CPPL1 : TD 0 : Qt Creator

Nicolas Vansteenkiste Romain Absil Jonas Beleho *
(Esi – He2b)

Année académique 2017 – 2018

Ce TD décrit quelques manipulations récurrentes lors de l'utilisation de l'environnement de développement intégré Qt Creator.

Table des matières

*Et aussi, lors des années passées : Monica Bastreghi, Stéphan Monbaliu, Anne Rousseau et Moussa Wahid.

1 Portabilité

Les binaires exécutables ou objet produits à partir de codes C ou C++ ne sont pas portables d'un système d'exploitation à l'autre, voire d'un compilateur à l'autre sur un même système d'exploitation. Par contre, les sources peuvent être portables. Pour ce faire, il faut, au moins :

— respecter la casse [1] dans les noms de fichiers [2] ;

— utiliser le séparateur « / » [3] entre les répertoires dans les chemins d'accès lors des inclusions de fichiers d'en-têtes.

2 Configuration de Qt Creator

Commençons par configurer Qt Creator [4] pour que les chemins vers les bibliothèques statiques [5] fournies au long des TD et des évaluations soient correctement résolus.

Démarrez Qt Creator [6], puis :

(a) dans le menu *Tools*, choisissez *Options...*

(b) sélectionnez *Build & Run* dans le panneau de gauche ;

(c) sous l'onglet *General*, remplacez ce qui se trouve à droite de l'étiquette *Default build directory* située tout en bas par un simple point : .

(d) cliquez *OK*.

1. https://fr.wikipedia.org/wiki/Casse_(typographie)
2. https://stackoverflow.com/q/1951969
3. https://stackoverflow.com/q/5790202
4. https://doc.qt.io/qtcreator
5. https://en.wikipedia.org/wiki/Static_library
6. https://wiki.qt.io/Category:Tools::QtCreator

3 Création d'un projet C

Pour créer un projet C :

(a) Choisissez dans la barre de menu l'entrée *File* et dans celle-ci *New File or Project...*

(b) Dans la boîte de dialogue ouverte par ce choix, cliquez *Non-Qt Project* dans le panneau de gauche. Par défaut, *Plain C Project* est sélectionné, cela nous convient. Appuyez sur le bouton *Choose...* en bas à droite.

(c) Donnez le nom de votre choix au projet, par exemple ***xXx***, et créez-le soit en local sur votre bureau, soit quelque part dans votre espace disque personnel `Z:\`. Cliquez *Next >*.

(d) À l'écran *Kit Selection*, cliquez *Next >* sans rien modifier.

(e) À l'écran *Project Management*, cliquez *Finish* sans rien modifier.

Le projet est alors créé et ouvert en mode d'édition.

3.1 C99

Pour configurer le projet [7] de sorte que le compilateur travaille avec le standard C99 [8], voici ce qu'il faut faire :

(a) Ouvrez le fichier `xXx.pro`.

(b) Ajoutez-lui les lignes :

```
QMAKE_CFLAGS += -std=c99 \
                -pedantic-errors \
                -D__USE_MINGW_ANSI_STDIO
```

où le dernier *flag* ne doit être utilisé que si vous travaillez sous *MS-Windows*.

3.2 C11

Pour configurer le projet de sorte que le compilateur [9] travaille avec le standard C11 [10], voici ce qu'il faut faire :

(a) Ouvrez le fichier `xXx.pro`.

(b) Ajoutez-lui les lignes :

```
QMAKE_CFLAGS += -std=c11 \
                -pedantic-errors \
                -D__USE_MINGW_ANSI_STDIO
```

où le dernier *flag* ne doit être utilisé que si vous travaillez sous *MS-Windows*.

7. https://stackoverflow.com/q/18520817
8. https://en.wikipedia.org/wiki/C99
9. https://gcc.gnu.org/onlinedocs/gcc-5.3.0/gcc/C-Dialect-Options.html#C-Dialect-Options
10. https://en.wikipedia.org/wiki/C11_(C_standard_revision)

4 Création d'un projet C++

Pour créer un projet C++ :

(a) Choisissez dans la barre de menu l'entrée *File* et dans celle-ci *New File or Project...*

(b) Dans la boîte de dialogue ouverte par ce choix, cliquez *Non-Qt Project* dans le panneau de gauche. Par défaut, *Plain C Project* est sélectionné, cela ne nous convient pas. Changez la sélection en *Plain C++ Project* puis appuyez sur le bouton *Choose...* en bas à droite.

(c) Donnez le nom de votre choix au projet, par exemple ***yYy***, et créez-le soit en local sur votre bureau, soit quelque part dans votre espace disque personnel `Z:\`. Cliquez *Next >*.

(d) À l'écran *Kit Selection*, cliquez *Next >* sans rien modifier.

(e) À l'écran *Project Management*, cliquez *Finish* sans rien modifier.

Le projet est alors créé et ouvert en mode d'édition.

4.1 C++14

Pour configurer le projet [11] de sorte que le compilateur [12] travaille avec le standard C++14 [13], voici ce qu'il faut faire :

(a) Ouvrez le fichier `yYy.pro`.

(b) Remplacez-y la ligne :

```
CONFIG += console c++11
```

par :

```
CONFIG += console c++14
```

(c) Ajoutez-lui la ligne :

```
QMAKE_CXXFLAGS += -pedantic-errors
```

5 Ajout d'un fichier à un projet

5.1 Nouveau fichier

5.1.1 Fichier d'en-têtes

Prenons le projet C ***xXx*** créé à la section 3. Pour lui ajouter un fichier d'en-têtes inexistant au préalable, procédez de la sorte :

(a) Choisissez dans la barre de menu l'item *File* et dans celui-ci *New File or Project...*

11. https://stackoverflow.com/q/38293989
12. https://gcc.gnu.org/projects/cxx-status.html#cxx14
13. https://en.wikipedia.org/wiki/C%2B%2B14

(b) Dans le panneau de gauche de la fenêtre qui s'ouvre alors, sous *Files and Classes*, sélectionnez *C++*, puis *C++ Header File* dans le panneau central et cliquez *Choose...*

(c) Fournissez le nom de votre choix au fichier d'en-têtes, `test`[14], par exemple.

Le fichier d'en-têtes `test.h` est alors créé, ajouté au projet courant — pour vous en convaincre, allez voir le fichier `xXx.pro` — et ouvert dans l'éditeur.

Projet C++ Pour ajouter un nouveau fichier d'en-têtes à un projet C++, tel ***yYy*** de la section 4, il faut procéder exactement de la même manière, à savoir passer par la création d'un nouveau *C++ Header File*.

5.1.2 Fichier source

Continuons avec le projet C ***xXx***. Pour lui ajouter un fichier source inexistant au préalable, procédez de la sorte :

(a) Choisissez dans la barre de menu l'item *File* et dans celui-ci *New File or Project...*

(b) Dans le panneau de gauche de la fenêtre qui s'ouvre alors, sous *Files and Classes*, sélectionnez *C++*, puis *C++ Source File* dans le panneau central et ensuite cliquez *Choose...*

(c) Fournissez le nom de votre choix au fichier source, `test.c`[15], par exemple.

Le fichier d'en-têtes `test.c` est alors créé, ajouté au projet courant — pour vous en convaincre, allez voir le fichier `xXx.pro` — et ouvert dans l'éditeur.

Projet C++ Pour ajouter un nouveau fichier source à un projet C++, il faut procéder semblablement, à savoir passer par la création d'un nouveau *C++ Source File*. Lors du nommage du nouveau fichier, vous pouvez omettre l'extension `.cpp`, car, par défaut, un fichier source d'extension `.cpp` est créé par Qt Creator.

5.2 Fichier existant

5.2.1 Fichier d'en-têtes ou fichier source

Pour ajouter un fichier d'en-têtes ou un fichier source existant à un projet C ou C++, voici la marche à suivre :

(a) Affichez la vue *Activate Projects* en sélectionnant l'entrée *Projects* dans la boîte combinée[16] (*combobox*) en haut du panneau à droite des boutons *Welcome*, *Edit*, *Design*, etc., ou, plus simplement, en frappant les touches Alt-x.

(b) Cliquez avec le bouton droit sur le nom du projet auquel vous voulez ajouter le fichier et sélectionnez *Add Existing Files...* dans le menu contextuel surgissant.

14. Ou `test.h`.
15. N'oubliez surtout pas l'extension `.c` !
16. https://fr.wikipedia.org/wiki/Bo%C3%AEte_combin%C3%A9e

(c) Naviguez jusqu'à l'emplacement du fichier et cliquez *Open* en bas à droite de la boîte de dialogue.

5.2.2 Fichier de bibliothèque statique

Pour ajouter une bibliothèque statique à un projet C ou C++, voici la marche à suivre :

(a) Affichez la vue *Activate Projects* en sélectionnant l'entrée *Projects* dans la boîte de choix en haut du panneau à droite des boutons *Welcome*, *Edit*, *Design*, etc., ou, plus simplement, en frappant les touches Alt-x.

(b) Cliquez avec le bouton droit sur le nom du projet auquel vous voulez ajouter le fichier et sélectionnez *Add Library...* dans le menu contextuel surgissant.

(c) Comme *Type*, sélectionnez *External library* puis cliquez *Next >*.

(d) Dans les *Details*, naviguez jusqu'à l'emplacement de fichier bibliothèque [17] et renseignez, si nécessaire, le chemin vers les fichiers d'en-têtes des fonctions, classes, etc. de la bibliothèque. Indiquez la plate-forme de développement et les spécifications d'édition des liens éventuelles. Soyez attentif aux différentes cases à cocher [18] (*checkbox*).

(e) Cliquez *Next >* puis *Finish*.

6 Production, test et débogage d'un exécutable

6.1 Production

Pour compiler les fichiers sources du projet ***xXx***, puis éditer les liens des fichiers objets produits en vue de générer un exécutable, cliquez *Build Project "xXx"* dans le menu *Build*, après avoir sauvegardé tous les fichiers sources, d'en-têtes, etc.

6.2 Exécution

Pour tester l'exécutable produit, cliquez *Run* dans le même menu *Build*.

6.3 Débogage

Pour déboguer l'exécutable produit :

(a) Placez — éventuellement ! — d'abord un point d'arrêt sur une ligne du code source. Pour cela, éditez le fichier source correspondant et cliquez dans la rigole à gauche de la ligne désirée : un gros point rouge avec un sablier devrait apparaître.

(b) Cliquez *Start Debugging* dans le sous-menu *Start Debugging* du menu *Debug*.

17. Comme nous travaillons avec gcc, la bibliothèque statique de nom ***bBb*** se trouve dans le fichier `libbBb.a`.

18. https://fr.wikipedia.org/wiki/Case_%C3%A0_cocher

(c) Plusieurs scénarios sont possibles :

(i) L'application termine proprement sans atteindre le point d'arrêt : le débogage finit lui aussi proprement.

(ii) L'application plante avant d'atteindre le point d'arrêt : dans ce cas, la ligne problématique est pointée dans le perspective *Debug* et la vue *Stack* renseigne la *stack trace*[19].

(iii) Le point d'arrêt est atteint : la perspective *Debug* indique une multitude d'informations, utilisez *Continue*, *Step Over*, *Step Into* et *Step Out* à bon escient.

7 Alternance de travail à l'école et ailleurs

Pour travailler, par exemple, chez vous sur un projet commencé à l'école, ou l'inverse, il faut d'abord avoir bien sauvegardé le(s) répertoire(s) contenant tous les fichiers du projet. Pour retrouver votre projet, utilisez l'entrée *Open File or Project...* du menu *File* et choisissez dans la fenêtre de sélection de fichier qui s'ouvre alors le fichier d'extension `.pro` correspondant au projet.

Il se peut qu'à l'ouverture du projet, une boîte de dialogue de titre *Settings File for 'xXx' from a different Environment ?*, où *xXx* est le nom du projet, apparaisse. Le plus simple dans ce cas est de cliquer *Yes* et d'essayer *Build* ⇨ *Build Project "xXx"*. Si cela échoue, tentez *Build* ⇨ *Run qmake* puis *Build* ⇨ *Build Project "xXx"*. Si cela ne fonctionne toujours pas, vous devez configurer explicitement le projet pour la nouvelle machine en commençant par cliquer sur le bouton *Projects*, dans la colonne de gauche de l'interface de Qt Creator, entre les boutons *Debug* et *Analyze*.

8 *Plugins* à activer (éventuellement)

Pour activer un *plugin* inactif de Qt Creator, il faut commencer par afficher l'état des *plugins* en sélectionnant l'entrée *About Plugins...* du menu *Help*. Une fois la demande d'activation du *plugin* réalisée en cochant sa case *Load*, il faut redémarrer Qt Creator pour la rendre effective.

8.1 Beautifier

Le *plugin* Beautifier utilise un logiciel tiers pour mettre en forme les codes sources. Il est rangé parmi les *plugins C++*.

À l'école, c'est le logiciel libre Artistic Style[20] qui est utilisé par Beautifier. Voici comment configurer ce *plugin* :

(a) dans le menu *Tools*, choisissez *Options...*

19. https://en.wikipedia.org/wiki/Stack_trace
20. http://astyle.sourceforge.net/

(b) sélectionnez *Beautifier* dans le panneau de gauche ;

(c) sous l'onglet *Artistic Style*, fournissez le chemin :

```
C:\Program Files (x86)\AStyle\bin\AStyle.exe
```

à droite de l'étiquette *Artistic Style command* ;

(d) selon vos préférences, choisissez un fichier de style global ou propre à chaque projet ;

(e) cliquez *OK*.

À titre d'exemple, voici le contenu du fichier de style utilisé par l'auteur de ce TD :

```
 --style=allman
--indent=spaces=4
--indent-modifiers
--indent-switches
--indent-cases
--indent-col1-comments
--pad-oper
--pad-header
--align-pointer=middle
--align-reference=middle
--break-closing-brackets
--keep-one-line-statements
--convert-tabs
--max-code-length=70
--break-after-logical
--close-templates
--lineend=linux
```

Davantage d'informations sur les options de formattage de code sont disponibles sur le site web [21] d'Artistic Style.

8.2 Todo

Le *plugin* Todo permet d'atteindre rapidement des étiquettes disséminées dans les sources d'un projet. Il est rangé parmi les *plugins Utilities*.

Pour le configurer et ajouter de nouveaux *tags*, faites comme pour le *plugin* Beautifier mais choisissez *To-Do* dans le panneau de gauche de la fenêtre *Options*. Le résultat de l'analyse du fichier ou du projet courant, selon la configuration, apparaît en bas de la zone d'édition du code, dans la vue *To-Do Entries*.

21. http://astyle.sourceforge.net/astyle.html