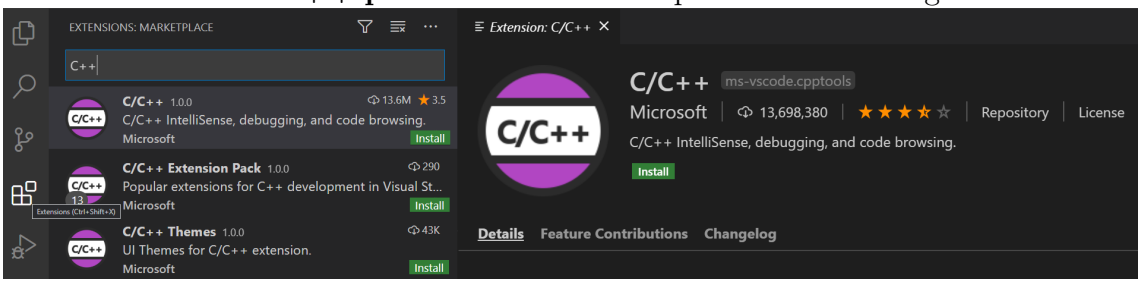
Installer VS Code pour C++ sur Linux

1 Pré-requis

– Installer **Visual Studio Code** : `https://code.visualstudio.com`
– Installer l'**extension C++ pour VS Code**. Aller pour cela dans l'onglet **Extensions**.



2 Compilateur

Vérifier que le compilateur est installé. Pour cela, dans un **terminal**, entrer la commande suivante :

```
gcc -v
```

Si gcc n'est pas installé, commencer par mettre à jour les paquets d'installation :

```
sudo apt-get update
```

Ensuite installer le compilateur et aussi le débogueur GDB :

```
sudo apt-get install build-essential gdb
```

3 Créer son espace de travail sur VS Code

– Commencer par ouvrir un nouveau dossier dans VS Code qui sera notre espace de travail :
Chercher **Open Folder** dans le menu **File** ou dans **Explorer**.
– Créer un nouveau fichier **main.cpp** et y ajouter le code suivant :

```
#include <iostream>
using namespace std;

int main()
{
    cout << "Hello world!" << endl;
    return 0;
}
```

– On peut cocher la case **Auto save** dans le menu **Fichier** pour que les fichiers soient automatiquement sauvegardés à chaque modification.

4 Makefile : le fichier de compilation

– Créer un fichier qui s'appelle **Makefile** (avec un M majuscule et sans extension) et y qjouter le code suivant :

```
CFLAGS = -g -pedantic -Wall -Wextra

monprogramme : main.o
   g++ -g -o monprogramme main.o

main.o : main.cpp
   g++ -c $(CFLAGS) main.cpp

clean:
   rm *.o monprogramme
```

Vous pouvez mettre le nom que vous voulez à la place de `program`, ce sera le nom de votre exécutable.

– Pour compiler, il suffit de saisir dans le terminal la commande :

```
make
```

– Pour faire le ménage, i.e. supprimer l'exécutable et les fichiers de compilation, saisir la commande suivante dans le terminal :

```
make clean
```

– Un simple `ls` dans le terminal permet de voir les fichiers générés. Il ne reste plus qu'à lancer le programme :

```
./monprogramme
```

Cela suffit déjà à compiler et lancer des programmes. Mais des fois il y a besoin d'aller plus loin et notamment de déboguer notre programme. C'est là qu'interviennent les sections suivantes.

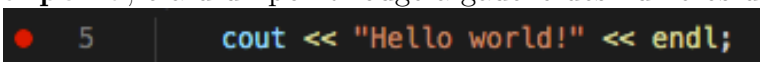
5 Débogueur

– Dans le menu, choisir **Run/Add configuration** puis choisi **C++ (GDB/LLDB)**.

– Cela va générer un fichier `launch.json` dans le répertoire `.vscode`.
Modifier le fichier pour qu'il ressemble à ce qui suit, notamment :
Saisir dans la ligne `""program": "${workspaceFolder}/monprogramme","` où `monprogramme` doit être le nom du programme principal, tel que vous l'avez spécifié dans le Makefile.

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "g++ build and debug active file",
      "type": "cppdbg",
      "request": "launch",
      "program": "${workspaceFolder}/monprogramme", // A MODIFIER
      "args": [],
      "stopAtEntry": false,
      "cwd": "${workspaceFolder}",
      "environment": [],
      "externalConsole": false,
      "MIMode": "gdb",
      "setupCommands": [
        {
          "description": "Enable pretty-printing for gdb",
          "text": "-enable-pretty-printing",
          "ignoreFailures": true
        }
      ]
    }
  ]
}
```

– Ajouter un **breakpoint** ; c-à-d un point rouge à gauche des numéros de ligne.



– Lancer le débogueur en cliquant sur **Run/Start debugging** et utiliser les icônes pour avancer dans le code.

– **Attention de bien recompiler après toute modification du code source, avant de lancer le débogueur, sinon il lancera la version précédente du programme !** Pour recompiler automatiquement avant de lancer le débogueur, voir la section suivante.

6 Automatiser la compilation avant le lancement du débogueur

– Cliquer dans le menu sur **Terminal/Configure Default build task** puis cliquer sur **Create tasks.json file from template** puis choisir **Others**.
– Saisir ensuite le fichier **tasks.json** pour qu'il ressemble à ce qui suit :

```
{
   "version": "2.0.0",
   "tasks": [
      {
         "type": "shell",
         "label": "build", //donner le nom qu'on veut
         "command": "make", //appeler le Makefile
         "problemMatcher": [
            "$gcc" //bien nommer le debugger
         ],
         "group": {
            "kind": "build",
            "isDefault": true
         }
      }
   ]
}
```

– Puis dans le fichier **launch.json**, ajouter la ligne

```
"preLaunchTask": "build"
```

Attention, il faut que la valeur de `preLaunchTask` (`build` ici) soit la même que celle du `label` du fichier `tasks.json`.

Ce qui donne pour le fichier **launch.json** :

```
{
  "version": "0.2.0",
  "configurations": [
    {
      "name": "g++ build and debug active file",
      "type": "cppdbg",
      "request": "launch",
      "program": "${workspaceFolder}/monprogramme", // A MODIFIER
      "args": [],
      "stopAtEntry": false,
      "cwd": "${workspaceFolder}",
      "environment": [],
      "externalConsole": false,
      "MIMode": "gdb",
      "setupCommands": [
        {
          "description": "Enable pretty-printing for gdb",
          "text": "-enable-pretty-printing",
          "ignoreFailures": true
        }
      ],
      "preLaunchTask": "build"
    }
  ]
}
```