Introduction à l'EDI Code::Blocks (v13)

B. Baert et F. Ludewig – 2015
Introduction à la programmation, P. Schlagheck

Le logiciel Code::Blocks peut être téléchargé à l'adresse suivante : http://www.codeblocks.org/downloads/binaries (choisir la version incluant MinGW)

L'EDI Code::Blocks : un Environnement de Développement Intégré

(IDE - *Integrated Development Environment* en anglais)

Un environnement de développement intégré comme Code::Blocks est un programme qui regroupe un ensemble d'outils pour la programmation et le développement de logiciels. Il est principalement constitué :

- d'un éditeur de texte (pour modifier le code source) ;
- d'une interface avec un compilateur (pour générer le programme exécutable) ;
- d'un débogueur (pour analyser les erreurs dans un programme) ;
- de divers outils d'aide à la génération de code.

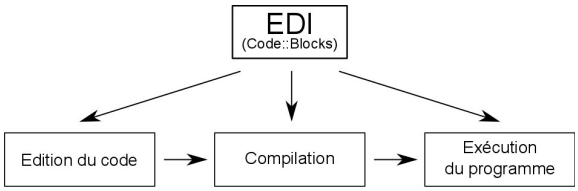

Fig. 1 : Environnement de développement intégré (EDI)

L'éditeur de code

Il s'agit d'un éditeur de texte adapté au traitement d'un code source. Il intègre généralement une coloration syntaxique automatique, qui met en évidence les mots clés du langage concerné.

Le compilateur

Le code source produit dans l'éditeur de code peut être directement compilé à partir de l'EDI : celui-ci va faire appel à un compilateur externe[1] qui génèrera un programme exécutable. Une fenêtre spécifique de l'EDI permet de suivre le processus de compilation et de voir les éventuels messages d'erreurs produits par le compilateur.

Le débogueur

Le débogueur permet de rechercher des erreurs dans le fonctionnement d'un programme. Lors du débogage, l'EDI contrôle l'exécution du programme étape par étape. Il permet également de suivre l'évolution des valeurs des différentes variables du programme tout au long de l'exécution.

1 Dans le cas de Code::Blocks, il s'agit de MinGW, une version adaptée à Windows du célèbre compilateur GCC

Paramètres du compilateur à utiliser sur votre ordinateur personnel

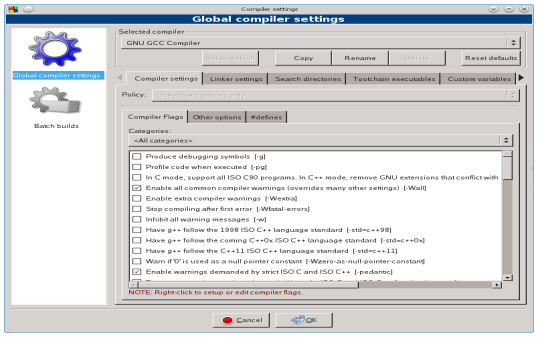

Fig. 2 : Paramètres du compilateur

Les ordinateurs utilisés dans les salles informatiques lors des TPs et de l'examen sont configurés avec certains paramètres spécifiques pour le compilateur. Il est recommandé d'utiliser ces mêmes paramètres sur vos ordinateurs personnels, afin d'avoir des résultats identiques.

Pour modifier ces paramètres, utilisez le menu *Settings → Compiler* (Fig. 2) et cochez les cases *-Wall*, *-pedantic*, *-pedantic-errors*, *-Wfloat-equal* et *-Wshadow*

Création d'un nouveau projet console C++ avec Code::Blocks

Lancez l'environnement de développement intégré (EDI) Code::Blocks. Créez un nouveau projet soit via le menu *File → New → Project...*, soit en cliquant sur le raccourci *Create a new project* de la fenêtre d'accueil (Fig. 3).

Dans la fenêtre qui s'ouvre (Fig. 4) choisissez *Projects* dans la liste de gauche et *Console application* dans la liste de droite et cliquez sur *Go.*

L'écran suivant permet de choisir le langage désiré. Sélectionnez *C++* et cliquez sur *Next.* La fenêtre qui suit (Fig. 5) permet de donner un nom au projet et de préciser où celui-ci doit être enregistré sur le disque.

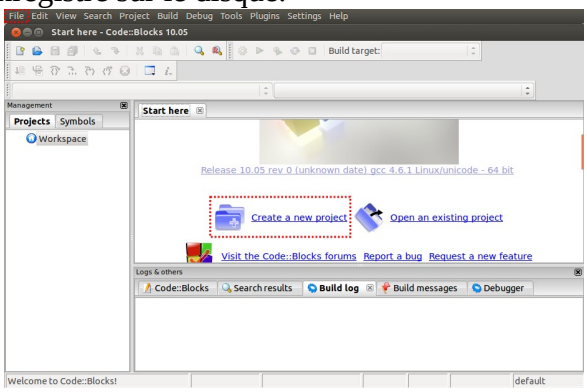

Fig. 3 : écran d'accueil

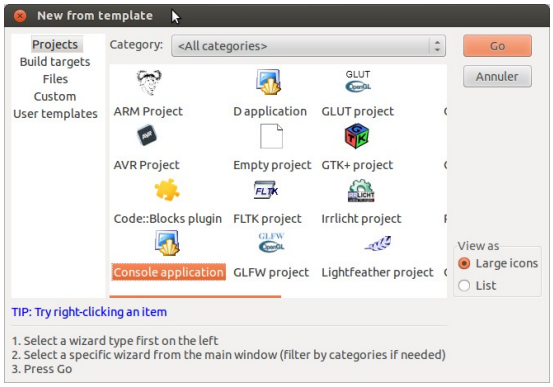

Fig. 4 : nouveau projet

> Remarque : les projets C ou C++ sont souvent constitués de plusieurs sources (d'extension *c* ou *cpp* respectivement) et de fichiers d'en-tête (d'extension .h). Tous ces fichiers sont enregistrés dans un dossier portant le nom du projet donné dans *Project title* et localisés à l'emplacement spécifié par *Folder to create project in.* En plus du dossier projet, il existe un fichier projet (d'extension .cbp) permettant de stocker tous les paramètres du projet ainsi que la liste des fichiers appartenant au projet. La boîte de texte *Resulting filename* donne le chemin absolu complet vers le fichier projet !

Remplissez la case *Project title* avec un nom de projet et la case *Folder to create project in* avec un chemin vers un dossier du disque dur. Laissez les deux autres cases avec leur valeur par défaut et cliquez sur *Next.*

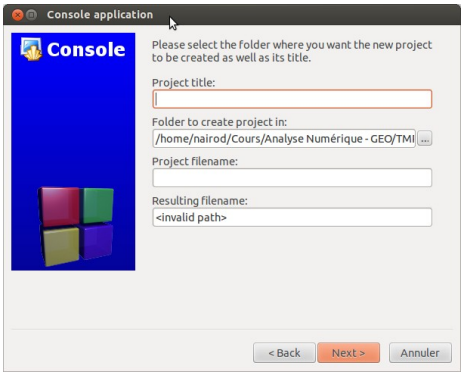

Fig. 5 : nom et localisation du projet

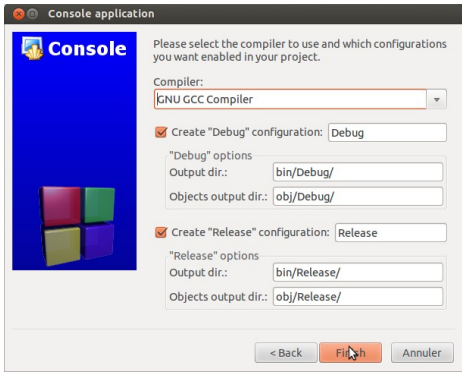

Fig. 6 : configuration du projet

Ne modifiez rien dans la fenêtre de configuration (Fig. 6) et cliquez sur *Finish*.

Par défaut, Code::Blocks a créé un fichier nommé *main.cpp* contenant une fonction *main* et un code affichant « *Hello World !* » Vous pouvez commencer à coder directement dans ce fichier[2].

Gestion des fichiers dans un projet

La liste des fichiers appartenant à un projet est disponible sous le nom du projet de l'onglet *Projects* de la boite *Management* (Fig. 7). Si la boîte *Management* n'est pas visible, affichez-la via le menu *View → Manager*. Il est utile de garder cette boite à portée de main, elle est ancrable sur les bords de la fenêtre (cliquez sur la barre de titre et maintenez le bouton enfoncé, déplacez la boîte jusqu'à un rebord (un cadre semi-transparent confirme l'attache), relâchez la souris). Pour ouvrir un fichier, double-cliquez sur son nom dans la boîte *Management*[3].

La liste des fichiers ouverts actuellement dans l'éditeur se situe dans la barre d'onglets de la fenêtre de travail (Fig. 8). Les fichiers précédés d'une astérisque ont subi des modifications et n'ont pas encore été enregistrés.

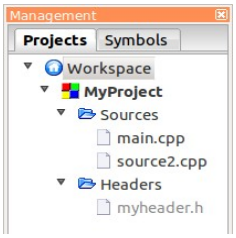

Fig. 7 : manager de projet

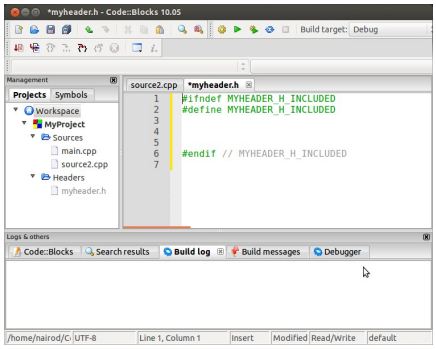

Fig. 8 : fenêtre de travail

Compiler et exécuter un projet

On peut compiler et exécuter un projet via la barre d'outil *compiler* (Fig. 9). Si celle-ci n'est pas visible, on peut l'afficher via le menu *View → Toolbars → Compiler*. L'icône en forme de roue dentée sert à compiler l'ensemble des fichiers du projet et à créer un fichier exécutable. L'icône en forme de triangle (play) vert sert à lancer le fichier exécutable préalablement créé par la compilation. L'icône suivante effectue ces deux opérations l'une à la suite de l'autre. Les deux flèches bleues formant un cercle servent à reconstruire l'ensemble du projet[4]. L'icône en forme de

2 Il est usuel que le fichier contenant la fonction *main* s'appelle *main.cpp*, mais ce n'est pas obligatoire.

3 Les fichiers présents dans le projet sont directement triés suivant leur type (source, en-tête) dans deux « dossiers » virtuels *Sources* et *Headers*. Ces dossiers n'existent pas réellement sur le disque, mais permettent de distinguer facilement les fichiers de source et les fichiers d'en-tête dans l'interface de Code::Blocks.

4 L'EDI ne recompile pas l'entièreté du projet à chaque fois, mais uniquement ce qui est nécessaire. Il peut arriver

croix grisée devient rouge lorsque le projet est en cours de compilation ou d'exécution et permet un arrêt d'urgence. La liste déroulante *Build target* permet de déterminer la configuration à utiliser pour la compilation. Laissez celle-ci sur *Debug*[5] lors de l'élaboration du projet.

Fig. 9 : barre d'outil *compiler*

Pendant la compilation, l'EDI affiche un certain nombre d'informations sur l'état d'avancement de la construction du projet dans la fenêtre de sortie (Fig. 10). Si celle-ci n'est pas visible affichez-la via le menu *View → Logs*. Une fois la compilation finie, si celle-ci indique *0 error(s), 0 warning(s)*, la compilation s'est bien passée (si seuls des *warning* sont présents, un fichier exécutable a quand-même été généré). Si des erreurs se sont produites lors de la compilation, elles sont répertoriées sous l'onglet *Build messages*[6].

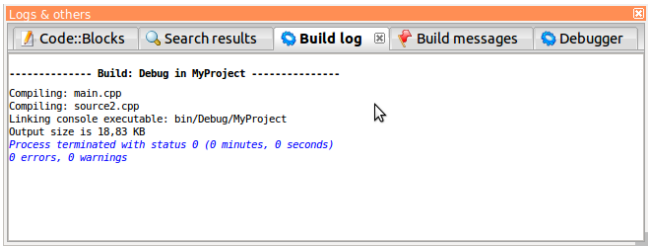

Fig. 10 : fenêtre de sortie : Build Log

Ouvrir/Sauver/Fermer un projet

Pour ouvrir un projet préalablement enregistré, choisissez le menu *File → Open...* et naviguez jusqu'à votre fichier projet (extension .cbp). Code::Blocks peut ouvrir plusieurs projets en même temps, mais c'est rarement une bonne idée. On peut fermer un projet via le menu *File → Close Project*. Dans le menu *File*, vous trouverez également *Save project* et *Save all files* permettant respectivement de sauver le projet en cours et de sauver l'ensemble des modifications apportées aux fichiers. L'ensemble des fichiers d'un projet sont automatiquement sauvegardés lorsque le projet est compilé[7].

Indentation automatique dans Code::Blocks

Si votre code est mal indenté[8], il existe un *plugin* dans Code::Blocks permettant de ré-indenter votre code de façon automatique. Dans le menu *Plugins* sélectionnez *Source code formatter*.

---

cependant, par exemple lorsque des fichiers ont été modifiés en-dehors de l'éditeur, que Code::Blocks s'y perde et ne compile pas correctement. Reconstruire complètement le projet permet de corriger ces erreurs.

5 La configuration *Debug* produit un programme plus lent que la configuration *Release*, mais permet de détecter certaines erreurs.

6 Les erreurs sont affichées les unes à la suite des autres. Lors de la correction des erreurs, il préférable de commencer par la première car certaines erreurs peuvent en entraîner d'autres.

7 La compilation étant effectuée par un programme différent (le compilateur), les fichiers doivent être sauvés sur le disque avant chaque compilation (ceci est fait automatiquement), sinon les modifications dans l'EDI ne seraient pas appliquées dans l'exécutable nouvellement compilé.

8 Un code bien indenté est plus facile à lire. Le formatage de la source suit la structure du code et permet d'identifier rapidement le début et la fin des blocs de codes.