Advanced C++ Project

Babel

Koalab koala@epitech.eu

Abstract: The project consists in a client/server architecture voice over IP application, similar to ***Skype*** *or* ***TeamSpeak****.*

Table des matières

Chapitre I

Generalities

I.1 Platforms

The project MUST be OS independent. It has to compile and run in a similar manner on at least one `Unix` system and one `Windows` system.

> `You can use the Fedora Linux and Microsoft Windows 7 present on your rack/laptop, but feel free to test other platforms, like FreeBSD for instance.`

I.2 Protocol

The `Babel` project aims to create a `SIP`-like Voice Over IP (VOIP) protocol. It MUST be usable over the Internet (no multicast or anything LAN specific).

To test your protocol you MUST write a server and a client implementation.

The protocol MUST be a client/server protocol but voice transport MUST be client to client (the server can have a proxy mode for conference calls or NAT-ed clients).

> `The protocol MUST be a binary protocol, not a text one. For instance, TCP is a binary protocol and HTTP is a text one.`

I.3 Libraries

Any non-explicitly allowed library is explictly forbidden. However :

- You are NOT allowed to use any SIP or other VOIP library.
- You MUST use the PortAudio library for anything sound related (http://www.portaudio.com/).
- You MUST use Opus for the compression codec (http://www.opus-codec.org/).
- You MUST use Qt (http://trolltech.com/) for the client's graphical user interface.
- You are allowed and encouraged to use Qt library on the client side.
- You are NOT allowed to use any Qt library on the server side.
- You are allowed and encouraged to use BOOST C++ libraries on the server side (http://www.boost.org).
- You are NOT allowed to use any BOOST C++ library on the client side.
- POCO C++ libraries (http://pocoproject.org/) are very cool but NOT allowed.
- PortAudio and Opus are C libraries, so you MUST create your own abstractions.

Have a look at BOOST Asio at http://www.boost.org/doc/libs/1_40_0/doc/html/boost_asio.html for networking on the server.

A good idea may be to implement your network abstraction in terms of BOOST Asio on the server side and in term of Qt Network on the client side. Anyway, your network abstraction could also be hand written on both side, it won't change your grade.

I.4 Versioning

You SHOULD use code versioning during the whole project.

A clever use of your versioning system WILL be checked during defense ! Use precise commit logs ! Don't mess up with your repositories ! In contrast with your second year, we take your usage of the versioning system into careful consideration.

Chapitre II

Mandatory part

II.1 Contents

The following items are mandatory at the end of the project :

- A **printed** version of your binary protocol for your communications (mandatory for **BOTH** follow-ups !)
- A **printed** fully UML compliant class diagram for **BOTH** client and server (mandatory for **BOTH** follow-ups !)
- A network abstraction implemented in terms of `Boost` or hand made server side, and implemented in terms of `QT` Network or hand made client side.
- `QT` GUI for the client
- `PortAudio` and a C++ abstraction for any sound-related code
- Transmit sound compressed using `Opus`, the library behind a C++ abstraction as well.
- Contact list
- Make a call
- Hang up

`To help you self rating your abstractions, think that way: Could you change your implementation without impacting the whole program ? For instance, could you use a different audio library and have only to change your sound abstraction's implementation ? Another instance : Could you get rid of Boost::Asio and still have to only change your network abstraction's implementation ? If the answer is "no", your abstraction is brocken.`

II.2 Steps

> It is important that **YOU** defend **YOUR** product. Koalas will rate only what you show them ! Organize your follow-ups and defense, share speaking time,...Long story short; be PREPARED ! Don't come as a tourist. It will be the first time for you that you'll pass such serious follow-ups and defense !

> Remember that you are rated individually ! This is not bullshit, you've been warned.

> Organize your team and affect roles to everyone. We **WILL** rate every aspect of your team, so don't play it "tech1 style".

- Conception follow-up : You must show and defend your conception during this follow-up. The two main points that will be heavilly discussed are your binary protocol and the design and conception of your server and client. Focus on designing clever abstractions for network, sound, GUI,...Be smart and creative ! The class diagram and protocol's description are MANDATORY and MUST be printed. Other documents such as use cases, sequence diagram and anything you consider relevant will be very appreciated and will improve our consideration of your seriousness.
- Implementation follow-up : We will check during this follow up the quality of your abstractions' implementations and their basic functionnalities. A functionnal prototype of Babel is expected, including (at least) calling a contact using your GUI. Focus on the accuracy between your conception and your implementation !
- Final defense : Last step of the project, we will check the achievement of your project and its functionnalities. We prefer small and fully functionnal Babels to huge but half-finished ones.

Although they are mandatory, the grades for the two follow-ups won't count in your final Babel grade. However, they are a great indicator for you of what we think about your work. Consider these follow-ups a privilege, not a punishment. Be serious and work hard, it'll be worth it.

II.3 Teams

For the first time you MUST work in team with one other group. As a consequence, you MUST team-up with one other `Babel` group and share a commonly designed binary protocol for your communications.

Most of the points available during the final defense will concern you ability to use your server with your team group client and your client with your team group server. We will of course test `Linux` server with `Windows` client, and vice versa.

It is VERY important that you register on the same time slot for the final defense or we WON'T be able to test your work. Don't mess-up. This is not relevant for the follow-ups though.

As you may have already understood :

- If one team fails, both fail.
- If one team is late or missing, both fail.
- If one team doesn't support a test, both fail.
- Both teams are responsible for the other team.
- Your final grade is not shared, but it will be heavily impacted by the behaviour of the other team.

II.4 Bonus

Bonuses will **NOT** count unless you have a working mandatory part. Here are a few ideas :

- Text chat (maybe 1 point, but we don't really care)
- Record a call and play it back (2 points)
- Answering machine (2 points)
- Conference call (3 points)
- Video (3 points)

If you implement a plug-in system and add the bonuses as plug-ins you will get up to 3 extra points, and a free coffee from your local person in charge.

Chapitre III

General setpoints

You are (more or less) free to do the implementation you want. However, here are a few restrictions.

- The only allowed functions from the `libc` are the ones that wrap system calls (and don't have C++ equivalents !)
- Any solution to a problem MUST be object oriented.
- Any not explicitly allowed library is explicitly forbidden.
- Any value passed by copy instead of reference or pointer MUST be justified, otherwise you'll lose points.
- Any non-`const` value passed as parameter MUST be justified, otherwise you'll lose points.
- Any member function or method that does not modify the current instance not `const` MUST be justified, otherwise you'll lose points.
- Koalas don't use any `C++` norm. However, any code that we deem unreadable or ugly can be arbitrarily sanctioned. Be rigorous !
- Any conditional branching longer than `(if ... else if ... else ...)` is FORBIDDEN ! Factorize !
- Keep an eye on this subject regularly, it could be modified.
- We pay a great attention to our subjects, if you run into typos, spelling mistakes or inconsistencies, please contact us at `koala@epitech.eu` so we can correct it.
- You can contact the authors by mail, see the header.
- The intranet's forum will contain informations and answers to your questions. Be sure the answer to you question is not already there before contacting the authors.

Chapitre IV

Delivery setpoints

You MUST deliver your project on the BLIH repository made available for you by the Bocal. Your repository's name MUST be cpp_babel.

We will ignore any commit which happened after the project's end. Complaining that "it wasn't 11 :42 pm on my watch" is useless. The only time that counts for us is Epitech's time.

Murphy's law corollary : "If you deliver your work during the last hour, something's gonna go wrong.".

Only the code on your repository will be evaluated during the defense.
Good luck !