C implementation of command server API

C library to communicate with mercurial command server.

Abstract

The purpose of this project is to create a C library that will communicate with mercurial command server. Until now there are some libraries written in other languages like Python, Java, Php, Ruby, etc. Now it's time to create a library for C language.
There is no code written in C for this library but the existence of other languages libraries will be a good point to start.
At the end of the summer there will be an API that will be used by all C and C++ users.

Detail Description

The goal of the command server is to facilitate the creation of wrapper libraries.
Those libraries are approachable by a variety of languages, like Python, Java, Ruby, etc. One of the common languages which is not in this list is C. With the creating of C library the mercurial will extend his horizons for the C users. Thus, it will be easier for C programmers to integrate their projects with mercurial.

The purpose of Command server is to communicate over a pipe and to eliminate the per-command start-up overhead. Libraries can then encapsulate the command generation and parsing to present a language-appropriate API to these commands. This strategy is similar to how applications typically communicate with SQL servers.

At the end of the summer there will be a C library that will communicate with command server. It will be well documented, with lots of tests and examples. The code will be on a subversion server, maybe on bitbucket and it will be free for everyone.

Also a wiki page from where users will learn how to install and how to use the library will be available.

Technical Details

Like I said before, I will use Bitbucket subversion because I am familiar with it and because it's a free service that offers mercurial hosting.
It will be easier for me to work with Bitbucket and it will be in the same place with other libraries.
All the commits will be on mercurial standards:

- first line of commit message is of the form "**topic: uncapitalized, no trailing period**"

In this way, the code will be closer to mercurial main project and the mentors will watch my activity and they will supervise me faster.
I will try to use a coding style that will be near to mercurial coding style (no foo_bar variable names!).
http://mercurial.selenic.com/wiki/CodingStyle

The language that I will use is going to be C, not C++. In this way everyone will use this API library, the C and the C++ users. Another reason is that it will be easier for mentors to help me. There aren't so many annoying warnings and there will be easier for me to make debugging on code.

The tools that I will use for the debugging will be: valgrind, gdb, lots of prints on the code and other helpful tools.
http://techblog.rosedu.org/valgrind-introduction.html
http://techblog.rosedu.org/gdb-a-basic-workflow.html

The unit tests are one of the many ways that prove the concept. There must be a way to test the code and to help users to use the API. With those tests and some examples the users will get familiar faster with the API.
http://check.sourceforge.net/
http://check.sourceforge.net/doc/check_html/check_2.html#SEC3

There will be lots of examples and unit tests, but those are not enough. There must be a strong documented code. And this thing will be done with doxygen tool. The explanations will be in the code, but there must be a way to see those explains from the outside of the code.
In this case the user will use the API faster and he will understand the workflow of the library.
http://www.stack.nl/~dimitri/doxygen/

I will communicate with server throw sockets about the implementation and establish a connection and through this connection I will communicate with command server.
http://www.linuxhowtos.org/C_C++/socket.htm
http://shoe.bocks.com/net/

I will create the specific command, runcommand and getencoding and through those commands I will implement the mercurial commands.

One of the inspiration points will be the python hglib.
http://www.selenic.com/repo/python-hglib/file/ca5f8f43e585/

I will watch the java and the Ruby implementations too and the cHg, which I think that in return, will help me with some ideas about the C coding style and stuff.

Timeline

Now- 27 May: I continue contributing to mercurial main project.
27 May – 17 June: I will speak with the mentors about the project and I will start to fix the workstation for the gsoc project. In this time I will have school session and I will study for my final exams. This means that I will not be full time on the irc.
18 June – 23 June: With the official GSoC start I will begin real coding and making my project work. I will create the socket connection and I will start playing with messages. I will make sure that the connection will be established well and I will test it.
23 June - 1 July: I will implement the runcommand function and start testing the code.
1 July – 15 June: I will implement the getenconding function and start the implementation for the rest of the functions
15 June - 29 June: I will start the doxygen documentation of the code. In the meantime I will start focusing on the midterm.

29 June – 12 August: I will finish the implementation of all API functions.
12 August – 26 August: I will focus on the tests and examples for the API
26 August – 9 September: I will finish the code documentation
9 September – 16 September: I will refactor the code and try to prepare for the final evaluation
16 September – 23 September: I will reevaluate the entire activity.

Every week I will write a blog post that will show my activity.
I will be in between 8 and 24 hours per day on the IRC and I will communicate with mercurial members.
One day per week I will take a break from the project and solve mercurial issues.

Why Me

My name is Iulian Stana, I'm from Bucharest, Romania and I study at The Politehnica University of Bucharest, the Faculty of Automatic Control and Computer Science. I'm a third year student with lots of free time that I am willing to spend studying and working on this project.

I have lots of skills in computer science, basically in C, Python, Java and knowledge in network programming in C.

Last summer I participated at Rosedu Summer of Code, a program just like GSoC made by an organization from my university. There I was successful in creating a chat client for WoUSO, a browser game, written in Python&Django where I used javascript with ajax and socketsIO[1].
I am confident that I can make a commitment to work on GSoC full-time, and I think that I can achieve this commitment, because I am used to this kind of schedule.

I can't say that I had written any code yet, but I started making lots of researching and I made some issues from bugzilla page for the mercurial. With those issues I managed to understand the code and to see how the community works.

In my past I implemented several times client-server application and understood the sockets from C language. I think that is a good opportunity to contribute to a real project and to build a client API for mercurial.

Contact Information

- e-mail: julian.stana@gmail.com
- irc nick: iulians
- IM: google talk (gmail address)