

Le même programme en C++

Objectif : configurer l'éditeur pour un second langage, compilé, et comparer son programme à celui écrit en Python

`cours1/1e_hello_cpp/` $\approx 10'$

Installer l'extension C/C++

Un autre langage demande une autre extension. Celle-ci apporte la coloration, la vérification et un bouton d'exécution pour C et C++.

L'action	Ce qu'elle donne
Ctrl + Maj + X, chercher « C/C++ »	l'extension publiée par Microsoft, à installer
Ouvrir <code>cours1/1e_hello_cpp/</code> , puis <code>bonjour.cpp</code>	le code se colore, et les fautes d'écriture se soulignent

Attention. L'extension n'apporte pas de compilateur. Elle sait lire le C++, pas le traduire : c'est l'objet de la diapositive suivante.

Installer un compilateur

Python vient avec l'environnement du module. Un compilateur C++, non : Windows n'en fournit aucun.

	Linux, macOS	Windows
Le compilateur	g++, presque toujours déjà là	aucun d'origine
Comment l'obtenir	rien à faire	<code>conda install -c conda-forge "gxx=15.3.0"</code>
Ce qu'on tape ensuite	<code>g++ ...</code>	<code>x86_64-w64-mingw32-g++ ...</code>

L'environnement conda ne sert pas qu'à Python : il installe aussi des outils. La version est fixée parce que la dernière, 16.2.0, ne termine pas une compilation sous Windows : c'est la diapositive suivante.

L'erreur crt2.o sous Windows

Avec la dernière version du paquet, la traduction du fichier réussit et l'assemblage du programme échoue : les fichiers de démarrage sont installés là où le compilateur ne les cherche pas.

Terminal — cours1/1e_hello_cpp

```
(info01) ...\\1e_hello_cpp> x86_64-w64-mingw32-g++ bonjour.cpp -o bonjour.exe
ld.exe: cannot find crt2.o: No such file or directory
ld.exe: cannot find default-manifest.o: No such file or directory
collect2.exe: error: ld returned 1 exit status
```

Relevé sur un poste de l'école en septembre 2026, avec gxx 16.2.0. La version 15.3.0 installée par la commande de la diapositive précédente ne montre pas l'erreur.

Compiler, puis lancer

Deux commandes au lieu d'une : la première produit un fichier, la seconde exécute ce fichier.

	Ce qu'il faut faire
1	Terminal → Nouveau terminal, dans <code>cours1/1e_hello_cpp/</code>
2	taper <code>g++ bonjour.cpp -o bonjour</code> , puis Entrée
3	taper <code>./bonjour</code> , puis Entrée
4	en option : mettre tout <code>bonjour.cpp</code> sur une seule ligne, recompiler

L'étape 2 n'affiche rien, et c'est normal : elle produit un fichier. Sous Windows, la commande est `x86_64-w64-mingw32-g++`, le fichier produit s'appelle `bonjour.exe` et se lance par `.\bonjour.exe`.

Ce que chaque lancement a produit

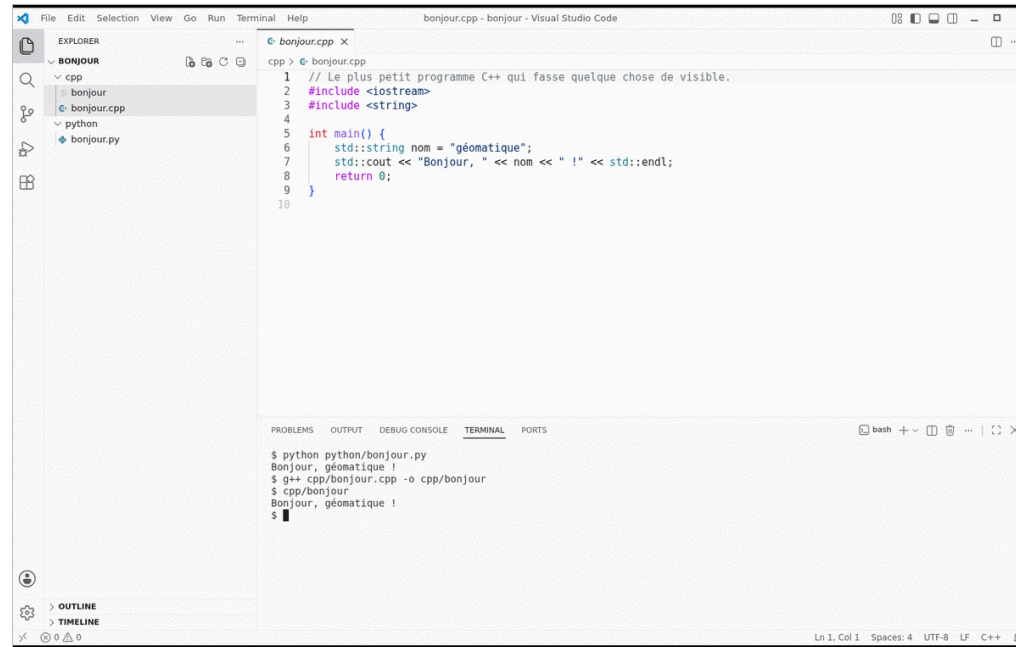
Les deux programmes affichent la même phrase. Ce qui les distingue est le nombre d'étapes, et ce qui reste sur le disque.

	bonjour.py, TD 1c	bonjour.cpp
Nombre d'étapes		
Ce qui apparaît dans l'arborescence		
Taille du fichier source	121 octets	230 octets
Taille du fichier produit		

La taille de l'exécutable dépend du compilateur et du système ; le rapport à la source, près de cent fois, n'en dépend pas.

Les deux exécutions dans l'éditeur

Le terminal de l'éditeur garde la trace des trois commandes, et l'arborescence montre le fichier que la compilation vient de produire.



The screenshot shows the Visual Studio Code interface. On the left, the Explorer sidebar shows a project named 'BONJOUR' with a subdirectory 'cpp' containing 'bonjour.cpp' and 'python' containing 'bonjour.py'. The main editor displays the content of 'bonjour.cpp', which is a simple C++ program that prints 'Bonjour, géomatique !'. Below the editor, the TERMINAL panel shows the execution of three commands: a Python command that prints the filename, a g++ compilation command, and a ./cpp/bonjour command that runs the compiled program and prints the output.

```
1 // Le plus petit programme C++ qui fasse quelque chose de visible.
2 #include <iostream>
3 #include <string>
4
5 int main() {
6     std::string nom = "géomatique";
7     std::cout << "Bonjour, " << nom << " !" << std::endl;
8     return 0;
9 }
10
```

```
$ python python/bonjour.py
Bonjour, géomatique !
$ g++ cpp/bonjour.cpp -o cpp/bonjour
$ ./cpp/bonjour
Bonjour, géomatique !
$
```

bonjour n'existait pas avant la deuxième commande. Le programme Python, lui, n'a rien laissé.