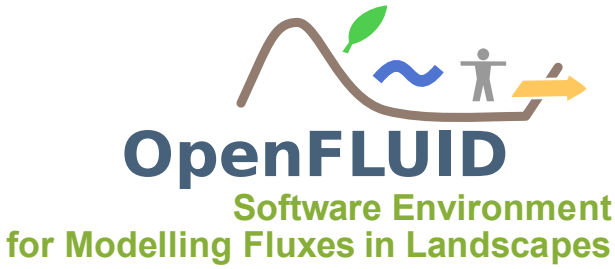


TP1 : Création d'une fonction de simulation

Objectifs:	Créer une fonction de simulation et sa signature, construire et compiler pour une utilisation avec le moteur
Pré-requis:	TP0

1 Créer une fonction de simulation

Il existe 3 façons de créer le code source d'une fonction de simulation :

- Ecrire le code source "à la main", à partir d'un code vierge : long, fastidieux, source d'erreurs, ...
- Utiliser le buddy `newfunc` via l'application `openfluid-engine` : pratique, basique, ...
- utiliser le plug-in OpenFLUID pour Eclipse : facile, assisté, intégré à l'environnement de développement, ...

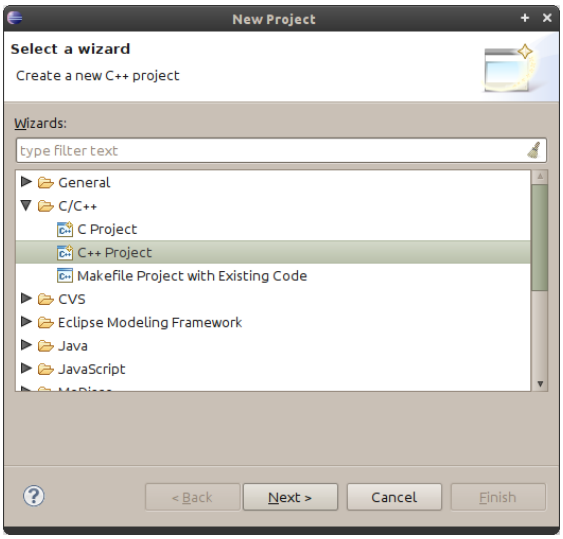
Nous allons utiliser cette 3ème possibilité pour cet exercice.

1.1 Lancement d'Eclipse

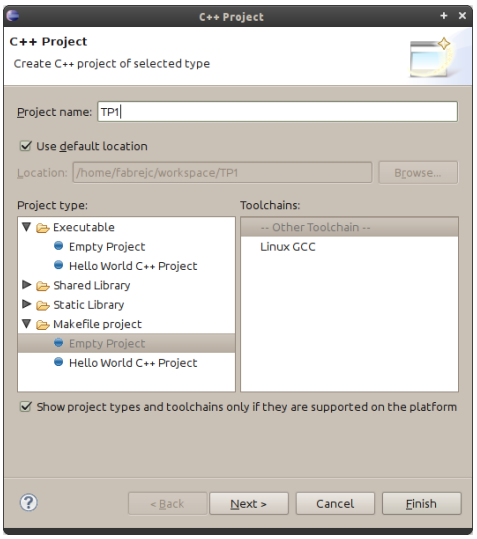
Pour lancer Eclipse, cliquez sur l'icône `Eclipse` présente sur le Bureau. Au lancement d'Eclipse, il peut vous être demandé de choisir le chemin du workspace que vous souhaitez utiliser. Il est conseillé d'utiliser celui proposé par défaut (sous Linux : `/home/openfluid/formation/workspace`). Le workspace est le répertoire qui contiendra l'ensemble des projets de fonctions de simulations développées dans cet exercice ainsi que les suivants.

1.2 Création du projet C++

Pour créer un nouveau projet C++, aller dans le menu *File* > *New* > *Project*.... Choisir *C++ Project* et cliquer sur *Next*



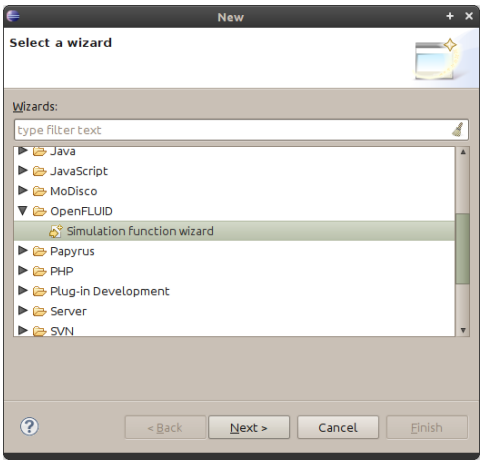
Nommer le projet `training.signal.prod` dans le champ *Project name*, choisir *Makefile project / Empty project* comme *Project type*. La rubrique *Toolchains* est à positionner sur *Other Toolchain*. Cliquer sur *Finish*



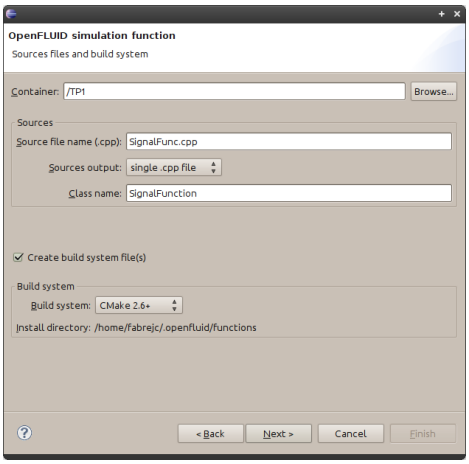
1.3 Création du squelette de la fonction de simulation

Nous allons maintenant utiliser le plug-in OpenFLUID pour Eclipse afin de générer le code source "vide" d'une fonction de simulation.

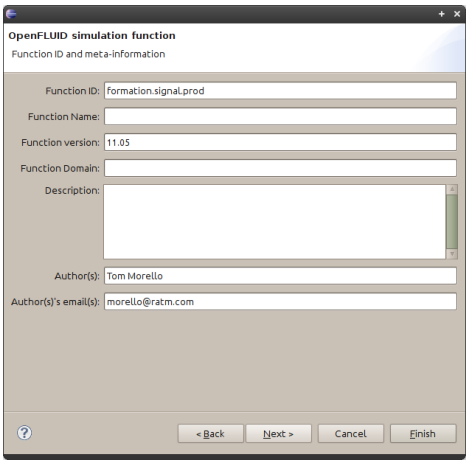
Faire un clic-droit sur le projet que vous venez de créer et choisir *New > Other...*. Dans la fenêtre qui s'ouvre, choisir *OpenFLUID > Simulation function wizard* et cliquer sur *Next*.



La fenêtre suivante est la 1ère étape de la création de la fonction de simulation. Cette première étape consiste à paramétrer ce qui va être créé.
Choisir `/training.signal.prod` comme nom de *Container*. Choisir `SignalFunc.cpp` comme nom de fichier (*Source file name*), `SignalFunction` comme nom de classe C++ qui contiendra la fonction (*Class name*). *Sources output* doit être positionné à *single .cpp file*, *Create build system* doit être coché, et *Build system* doit être positionné sur *CMake 2.6+*.
Cliquer sur *Next* une fois cette étape complétée.



La deuxième étape consiste à renseigner les informations sur la fonction de simulation, et notamment son identifiant (*ID*) par lequel elle va être connue. Taper `training.signal.prod` comme ID de fonction. Les autres champs sont facultatifs, vous pouvez y renseigner votre nom et email. Cliquer sur *Next* une fois cette étape complétée.

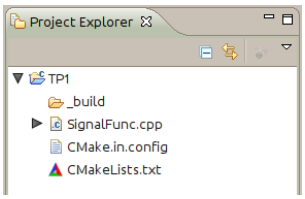


La troisième et dernière étape consiste à renseigner les informations sur les variables/paramètres/données d'entrées/évènements qui sont produits/utilisés/mis à jour par la fonction de simulation. Nous n'utiliserons pas cette troisième étape, vous pouvez donc cliquer sur *Finish*.

Si tout s'est bien passé, vous devriez obtenir 3 fichiers dans le *Project Explorer* d'Eclipse :

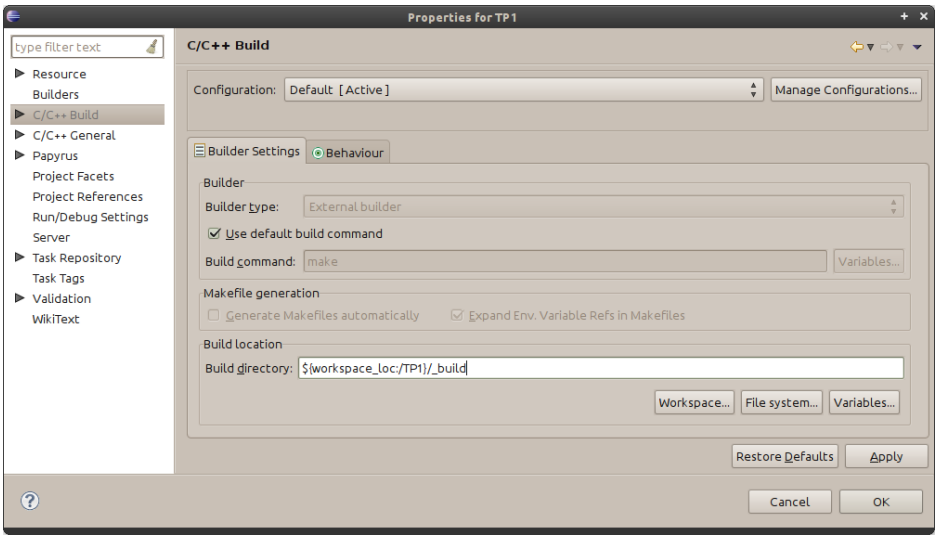
- `SignalFunc.cpp` : fichier source de la fonction
- `CMakeLists.txt` : fichier du système de construction de la fonction (à priori, à ne pas modifier)
- `CMake.in.config` : fichier de configuration de la construction de la fonction

ainsi qu'un sous-répertoire `_build` qui contiendra les résultats de compilation/construction de la fonction.

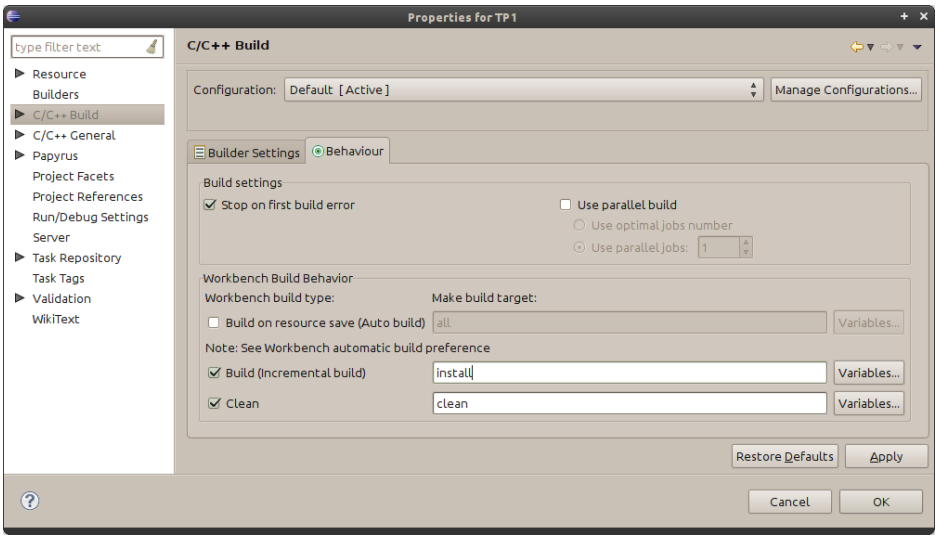


1.4 Finaliser le projet C++

Avant de construire la fonction de simulation créée, il est nécessaire de paramétrer plus finement le projet Eclipse. Pour cela, nous allons nous rendre dans les propriétés du projet, en faisant un clic-droit sur le nom du projet, puis en cliquant sur *Properties*.
La fenêtre *Properties* du projet Eclipse s'ouvre, se rendre alors dans la rubrique *C/C++ Build*. Tout d'abord, indiquer dans le champ *Build directory* que la construction de la fonction doit se faire dans le sous-répertoire `_build` du projet.
`${workspace_loc:/training.signal.prod}/_build`



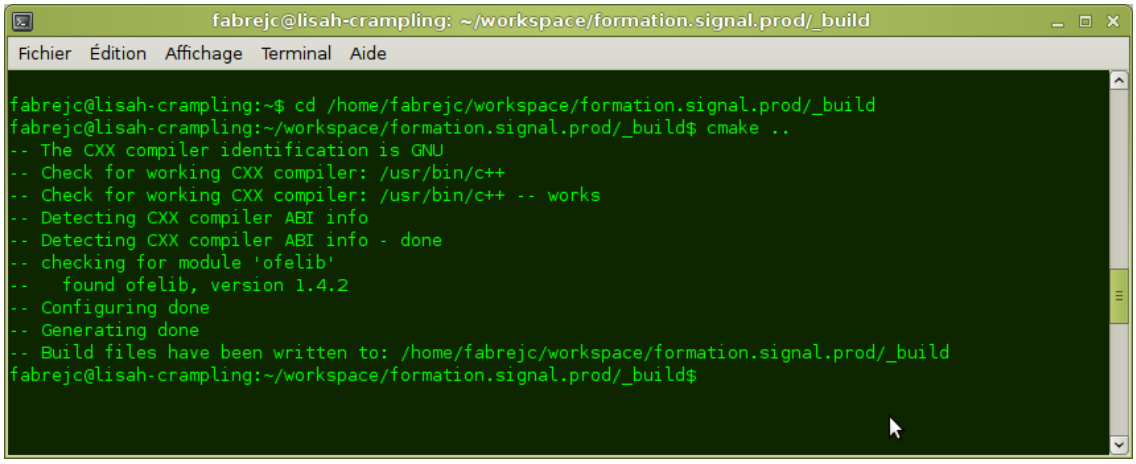
Ensuite passer sur l'onglet *Behaviour*, mettre la valeur `install` dans le champ *Build (Incremental build)* afin que la fonction de simulation soit automatiquement placée dans un répertoire connu par le framework OpenFLUID et cliquez ensuite sur OK.



2 Construire et installer la fonction de simulation

En premier lieu, il est nécessaire d'initialiser le système de construction. Pour cela, nous allons utiliser la ligne de commande du système d'exploitation. Les instructions qui suivent s'appliquent à un système Linux, mais peuvent très facilement être reproduites sous un autre système (Win32, MacOSX, ...)
Après avoir ouvert une console ligne de commande (Terminal), se rendre dans le sous-répertoire `_build` du projet via la commande :
`cd /home/openfluid/formation/workspace/training.signal.prod/_build`.
Une fois positionné dans ce sous-répertoire `_build`, exécuter la commande
`cmake ..`

qui effectue un ensemble de vérifications et met en place le système de construction. Elle n'est nécessaire qu'à la première construction du projet, ou lors de la modification du fichier `CMakeLists.txt` ou `CMake.in.config`, ce qui ne devrait pas être nécessaire au cours du présent TP. Si tout c'est bien passé, vous devriez obtenir un écran comme celui-ci :



La construction/installation peut se faire de deux manières :

- Soit en utilisant la ligne de commande en exécutant la commande `make install` depuis le sous-répertoire `_build`
- Soit depuis Eclipse, dans le menu *Project* > *Build Project*. Cette 2ème méthode est à privilégier.

Cette construction/installation est à exécuter à chaque fois que le code source de la fonction de simulation est modifié.

Pour vérifier que la fonction a été correctement construite et installée, exécuter la commande suivante dans un terminal :
`openfluid-engine -f`.
La fonction devrait alors apparaître dans la liste.