented, and easier to browse. Making the API documentation "future-proof" and synced with new releases is a key goal, as both codebases are actively evolving.
- Incorporate feedback from documentation testers in the project and the user community.

The scope of the interactive analysis component continues with:

- Demonstrate interactive Clang-Repl use cases in the LLVM documentation by developing basic documentation and tutorials apt for Clang-Repl
- Develop advanced Clang-Repl tutorials integrate Clang-Repl into Xeus and contribute to a community blog post on them.
- Review and enhance the developers' documentation, examples and tutorials

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

We will track the numbers of pull requests, Docker, and tarball downloads of the Rivet and Gambit packages, and the numbers of submitted plugins and published papers citing them. We will consider the project successful if the numbers of pulls/downloads increase by 20% on the tutorial branches, and the numbers of contributions/publications increases by 10% within a year of the new documentation becoming available.

The interactive analysis development success metrics include: publishing at least 2 blog posts on blog.llvm.org; publishing at least 3 LLVM help document in clang.llvm.org/docs/; publish at least 2 tutorials about more advanced clang-repl features at compiler-research.org/tutorials/; Reviewing and improving developers' documentation.

Timeline

The project can potentially involve 3 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.

For the reinterpretation areas, July-August will be spent on developing the new onboarding guides and tutorial material, and incorporating feedback. September will be spent on improving and integrating the new material with the packages' auto-generated documentation.

The interactive analysis technical work will nominally continue through October, however of course the 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 documentation audit.
July	Create onboarding guides and tutorials for Rivet and Gambit Demonstrate interactive C++ use cases in the LLVM documentation by developing basic documentation and tutorials apt for Clang-Repl (3 weeks). 1 week buffer period.
August	Test and improve guides for Rivet and Gambit Develop advanced Clang-Repl tutorials integrate Clang-Repl into Xeus (2 weeks). 2 weeks for communicating with the Xeus-Cling team and possibly porting the kernel to LLVM
September	Integration with generated code-doc systems in Rivet and Gambit Write a blog post on a working notebook demonstrating a tutorial (1 week). 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 compiler (Interactive analysis/clang-repl code, reinterpretation/Rivet code, and reinterpretation/Gambit code). Budgets were derived assuming the same wage per effort across each project component.

Item	Amount	Running total	Notes/justifications
Technical writer for Clang-Repl documentation	6500	6500	
Volunteer Stipends	1000	7500	Stipend for two student mentors ($500/each) to assist with Clang-Repl tutorial development
Technical writer for Rivet documentation	3750	12250	
Technical writer for Gambit documentation	3750	15000	

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 two previous projects with Season of Docs (2019 and 2020). All project mentors have previous experience in GSoC: a total of around 30 projects and between 2 and 10 years of participation. 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: Previous experience with semi-automated code documentation using Doxygen and Sphinx, plus wiki/MarkDown self-guided tutorials, and LaTeX/PDF-based manuals and live tutorial slides. While amateur, this experience of the importance of synchronisation and of what users need to know will help in supervising the technical writer to work in the required way and to focus on the key aspects.