



Boost Context Module Implementation for s390x

Naveen Naidu

naveennaidu479@gmail.com

Junior Year - Computer Science Undergraduate

PES University - Electronic City Campus

Bangalore, 560100

India

Why would you like to execute on this particular project and why would you be the best individual to do so ?

There are many libraries which are directly dependent on `boost.context` and are blocked from being available on the s390x due to the lack of `boost.context`. The absence of this module blocks the following modules:

- `boost.coroutine`
- `hhvm`
- `kicad`

The implementation of this project would clear this hurdle and thus would help directly and indirectly for at least 4 libraries/projects to be available on s390x. This would also help decrease the entry barrier for many system administrators who are unable to switch to mainframe due to the lack of tools in s390x.

This could also help increase the community/users of the s390x. This project would have a great impact for the community. And to add to that, the very idea of me helping the mainframe community in any small way possible, excites me.

As well as, I'm sure it will be an exciting challenge to get to grips with the s390x kernel architecture for mainframe. Learning more about how this architecture provides the flexibility of running Linux with the advantages of [fault-tolerant mainframe hardware](#) capable of over 90,000 I/O operations per second and with a [mean time between failure \(MTBF\)](#) measured in decades, is an opportunity I would not want to miss. This project would help me quench the thirst of my recently upsurging interest in mainframes.

Why me?

I am inquisitive by nature and have a burning desire to explore various fields to help people benefit from technology. I am an open source aficionado and have been a regular contributor to various open source projects. For this particular project, I have already read through the documentation of `boost.context` and can confidently say that I have a fair amount of the understanding of the source code of `boost.context` (I have explained my understanding about the project in the proposal below, please have a look there). I have also read through the various papers that had been published related to `boost.context` by [Oliver Kowalke](#) (maintainer and creator of `boost.context`). A few of the links to the papers

are:

1. <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2014/n4024.pdf>
2. <http://www.open-std.org/jtc1/sc22/wg21/docs/papers/2018/p0876r0.pdf>
(Currently reading this)

This project requires me to have a good hold on Assembly Language. I have taken courses in Basic Electronics, Analog and Digital Electronics , Embedded Computing Systems as a part of my undergraduate curriculum, and my lab experience consisted of programming tasks in MASM(Microsoft Macro Assembler). I had also taken a MOOC named `Build a Modern Computer from First Principles: From Nand to Tetris`, where I programmed the assembly codes for memory registers, counters, CPU, and Video Drivers from scratch . As a final test my assembly codes were successfully run on a emulator. To add to it, I've given a series of introductory talks on `Basic Electronics 101` to the junior students of my university.

Hence I can say that I have a good hold of working with assembly files and if need be, can learn any new language in a short span of time.


As a Google Code In mentor for the coala organisation, I was constantly bombarded with questions from the participants, which pushed me to dive deeper and learn additional concepts about various repositories associated with coala and become capable of explaining the intricacies of those repositories in a simple way to the students so that they can achieve the goals of their task . This experience strengthened my ability to understand huge code bases faster.

I also do not have any other commitments and I would be available throughout the summer. I am willing to work for 40 hours a week and would gladly put in more hours if need be for the successful completion of the project.

Though I have not yet contributed directly to the Boost community but I am more or so familiar with the works of the community and I can assure you that, I will be putting in my best efforts to learn any new technology that might be required for the project.

Please share details of your academic, industry and/or open source development experience, as well as other details as you see fit.

Academic Details



During my freshman year at University, my teammates and I were among the top 5 teams among 100 teams selected for a National Level Hackathon (Ingenius 2017). We developed an application named CRYsis - This application exploited the bluetooth connectivity of smartphones to create a **mesh network**. The aim of this software was to help people connect with each other in the times of natural calamities, when the cellular networks get wiped out entirely. Our application only requires the user to switch on bluetooth and it creates a bluetooth mesh network. This application supported the hopping of message via another devices to reach its destination.

I have been part of my university Computer Science Research team(SciBase). I led the development of an web scraper which helped the Researchers of my College gather metrics about journals from the various scientific journal websites. The scraper was designed in a way such that it can overcome the blockings from the research websites. This was used by the researchers to gather data to overcome the problem of fake citations.

I also have past experience working in the field of Computer Science as a Intern at a Fintech Company named **Fullerton India Credit Company Limited**. During the summer of my sophomore year, my team and I had developed a game to help people understand about 'Financial Literacy'. The product developed by us was also selected as a finalist in their annual Startup Hunt – Finnovatica. This internship gave me experience in working closely with end users in developing solutions.


During my 5th semester at University, I along with my teammates, had developed an application, which we named - DTS (Data Transmission through sound) . It was a hacky tool developed by us, using the zxing library provided by google. This application, took the text from the user - encrypted it - and transmitted it in a high frequency tone. This application was selected as the finalist for another National Level hackathon.

I have also taken various workshops for the students of my University and introduced them to Open Source development and the various benefits of being associated to it.

I was a part of the Technical Core Team for the annual hackathon hosted by my University. My responsibilities included handling the technical support of the event. During this time, I had set up various websites, built django applications and provided the team with various scripts to automate mailing.

Open Source Development

I have always been mesmerized by the impact an open source project can have. I have always had this intense desire to help people and open source had given me the platform to do so. And that's my main motive behind my contributions to open source organizations.



It gives me the adrenaline rush to just think about how my little change in a huge code base can make someone's life easy.

It's almost been a year since I have started contributing to Open source projects. I go by the handle [@Naveenaidu](#). I must say, I was hooked onto it from the very first day. I started out by contributing to an open source project named [coala](#), which is a static analysis tool built for most programming languages. I now am, on the *main developer team of coala*. I have submitted various PR's, opened and reviewed many issues and have helped many newcomer's set up their environment and get started with open source development.

I also have been the **Google Code In Mentor - 2018** for coala. I was mainly in charge of the coAST, mobans and documentation tasks.

I also contribute to [moremoban](#) organisation. Particularly to the [moban](#) repository and pypi - moban. Moban is high performance template engine (JINJA2) for web into static text generation. I have been recently added as one of the collaborators of the [pypi-moban](#) repositories.

Whenever I am free, I also make few small hacks for fun.

Overview

What is Boost?

Boost is a set of libraries for the C++ programming language that provide support for tasks and structures such as linear algebra, pseudorandom number generation, multithreading, image processing, regular expressions, and unit testing. It contains over eighty individual libraries.

The libraries are aimed at a wide range of C++ users and application domains. They range from general-purpose libraries like the smart pointer library, to operating system abstractions like Boost FileSystem, to libraries primarily aimed at other library developers and advanced C++ users, like the template metaprogramming (MPL) and domain-specific language (DSL) creation (Proto).

What is Boost.Context ?

boost.context is a foundational library that provides a sort of cooperative multitasking on a single thread. boost.context uses a concept called as fiber. A fiber provides the means to suspend the current execution path and to transfer execution control, thereby permitting another fiber to run on the current thread. This state full transfer mechanism enables a fiber to suspend execution from within nested functions and, later, to resume from where it was suspended.

The advantage of this library is that it reduces the number of CPU cycles drastically for context switching when compared to the context switching via system calls which usually involves OS kernel and thousand of CPU cycles. boost.context with it's faster execution time and better implementation proves to be a great tool for developers.

Project Overview

TITLE: Boost Context Module Implementation on s390x architecture

Aim

To implement the Context library on s390x architectures so that it can reach parity with other platforms.

Why is this project necessary?

There are many libraries which are directly depended on boost.context and are blocked from being available on the s390x . The absence of this module blocks the following modules:

- boost.coroutine
- hhvm
- kicad

The implementation of this project would clear this hurdle and thus would help directly and indirectly for at least 4 libraries/projects to be available on s390x. This could also help increase the community/users of the s390x.

Few Discussions - which tells the importance of this module

- <https://bugs.launchpad.net/ubuntu/+source/boost/+bug/1694926>

Abstract

In the following paragraph, I would like to explain the higher level diagram of boost.context works.

Fiber

boost.context uses the concept of **fiber**([Research Paper on fiber](#)). We can regard the term *fiber* to mean *user-space thread*.

A fiber is capable of the following things:

- Can be detached from the launching code
- One fiber can join another
- Can sleep until a specified time, or for a specified duration
- Multiple conceptually-independent fibers can run on the same kernel thread

Blocking a fiber implicitly transfers control to a fiber scheduler to dispatch some other ready-to-run fiber.

The key *difference between fibers and kernel threads* is that fibers use cooperative context switching, instead of preemptive time slicing.

Fiber context switching does **not** engage the kernel: it takes place entirely in user space. This permits a fiber implementation to switch context significantly faster than a thread context switch.

fcontext_t

boost.context() **depends heavily on fcontext_t** to deal with context switching. A *fcontext_t* provides the means to suspend the current execution path and to transfer execution control, thereby permitting another *fcontext_t* to run on the current thread. This state full transfer mechanism enables a *fcontext_t* to suspend execution from within nested functions and, later, to resume from where it was suspended.

`fcontext_t` is implemented as a structure in the source code. ([Refer here](#)). Each instance of `fcontext_t` represents a context (CPU registers and stack space). Together with its related functions `jump_fcontext()` and `make_fcontext()` it provides a execution control transfer mechanism similar interface like `ucontext_t`. `fcontext_t` and its functions are located in `boost::context` and the functions are declared as extern "C".

The following are the functions which work alongside `fcontext_t` to make context switching possible:

I. `jump_fcontext()`

Calling `jump_fcontext()` **invokes the context_function in a newly created context** complete with registers, flags and stack and instruction pointers.

The current context information (registers, flags and stack and instruction pointers) is saved and the original context information is restored.

Calling `jump_fcontext()` again resumes execution in the second context after saving the new state of the original context.


II. `make_fcontext()`

A new context supposed to execute a context-function (returning void and accepting void * as argument) **will be created on top of the stack** (at 16 byte boundary) by function `make_fcontext()`.

Different architectures have different stack structures and different ABI (*Application binary interface*) interfaces. It is not possible to implement the context switching on different architectures by a high level programming language, because - it won't be efficient nor would it allow us to use the special functions, an architecture might have. Hence it becomes necessary to write the executing code for the above functions in assembly.

The assembly codes for the above functions[`jump_fcontext()` and `make_fcontext()`] are stored in the directory [context/src/asm/](#). There are three assembly files for each of the different architectures which are implemented and optimized for the particular architecture.

The **naming convention** for each assembly file is:



< function > _ < arch > _ < ABI > _ < Binary format > _ < assembler > . S

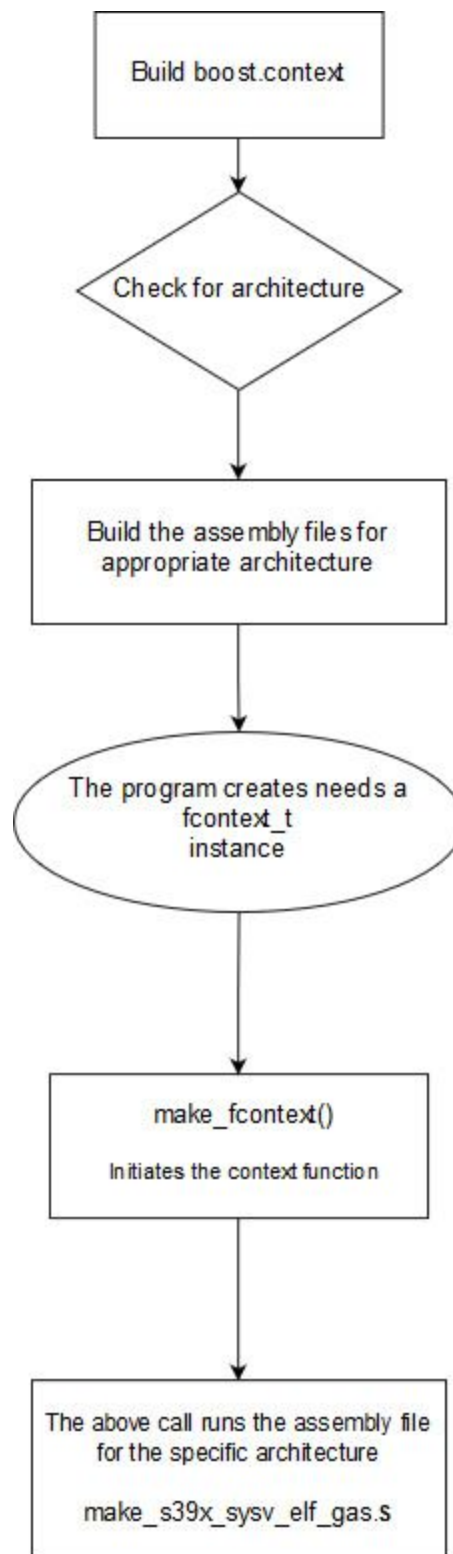
Eg: An example assembly file for s390x architecture would be: **jump_s39x_sysv_elf_gas.S**

Let's look at what the each assembly file do:

1. **jump_s390x_sysv_elf_gas.S** : context switch between two fibers/fiber_contexts
2. **make_s390x_sysv_elf_gas.S** : prepares stack for the first invocation
3. **ontop_s390x_sysv_elf_gas.S** : execute function on top of a fiber/fiber_context

The summary of the above text is that whenever a user calls `jump_context()` or `make_context()`, the `boost.context` library selects the appropriate assembly files and execute the instructions present in them.

The image in the next page, shows an example scenario of how `boost.context` works



Implementation

Before I start implementing the project, I would have to gather the knowledge regarding:

- The binary format
- Which ABI is used (ABI describes the calling convention)
- Information about the Stack frame layout

From my research I was able to find the answers to the above two questions.

- **The binary format** - ELF (since we are focusing on Linux)
- **The ABI** - System V (sysv)
- **Stack frame layout** - Yet to read about the s390x architecture.

Goals

1. Include s390x architecture in the boost's build property.
2. Implement **jump_s390x_sysv_elf_gas.S** assembly file.
3. Implement **make_s390x_sysv_elf_gas.S** assembly file.
4. Implement **ontop_s390x_sysv_elf_gas.S** assembly file
5. Test the working on a s390x hardware
6. Alternatively test if the libraries which were depended on context(coroutine,hvmm etc) are able to build
7. Document all the steps in the blog

Stretch Goals

Package the context module and the other modules which would build after successful implementation and send it upstream.

Planning

The implementation of the each Assembly files would be done in one sprint having three stages.

Stage 1: Planning

- I would like to take some time to understand how the assembly code works by reading the documentation and by looking at the implementation of these in existing architecture

Stage 2: Implementation

- Implement the assembly codes by following the standard coding practices
- Document the steps

Stage 3: Testing

- Test whether the particular assembly file works.
- Get feedback from mentor
- Implement the feedback

Timeline

I don't have any other commitments and I would be available throughout the summer except for a week, when I would be having my university finals. I prefer to follow the US schedule. According to the schedule, there would be about 45 days excluding the last week and weekends. I am willing to work for 40 hours a week. For the time, that I might lose during the one week of my finals, I would put in more time before and after my exams to compensate for the loss.

Also, my development methodology includes writing daily reports, where I would write about the work I do daily. This helps me keep track of my work and makes me more productive.

A tentative schedule of my plan is as follows.


Week	Deliverables
Community Bonding Period April 29 - May 17	<ul style="list-style-type: none">• Read Documentation for s390x• Read Documentation for

	boost.Context <ul style="list-style-type: none"> • Set up boost.context on system • Set up an account for IBM test cloud for testing
Week 1 - Week 2 (14 days) May 20 - June 1	<ul style="list-style-type: none"> • Set up the architecture in Jamfile.v2 • Implement <code>jump_s390x_sysv_elf_gas.S</code> • Document the process • Get feedback from the mentor
Week 3 - Week 4 (14 days) June 2 - June 15	<ul style="list-style-type: none"> • Implement <code>make_s390x_sysv_elf_gas.S</code> • Document the process • Get feedback from the mentor
Week 5 (7 days) June 15 - June 22 [Midterm Evaluation]	<ul style="list-style-type: none"> • Fixing up code for Midterm Evaluation • Document the process • Submission
Week 6 (7 days) June 23 - June 28	Would be having my college finals (Won't be able to work much)
Week 7 - Week 8(14 days) June 30 - July 13	<ul style="list-style-type: none"> • Implement the changes suggested in midterm evaluation • Implement <code>ontop_s39x_sysv_elf_gas.S</code> • Document the process • Get feedback from the mentor
Week 8 (7 days) July 14 - July 20	Buffer week - for any pending or extra work
Week 9 (Final Week) July 21 - July 27	<ul style="list-style-type: none"> • Clean up the documentation and run final tests
July 29 - August 2	Final Evaluations

Risk Management

If I get stuck on my project and my mentor is not around?

I would never let the unavailability of my mentor hinder my progress. There are many alternative ways that I can use to get help from. Such as:

- 
1. Taking help from the community members through:
 - a. Mailing List
 - b. Slack Channel
 - c. Community Forum
 2. Taking help from the boost.context maintainers
 3. Reading relevant articles, resources and figuring out on my own
 4. Taking help from those who might have already worked on similar project

Keeping the community informed of my progress

Open Mainframe is a large-scale organization. It also supports many ways of communication where ideas/information can be shared/discussed and these platforms have large outreach. Thus, I would be taking advantage of these platforms to inform the community of my progress and also discuss any problems or questions I might have over the course of the project.

The different ways of communications are:

1. Interacting with mentors
2. Interacting with community members through:
 - a. Mailing list
 - b. Community Forum
 - c. Creating issues on Github
 - d. Slack Channel