✅ Compiler son 1er programme C — Debian 12

class: lead

Introduction pratique, pas-à-pas, avec des diagrammes explicatifs 🧭

🎯 Objectifs

- Créer et compiler un programme simple en C
- Comprendre les étapes (préprocesseur → compilation → assemblage → linking)
- Debugger basiquement et automatiser avec `Makefile`

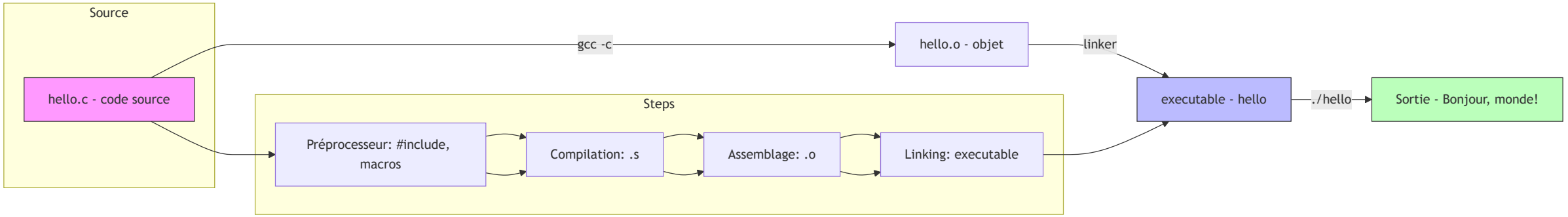
🧰 Prérequis

- Debian 12 (VM ou machine)
- Accès terminal, éditeur (VS Code, nano, vim)
- Connexion internet pour installer paquets

1) Écrire le programme `hello.c`

```c
#include <stdio.h>
int main(void) {
    printf("Bonjour, monde!\n");
    return 0;
}
```

Diagramme : pipeline de compilation 🔧



{.diagram width=70%}

> Ce diagramme montre les étapes du code source jusqu'à l'exécutable.
> *Astuce : ouvrez* `diagrams/compile_workflow.svg` *séparément pour voir les détails si besoin.*

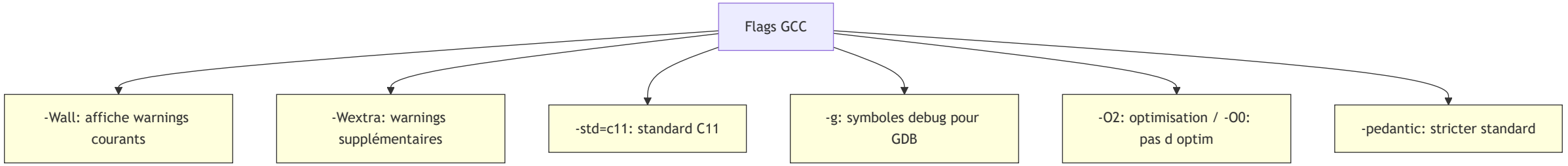
2) Installer les outils

- `sudo apt update`
- Outils de compilation de base : `sudo apt install build-essential make binutils`
- Debug & mémoire : `sudo apt install gdb valgrind`
- Analyse statique / qualité : `sudo apt install cppcheck clang-tidy clang-format`
- Compilateur alternatif (optionnel) : `sudo apt install clang`
- Vérifier l'installation : `gcc --version ; gdb --version ; valgrind --version`

3) Compiler & exécuter

- Compiler : `gcc hello.c -o hello`
- Exécuter : `./hello` → `Bonjour, monde!`

Diagramme : flags GCC & leur utilité ⚙️



> Explication visuelle des flags courants (`-Wall` , `-Wextra` , `-g` , `-O2` , `-std=c11`).

4) Makefile : automatisation

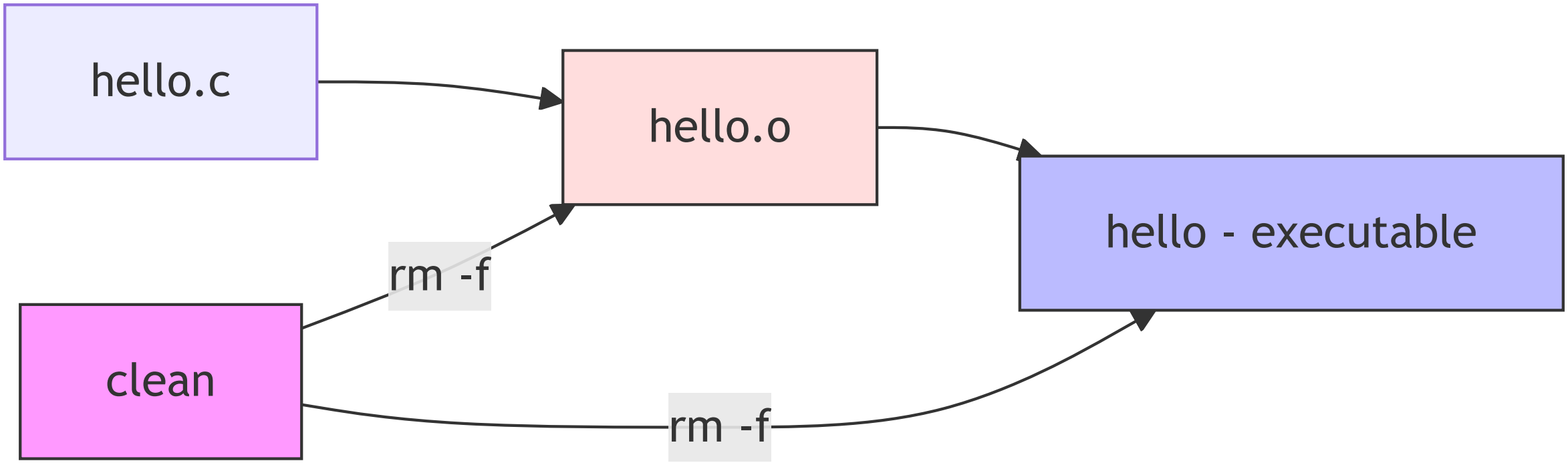
```
CC=gcc
CFLAGS=-Wall -Wextra -std=c11 -g

hello: hello.o
	$(CC) $(CFLAGS) hello.o -o hello

hello.o: hello.c
	$(CC) $(CFLAGS) -c hello.c -o hello.o

clean:
	rm -f hello hello.o
```

Diagramme : flux Makefile 📦

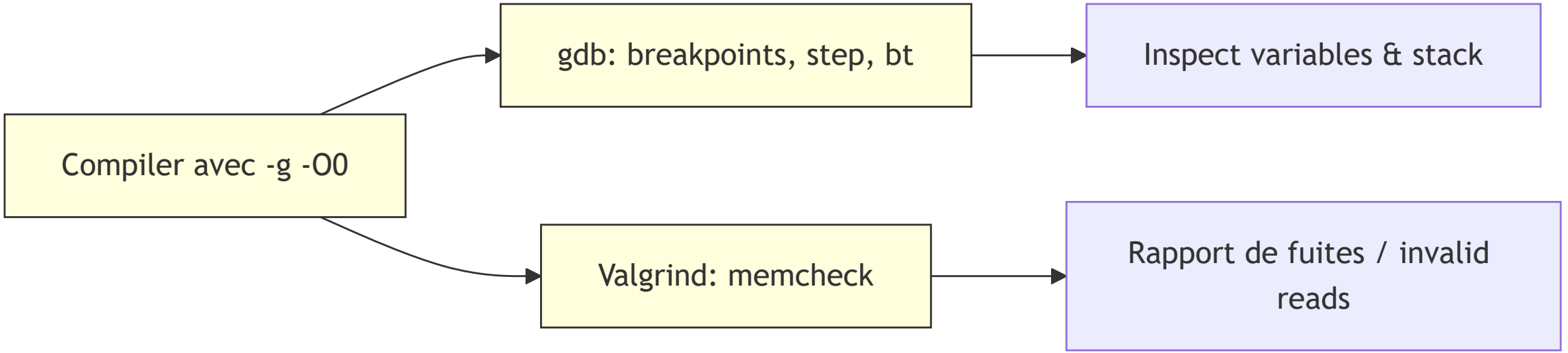


> Visualise les dépendances entre cibles et fichiers objets.
> *Astuce : ouvre* `diagrams/makefile_flow_lr.svg` *séparément pour voir la version détaillée.*

5) Debug & analyse

- Compiler pour debug: `gcc -g -O0 hello.c -o hello`
- Installer si nécessaire : `sudo apt install gdb valgrind cppcheck`
- Lancer `gdb ./hello` — commandes utiles : `break main`, `run`, `step`, `next`, `bt`, `print var`
- Vérifier fuites mémoire : `valgrind --leak-check=full ./hello`
- Analyse statique rapide : `cppcheck --enable=all .`
- Utiliser `clang-tidy` / `clang-format` : `clang-tidy hello.c --` et `clang-format -i hello.c`

Diagramme : debug & outils 🔍



> Résumé visuel : compiler avec `-g`, utiliser `gdb`, puis `valgrind` si besoin.

Exercices pratiques ✅

- Ex1: Lire un entier et afficher son carré
- Ex2: Introduire volontairement un warning et le corriger
- Ex3: Ajouter un `Makefile` et une cible `clean`

Ressources & commandes utiles

- `man gcc`, `man gdb`
- Outils : `valgrind`, `cppcheck`, `clang-tidy`

Fin

Bonne compilation! 🚀