Année universitaire 2016-2017
Université de Caen Normandie

Rapport sur les snippets

Théo Sarrazin
L2 Informatique

Table des matières

1 Snippets

1.1 Les snippets : explication

Les snippets sont des bouts de code qui sont réutilisables d'un projet à un autre (comme les conditions, les boucles et les définitions de fonction par example). Dans notre IDE nous utilisons des mots clés que l'on associes à ces bouts de code. Lorsque l'on tape un des mots clés puis que l'on presse la touche **tabulation**, le mot clé est remplacé par le bout de code qui correspond.

1.2 Intergration des bouts de code dans l'IDE

1.2.1 Utilisation du JSON

Pour l'intégration des snippets dans l'IDE, nous avons utilisés des fichier JSON, qui contiennent des dictionnaires avec comme clé le mot clé qui correspond au bout de code et en valeur le dit bout de code, suivit de nombre de ligne de le cursor doit remonter, puis le nombre de charactère char que le cursor doit parcourir vers la droite pour être à l'emplacement voulu. De plus nous avons un fichier JSON par language (C, python, etc ...)

```
{
    "int" : ["int maFonction(){\n\treturn 0;\n}", 2, 12],
    "void" : ["void maFonction(){\n\n}", 2, 12],
    "bool" : ["bool maFonction(){\n\treturn true;\n}", 2, 12],
    "char" : ["char maFonction(){\n\treturn \"a\";\n}", 2, 12],
    "double" : ["double maFonction(){\n\treturn 0.0;\n}", 2, 12],
    "if" : ["if(true){\n\tsomeStuffHere();\n}", 2, 4],
    "elseif":["else if(true){\n\tStuffHere();\n}", 2, 10],
```

FIGURE 1 – Extrait du fichier JSON pour le language C

1.2.2 Connection entre le JSON et l'IDE

Pour capturer l'appuie sur la touche **tabulation**, nous utilisons le méthode **keyPressEvent** de notre classes **Editeur**. Cette méthode est appelée à l'appuie sur une des touches du clavier. Par la suite, si la touche est la touche **tabulation**, nous parsons le JSON pour avoir touts les couples mot clé - bout de code. Puis en fonction du mot placé au niveau du curseur, on le remplace ou non par le bout de code correspondant.

FIGURE 2 – Exemple d'utilisation des snippets avec une boucle for en Python

2 Compilateur

2.1 Compilateurs utilisés

Pour notre IDE, nous avons utilisés le compilateur **GCC**, pour compiler les projets de type **C**. Mais nous utilisons aussi les interpréteur de python afin d'interpréter les projets de type **Python**. Par la suite, pour chaque nouveaux languages il nous suffit d'utiliser le bon compilateur ou le bon interpréteur.

2.2 Integration de GCC à notre IDE

Pour utilisé **GCC** avec notre IDE nous avons créé une fenêtre de configuration afin de permettre à l'utilisateur de choisir la configuration adéquate à la compilation de son projet.

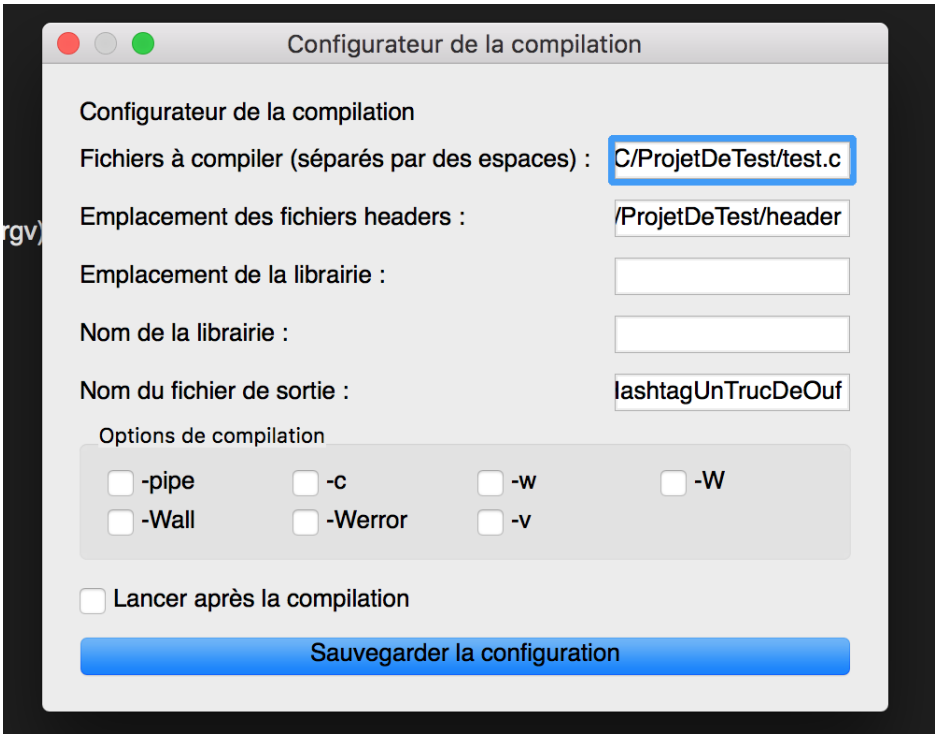

FIGURE 3 – Fenêtre de configuration de GCC

Par la suite, chaque information de cette fenêtre est convertie en une chaîne de caractère du type :

gcc /Users/theosarrazin/workplaceC/ProjetDeTest/test.c -I /Users/theosarrazin/workplaceC/ProjetDeTest/header -o MonExecutable

Cette chaîne de caractère est stockée dans le fichier XML de configuration du projet.

Par la suite on exécute cette ligne de commande, puis on récupère sur la sortie d'erreurs les eventuelles erreurs afin de pouvoir les afficher à l'utilisateur après les avoir parsé pour avoir un bon formatage.

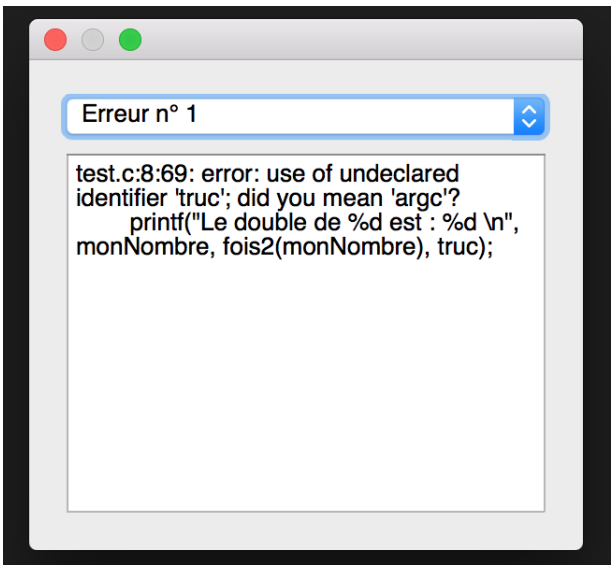

FIGURE 4 – Affichage des erreurs de GCC dans l'IDE

Pour la partie interpréteur la demarche est identique.

3 Cache

3.1 Utilité du cache

Afin de pouvoir faire de la coloration, nous avons utilisés **Lex** mais pour les fichiers de plusieurs 100ène de lignes **Lex** prend plusieurs secondes pour nous donner la listes des tokens du fichier. Nous avons donc décidé d'utiliser un fichier de cache afin de ne pas utiliser **Lex** pour colorer des lignes que nous avons déjà coloré de manière ultérieure.

3.2 Intégration du cache dans l'IDE

Pour le cache, nous avons décidé d'utiliser un fichier JSON pour l'enregistrer. Le fonctionnement est assez simple la clé est le bout de code que l'on donne à **Lex** et la valeur est la réponse de ce dernier. Par la suite il nous reste plus qu'à utiliser les valeurs du fichier JSON à la place de **Lex**

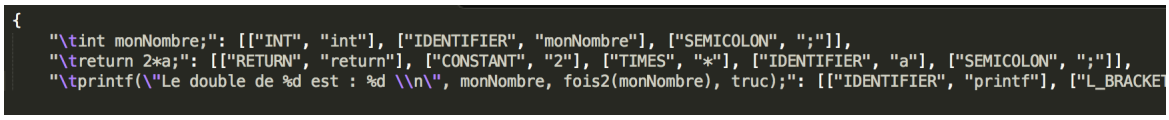

FIGURE 5 – Extrait d'un JSON contenant le cache

Grâce au cache nous avons un gain de temps considérable sur l'ouverture des documents.