LabLua (Project Proposal GSoC'18)

Rewrite/update Lua Bindings for Libgit2.

March 27th 2018

Satyendra Kumar Banjare
(IIT Roorkee Electrical Engineering Sophomore)
Mail to : satyendrabanjare99@gmail.com
Blog at : https://satyendrabanjare.github.io/
Github : https://github.com/SatyendraBanjare
Mobile : +91-9893415674 , +91-9685186526
Location: Roorkee, Uttarakhand, India
Time Zone: IST (UTC +5:30)

Basics

1. What is your preferred e-mail address?
-> satyendrabanjare99@gmail.com

2. What is your web page / blog / github?
-> Github : https://github.com/SatyendraBanjare/
Blog at : https://satyendrabanjare.github.io/

3. What is your academic background?
-> I am a second year undergraduate Electrical Engineering Student at IIT Roorkee enrolled in a 4 year B. tech course.

4. What other time commitments, such as school work, another job (GSoC is a full-time activity!), or planned vacations will you have during the period of GSoC?
-> My Semester exams end on may 10th after which I am completely free for the period May 14th to July 15. I do not have any other commitments. I can easily give up to 40-50 hours per week during this period.

I have my next semester classes starting from July 16th so I might spend a little bit less time during weekdays but I promise to make it up during weekends. Overall I'll be able to give 35-40 hours per week.

Experience

1. What programming languages are you fluent in? Which tools do you normally use for development?

-> I am fluent in python, c , c++, java,javascript and nodejs. I am also familiar with haskell and ruby(often require googling to get things done). As far as my expertise in Lua is considered, I will mark myself as more of an intermediate learner.

I use ubuntu 16.04 (xenial) system with virtualbox installed so as to test my code on other operating systems. I am editor-terminal guy and I prefer to edit code using sublime text and compile run from terminal. I frequently use vim editor for making short changes. I do not prefer using IDE too much unless it is android programming or c# development.

I use Git as my version control system. I am experienced with vagrant and frequently use different vagrant boxes to test out code.

I use django and node in most of my web development projects.

I am very familiar with package generation using make/cmake(for c and c++), cabal (for haskell) and now learned writing rockspec for lua related projects.

2. Are you familiar with the Lua programming language? Have you developed any project using Lua?

-> Yes, I am familiar with Lua Programming. I know about the fundamentals involving tables, metatables, etc. and over the period, I've also learnt about C-Lua API. I have developed a mobile game using corona framework which is based in lua. This was just a basic game I developed taking references from their example source code. Its apk is available at: https://drive.google.com/open?id=1FOOoGaskWgfTsODpoFjAAQuLuYiLhGmy and corresponding source files at https://drive.google.com/open?id=1FOOoGaskWgfTsODpoFjAAQuLuYiLhGmy and https://github.com/SatyendraBanjare/fishy_balls
(It's a simple game in which you need to prevent red ball to hit obstacles for 60 seconds by touching and moving it).

Apart from this I have developed a prototype of the updated **luagit2** library available at https://github.com/SatyendraBanjare/luagit2 . I have written lua bindings for few common libgit2 library's function. I have also written rockspec file which can be used to build the required shared object. I have also provided example usage in it .

3. Have you developed software in a team environment before?

-> Yes, I have a good experience in developing software in a team environment. I am part of ArIES (Artificial Intelligence and Electronics Section) IIT Roorkee which is a group of students constantly striving to innovate and foster technical activities at IIT Roorkee. I have worked on some electronics and software development projects with them. Apart from this I frequently take part in hackathons in a group of 2-4 people.

4. What kinds of projects/software have you worked on previously? (anything larger than a class project: academic research, internships, freelance, hobby projects, etc.)

->

- Project-HomeAutomation
  https://github.com/SatyendraBanjare/Project-HomeAutomation
  This is a simple home automation project me my other team members created while trying our hands on working with the electronic components to control them via mobile/web interface. We created a django amputity server and deployed on heroku and created rest API via Django Rest Framework. We took data from server on a single Raspberry Pi and later ran electronic components via Xbee nodes.

- ProSearch https://github.com/SatyendraBanjare/ProSearch
  This is the submission project we created as a part of rajasthan hackathon 3.0. It Basically generates questions out of input text for user to practice and improve their memory skills.

5. Are you (or have you been) involved with any open source development project? If so, briefly describe the project and the scope of your involvement.
-> Yes, I was contributing to Zulip, here are a list of prs that I tried to make https://github.com/zulip/zulip/pulls/SatyendraBanjare and Issues created https://github.com/zulip/zulip/issues/created_by/SatyendraBanjare .

-> I also fixed one issue in Haskel chat app haskell-chat-server-example. Code available in here. https://github.com/SatyendraBanjare/Chat-App

Project

1. Did you select a project from our list? If yes, which project did you select? Why did you choose this project? If you are proposing a project, give a description of your proposal, including the expected results.

-> Yes, I am very much excited to work on this project : Update/Rewriting lua bindings to libgit2. I came to know about the libgit2 library though this project idea and instantly fall in love with the library. I used the pygit2 initially to understand the code flow then I tried with their (Libgit2's) given c examples. After reading extensively about Lua-C API, I wrote an initial prototype model of the updated luagit2 library. With libraries of other languages so popular, I want to make luagit2 as the most user friendly library for libgit2 binding as possible.
I am also developing interest in the main libgit library and will try to contribute in that in future.

Overview

This Project aims at creating Lua Bindings to the updated libgit2 library which is a pure C implementation of Git Core methods. Therefore we should be able to call those C functions of library from lua. Now since there is a large set of final exposed

functions available to a user via their API as mentioned in here https://libgit2.github.com/libgit2/#HEAD , it will not be very useful to generate bindings for each one of them. Some of the api functions are helper methods and are very basic in their implementation and are also used inside some other more top level functions. Thus we should create bindings for those essential functions as required.
Currently the libgit2 has bindings supported in various languages such as ruby, csharp, python etc. A general observation of those binding libraries tells us the same.

The Project will cover these following things :

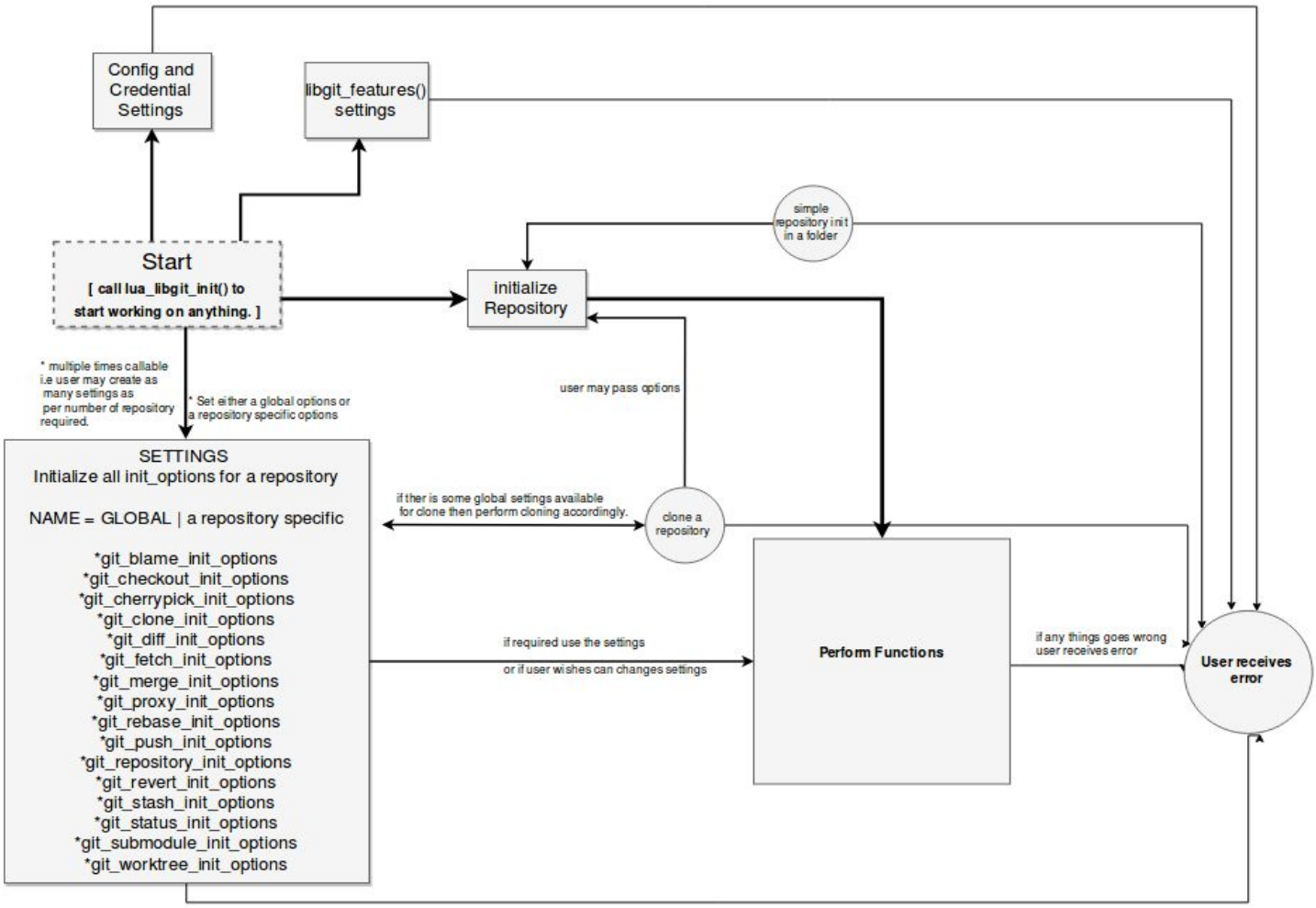
- Write Bindings - This part will cover on creating links so that users can call libgit2 API functions from lua.
- Error handling - This is the most important part according to me. Though libgit2 library has its style of error handling via error codes so we may prompt user if error code is resulted. I also wish to stress on what happens in my prototype version is that if something goes wrong, system goes in segmentation fault. Example cases include passing wrong path or passing integer argument that exceeds the limit of unit_16 type. Though a C user will be careful but lua has only number data type so need to make the user aware of such similar cases. I'll try to cover as much possible error handling for binded functions as possible.
- Write Tests - This part will cover on writing tests for written binding functions. Since this project involves both of the C and Lua languages in its making , I'll try to write tests for both the c part as well as lua part (basically generate lua scripts and run tests on them) .
- Documentation and Examples - This part focuses on writing completing documentation and example lua scripts for the users to use and learn.
- Publishing final module to luarocks and documentation to readthedocs.io .

Details

1. WORKING / ARCHITECTURE -

The Graphical representation of the working of this project according to me is given below.

- The user starts by importing the library and lua_libgit_init() function to in order to set up global state and threading. (lua equivalent of libgit_init()).
- The User can also set the libgit_opts() settings. I will create these initializing functions separately as it will affect all the working of the library functions.
- The User can also set the config and cred settings. I will also create this separately as these will also affect all the working of library.
- The user then proceeds with creating an instance of setting under name GLOBAL or REPOSITORY_NAME. User should pass all the To-Be-set variables in form of table. Back in the C side we can check for passed values and do change the settings accordingly . The settings will have libgit2_specific_default values by default.
- The User can then proceed with creating an instance of repository.
    - Either by simply instantiating by passing the repository's path.
    - Or by initializing a new repository using lua_repository_init()
    - Or by cloning a repository from github.
- The User can then call the functions to start working.



2. To be binded functions -

Taking a reference point of rugged bindings to libgit2 library, these are the set of important functions that I plan to write bindings for initially.

- Repository : This module will cover functions related to initializing a repository.
  [git_repository__cleanup|git_repository_config | git_repository_config_snapshot | git_repository_detach_head | git_repository_discover | git_repository_fetchhead_foreach

git_repository_free | git_repository_get_namespace |
git_repository_head |git_repository_head_detached
git_repository_head_for_worktree | git_repository_ident
git_repository_index| git_repository_init |git_repository_init_ext
git_repository_init_init_options| git_repository_is_bare |
git_repository_is_empty | git_repository_is_shallow |
git_repository_is_worktree| git_repository_item_path |
git_repository_mergehead_foreach | git_repository_message |
git_repository_message_remove git_repository_new |
git_repository_odb | git_repository_open |git_repository_open_bare
git_repository_open_ext |git_repository_open_from_worktree
git_repository_path |git_repository_refdb
git_repository_reinit_filesystem |git_repository_set_bare
git_repository_set_config |git_repository_set_head
git_repository_set_head_detached |
git_repository_set_head_detached_from_annotated |
git_repository_set_ident | git_repository_set_index |
git_repository_set_namespace |git_repository_set_odb|
git_repository_set_refdb | git_repository_set_workdir |
git_repository_state | git_repository_state_cleanup |
git_repository_submodule_cache_all]

- Cred : This module will cover functions related to changing credentials used while working and also setting namespace related settings. [
  git_cred_default_new | git_cred_free |git_cred_has_username
  git_cred_ssh_custom_new | git_cred_ssh_interactive_new
  git_cred_ssh_key_from_agent | git_cred_ssh_key_memory_new
  git_cred_ssh_key_new | git_cred_username_new | git_cred_userpass
  |git_cred_userpass_plaintext_new]

- Config : This module will cover functions related to changing configuration settings.

- Clone : This module will cover functions related to cloning a repository. [git_clone |git_clone_init_options]

- Libgit specific : [git_libgit2_features |git_libgit2_init |git_libgit2_opts git_libgit2_shutdown |git_libgit2_version]

- Blame : This module will cover functions related to git-blaming a file. [git_blame_buffer |git_blame_file |git_blame_free | git_blame_get_hunk_byindex |git_blame_get_hunk_byline | git_blame_get_hunk_count |git_blame_init_options]

- Commit : The module will cover functions related to commiting changes. [git_commit_amend | git_commit_author |git_commit_body | git_commit_committer |git_commit_create |git_commit_create_buffer | git_commit_create_from_callback |git_commit_create_from_ids | git_commit_create_v |git_commit_create_with_signature | git_commit_dup |git_commit_extract_signature |git_commit_free | git_commit_header_field |git_commit_id |git_commit_lookup git_commit_lookup_prefix |git_commit_message | git_commit_message_encoding |git_commit_message_raw | git_commit_nth_gen_ancestor |git_commit_owner |git_commit_parent | git_commit_parent_id |git_commit_parentcount | git_commit_raw_header |git_commit_summary |git_commit_time git_commit_time_offset |git_commit_tree | git_commit_tree_id]
- Branch : This module will cover functions related to git-branch related functions. [git_branch_create |git_branch_create_from_annotated | git_branch_delete |git_branch_is_checked_out |git_branch_is_head | git_branch_iterator_free |git_branch_iterator_new |git_branch_lookup | git_branch_move |git_branch_name |git_branch_next | git_branch_set_upstream |git_branch_upstream]

- Diff : This module will cover functions related to git-diff. [git_diff_blob_to_buffer|git_diff_blobs |git_diff_buffers | git_diff_commit_as_email |git_diff_find_init_options | git_diff_find_similar

|git_diff_foreach |git_diff_format_email |
git_diff_format_email_init_options |git_diff_free |
git_diff_from_buffer | git_diff_get_delta |git_diff_get_perfdata |
git_diff_get_stats |git_diff_index_to_index | git_diff_index_to_workdir |
git_diff_init_options | git_diff_is_sorted_icase |git_diff_merge |
git_diff_num_deltas |git_diff_num_deltas_of_type | git_diff_patchid |
git_diff_patchid_init_options |git_diff_print | git_diff_print_callback__to_buf
git_diff_print_callback__to_file_handle |git_diff_stats_deletions |
git_diff_stats_files_changed |git_diff_stats_free |
git_diff_stats_insertions |git_diff_stats_to_buf |git_diff_status_char |
git_diff_to_buf | git_diff_tree_to_index |git_diff_tree_to_tree |
git_diff_tree_to_workdir |git_diff_tree_to_workdir_with_index]

- Rebase : These functions will help rebasing head. [git_rebase_abort |
git_rebase_commit | git_rebase_finish | git_rebase_free |git_rebase_init
|git_rebase_init_options | git_rebase_inmemory_index |git_rebase_next
|git_rebase_open | git_rebase_operation_byindex
|git_rebase_operation_current |
git_rebase_operation_entrycount]

- Reference : This module will cover functions related to reference related functions. [git_reference__alloc |git_reference__alloc_symbolic |
git_reference_cmp |git_reference_create | git_reference_create_matching |
git_reference_delete |git_reference_dup |git_reference_dwim |
git_reference_ensure_log |git_reference_foreach |
git_reference_foreach_glob |git_reference_foreach_name |
git_reference_free |git_reference_has_log |git_reference_is_branch |
git_reference_is_note |git_reference_is_remote |git_reference_is_tag |
git_reference_is_valid_name |git_reference_iterator_free |
git_reference_iterator_glob_new |git_reference_iterator_new |
git_reference_list |git_reference_lookup |git_reference_name |
git_reference_name_to_id |git_reference_next |git_reference_next_name |
git_reference_normalize_name |git_reference_owner |git_reference_peel |
git_reference_remove |git_reference_rename |git_reference_resolve |

git_reference_set_target|git_reference_shorthand|git_reference_symbolic_create |git_reference_symbolic_create_matching|
git_reference_symbolic_set_target |git_reference_symbolic_target |
git_reference_target |git_reference_target_peel |git_reference_type]

- Tag : This module will cover functions related to tag functions [
git_tag_annotation_create |git_tag_create |git_tag_create_frombuffer
|git_tag_create_lightweight |git_tag_delete |git_tag_dup| git_tag_foreach |
git_tag_free |git_tag_id |git_tag_list |git_tag_list_match |git_tag_lookup |
git_tag_lookup_prefix | git_tag_message |git_tag_name |git_tag_owner |
git_tag_peel |git_tag_tagger |git_tag_target |git_tag_target_id|
git_tag_target_type]

- Blob : This module will cover functions related to Blob related functions. [
git_blob_create_frombuffer |git_blob_create_fromdisk|
git_blob_create_fromstream |git_blob_create_fromstream_commit |
git_blob_create_fromworkdir |git_blob_dup |git_blob_filtered_content |
git_blob_free |git_blob_id |git_blob_is_binary |git_blob_lookup |
git_blob_lookup_prefix |git_blob_owner |git_blob_rawcontent |
git_blob_rawsize]

- Object : Will cover functions related to object functions set. [
git_object__size |git_object_dup |git_object_free |git_object_id
git_object_lookup |git_object_lookup_bypath |git_object_lookup_prefix
git_object_owner |git_object_peel |git_object_short_id
git_object_string2type |git_object_type |git_object_type2string
git_object_typeisloose]

- OpenSSL locks : It'll will help in triggering openssl locks. [
git_openssl_set_locking]

- Push : git_push_init_options

- Checkout : [git_checkout_head |git_checkout_index | git_checkout_init_options |git_checkout_tree]

- CherryPick : [git_cherrypick |git_cherrypick_commit | |git_cherrypick_init_options]

MileStones Decided :

1. Complete bindings for config and initializing settings related functions.
2. Complete bindings for usage related functions.
3. Implement Error handling related logic.
4. Write C Unit tests.
5. Write Lua Unit Tests.
6. Write Integration Tests.
7. Generate the rockspec file and publish the library.
8. Entirely document the code and publish to readthedocs.io .

Other Libraries required:

- Lunit to write lua unit tests and final integration tests.
- Lua-C API to write bindings.

Schedule

The Request for Proposal timeline is as follows:

S.no	Timeline	Area of Work	Work	Milestone Reached
1.	April 23 - May 13	Community Bonding	Get in touch with community. Discuss and improving the design of the Library. Through experience I have learnt that designing a project is a very important part of any software project. I will also study some other Libraries(ruby, python)	none

			in detail.	
2.	May 14- May 20	Write bindings and error handling features	Will work on modules : cred, config, Libgit specific	1
3.	May 21- May 27	Write bindings and error handling features	Will work on modules : repository, Reference, Commit, clone	
4.	May 28- June 3	Write bindings and error handling features	Will work on modules : Blob, Branch, blame ,diff,	
5.	June 4- June 10	Write Bindings and error handling features	Will work on modules : Reference, tag, SSL, Push, Object, Cherry Pick, Checkout	2
6.	June 11 - June 17	Error handling + Writing Tests	Will write tests for modules : cred, config, Libgit specific, Object	
7.	June 18- June 24	Error Handling + Writing Tests	Will write tests for : repository, Reference, Commit, clone, tag	
8.	June 25 - July 1	Error Handling + Writing Tests	Will write tests for modules : Blob, Branch, blame ,diff, Reference, Cherry Pick, Checkout	3,4,5
9.	July 2 - July 8	Writing Tests	Will work on writing Lua scripts based on newly created bindings and then write tests for them.	6
10.	July 9 - July 15	Publish + Documentation	Write Rockspec file and Make files both. Publish to luarocks. Start writing documentation.	7
11.	July 16- July 22	Documentation	Write function specific documentation.	

12.	July 23 - July 29	Documentation	Complete with Documentation	
13.	July 30 - Aug 5	Write Examples	Write Example usages for People to learn	
14.	Aug 6 - Aug 14	Write Examples	Complete with writing examples and publish the docs to readthedocs.io	8

Expected Outputs

1. What will be showable one month into the project?
-> As can be seen from my timeline, I'll try to achieve Milestones 1,2 . Bindings to all the functions mentioned should be available for user to try.

2. What will be showable two months into the project?
-> As can be seen, I'll try to cover milestones 3,4,5,6 bindings to functions should be available. Completely written tests should be available now. Library should be having good error handling available so that user at least gets to know where he is going wrong.

GSoC

1. Have you participated as a student in GSoC before? If so, How many times, which year, which project?
-> No, This is my first experience with GSoC. My motivation this year is to get exposed to open source community and start contributing to some great projects. This project is the perfect combination of both for me.

2. Have you applied but were not selected? When?

-> No.

3. Did you apply this year to any other organizations?

-> No.

Extra | After GSoC

- I will write bi-weekly blog posts about the development process which will be available at my blog page link.
- If I cover this early on time, I'll try to make bindings for more functions and add them to library and accordingly add tests and documentation and example usages.
- Contribute more to lua community.