Subject: Proposal: Transition from SVN to GIT within 2020.1 release cycle
Posted by Klugier on Fri, 01 Nov 2019 13:17:30 GMT
View Forum Message <> Reply to Message

Hello,

As you may know U++ main version control system is SVN. We have only GitHub mirror available under https://github.com/ultimatepp/mirror/commits/master. It means that we can not use some powerful tools related to Git workflow. The main problem here is lack of Pull Request mechanism. Instead of using well defined process, we apply improvements directly mentioned on the forum or delivered within .diff files. The problem with that approach is that code review are extremely hard. Effectively, it scares away developers from making biggest contribution than several lines. Moreover, even if somebody made big change, it discourages Upp maintainers from applying this change especially when there are some comments related to the code. Personally, I remembered when I worked closely with omari on Macro Manager for TheIDE. It took some time to polish the code via discussing it on redmine... Moreover, I am pretty sure that we will be able to attract more developers to U++ itself, when we will allow for less painful and transparent contribution. At the end, I would like to add that with PR mechanism we are fully protected by unintentional changes like we have with current SVN privileges base approach.

Sometime ago Mirek mentioned that one of the problem with Git transition is the size of the Repository. In my opinion and with my current knowledge about the Git, it can be easily avoided. The main idea for that is to split main U++ repository into sub-modules like uppsr, examples, references etc. Each sub module is a small piece that doesn't weigh too much in compared with whole repository and it can be used separately. For example if you dwonload U++ you can replace your local uppsrc within the need of downloading full repository.

There is a one problem with submodules. It must be updated manually on the main repository level. However, this task can be easily automated by the script. For example we can launch script once per day that will update submodules and made the commit to the main repository. Then we will be one hundred percent sure that submodules doesn't live own life.

When we will adopt git then we can made some other improvements to the whole U++ world. The main idea I have in this area is remote package concept executed by the link to the github repository. The similar approched is used in GoLang. So, instead of copying packages from web you simply create package that points to some remote localization. Let's take into consideration upp-ccomponents developed by Oblivion. For example I would like to use MessageCtrl, so all I need to do is to add remote package that contains only link to the repository. The latest sources are downloaded automatically and you can use it directly in your project. Here is the example of .upp file modification:

```
description "A passive message ctrl and manager for U++\377";

uses
    Core,
    CtrlLib,
    https://github.com/ismail-yilmaz/terminal-ctrl.git; // <- Some modification can be saved here like branch used (master by default)
                                                        //    for example stable-11-19 etc. or include prefix to avoid
```

colissions.

file
MessageCtrl.h,
MessageCtrl.cpp,
MessageCtrl.iml,
Docs readonly separator,
src.tpp,
Info readonly separator,
ReadMe,
Copying;

The examplary usage of remote package would be:

#include <Remote/TerminalCtrl/TerminalCtrl.h> // <- All TerminalCtrl headers all avaliable from this place.

The git will be required for projects that will use above mechanism. So, the bare Upp shouldn't be affected by this dependency. However, all https requests will require additional OpenSSL dependency for TheIDE and UMK.

The above proposition aims to strengthen the whole U++ framework and increase it's popularity within developers. I think this changes should be incorporated as soon as possible and I think 2020.1 release cycle is the good moment to do them. Thanks!

Additional resources:
- https://github.com/golang/example/blob/master/hello/hello.go - usage of GitHub repo import in GoLang

Sincerely,
Klugier
