**3eme info B**

**Group 3**

**Med Oussama Braiek**

***Atelier 0: Initiation aux Outils de***

***Développement C sous Linux***

1. **Objectifs :**

- Initiation à l'utilisation de quelques outils de développement avec le langage C sous Linux.

- Développement en C sous Linux à partir d'un terminal (en mode console).

NB: l'utilisation de tout IDE tel que codeblocks est strictement interdite.

1. **Pré-requis :**

Machine virtuelle Linux Ubuntu.

1. **Outils de Développement :**

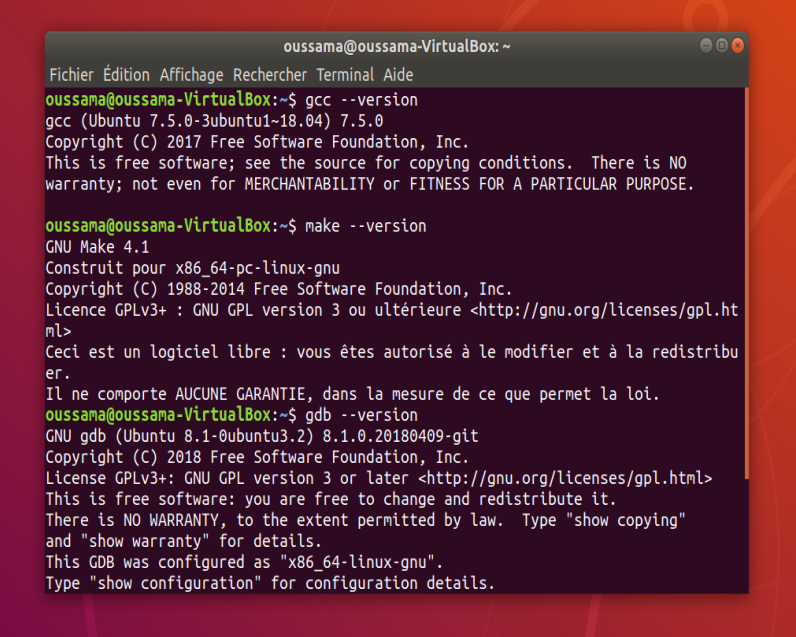
* GCC (GNU Compiler Collection).
* Make (Gestionnaire de Compilation Make).
* GDB (Gnu DeBugger)

1. **Préparation de l'Environnement :**

* Lancer terminal (ctrl + alt + t).
* sudo apt-get update && sudo apt-get install build-essential gdb :

commande pour installe les les outils **GCC**, **Make** et **GDB**.

* gcc - - version
* make - - version
* gdb - - version

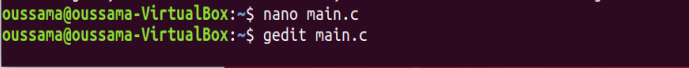


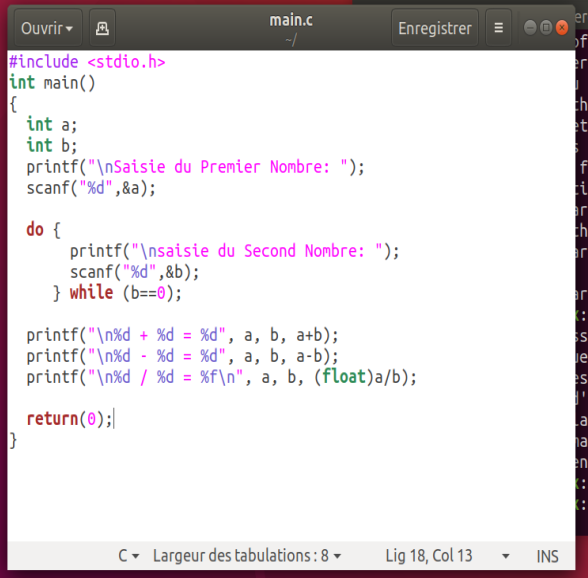
**Figure 1: Versions des Outils GCC, Make et GDB**

1. **Développement C sous Linux :**
   1. **Édition du Code Source :**

On utilise l'éditeur préinstallé **Gedit**.

sudo apt-get installe nano 🡪 pour installer l’éditeur **nano**.

**gedit main.c** 🡪 Éditer un fichier **main.c** .

****

**Figure 2: Code source main.c**

(ctrl + o) pour enregistrer et (ctrl + x) pour quitter l’éditeur.

* 1. **Compilation du Code Source :**

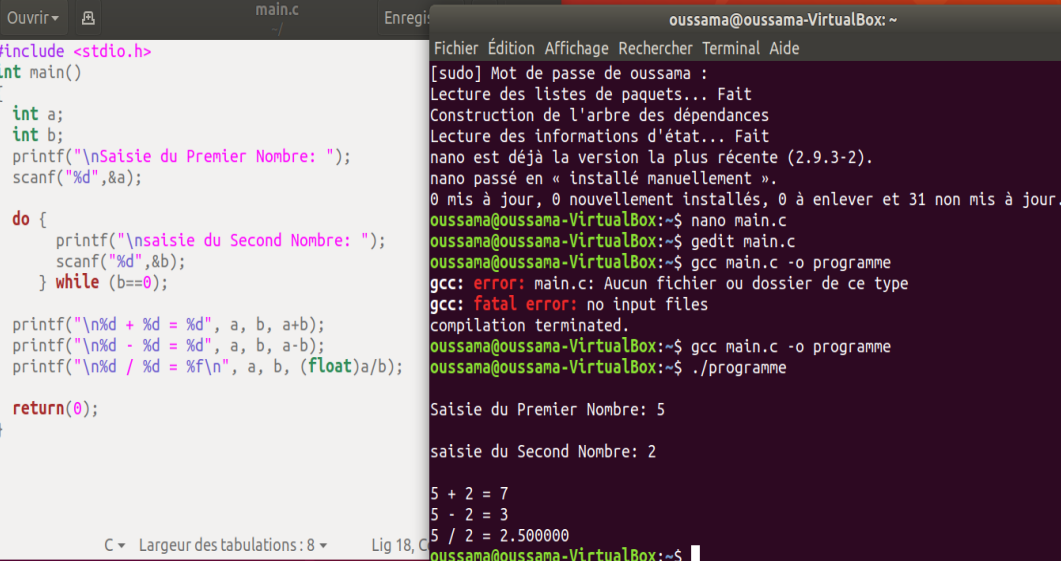
**gcc main.c -o programme** 🡪 Pour compiler le fichier **main.c**.

**- main.c** est le fichier source.

**- programme** est le binaire (exécutable) résultant de l'opération

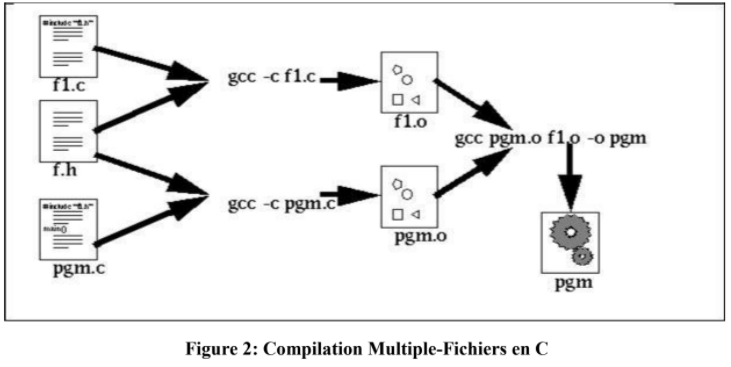
de compilation.

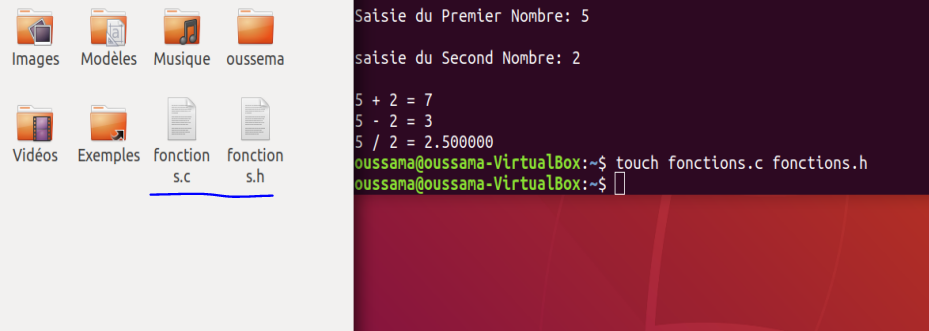
* 1. **Exécution du Fichier Binaire :**

.**/programme** 🡪 Exécution du fichier binaire (programme).

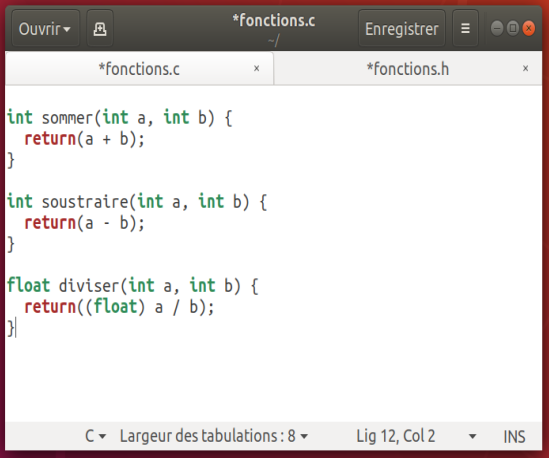
**Figure 3: Exécution**

**6. Compilation de Plusieurs Fichiers Sources :**

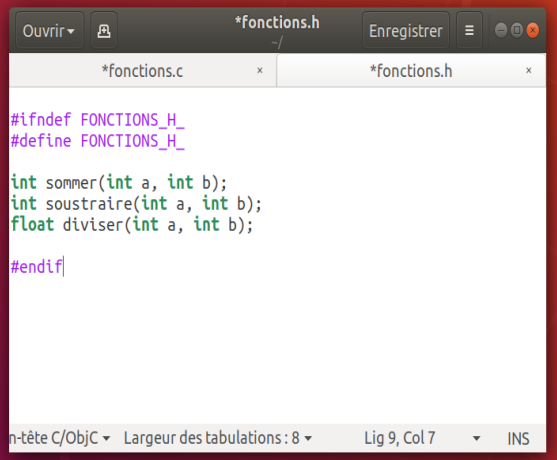


**touch fonctions.c fonctions.h** 🡪 Création de deux nouveaux fichiers nommés fonctions.c et fonctions.h .

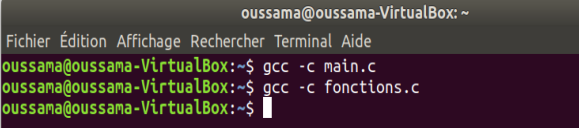
**Figure 4 : Création de deux fichiers**

**fonctions.c :**

**Figure 5: Fonctions.c**

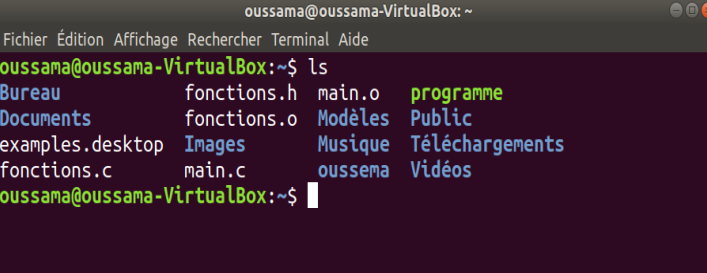
**Fonction.h :**

**Figure 6: Fonction.h**



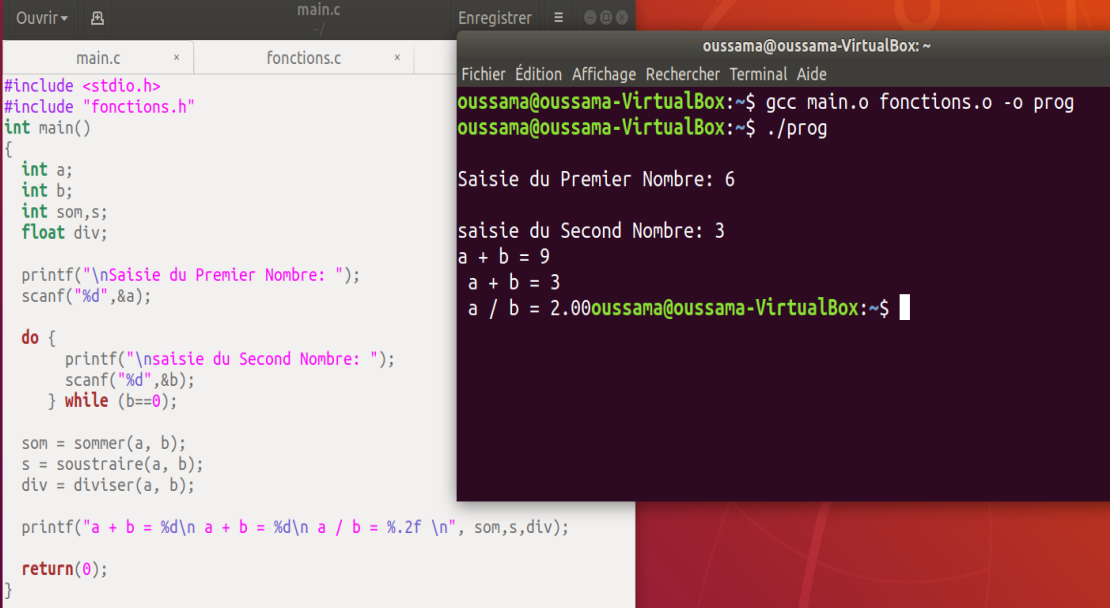
**-c** 🡪 permet de compiler le fichier en question sans passer à la phase d'édition de lien.

**-ls 🡪** lister le contenu du répertoire courant.



**Figure 7 : contenu du répertoire**

**gcc main.o fonctions.o -o prog 🡪** lier les fichiers objets en un seul fichier binaire exécutable (prog) moyennant l'option –o.

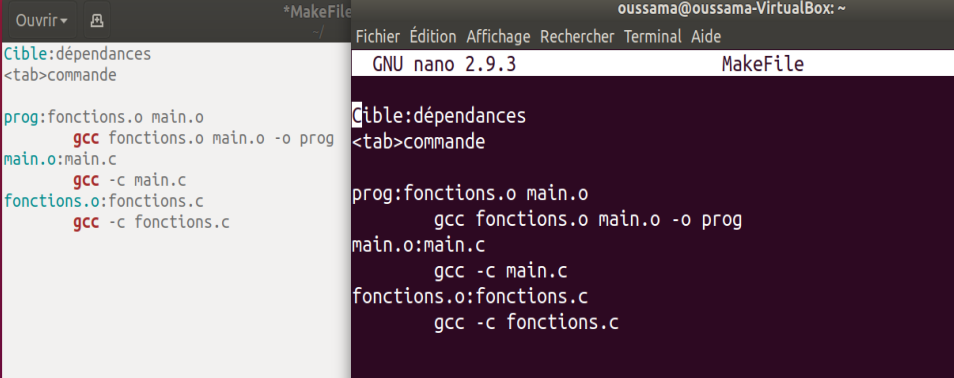


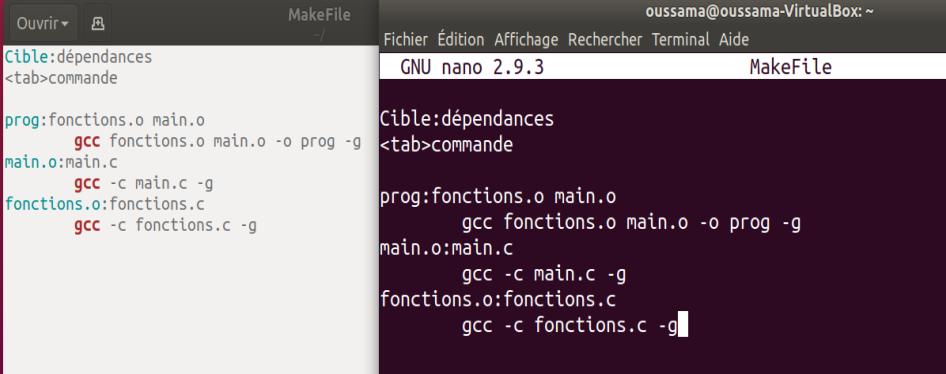
**Figure 8 : Exéction1**

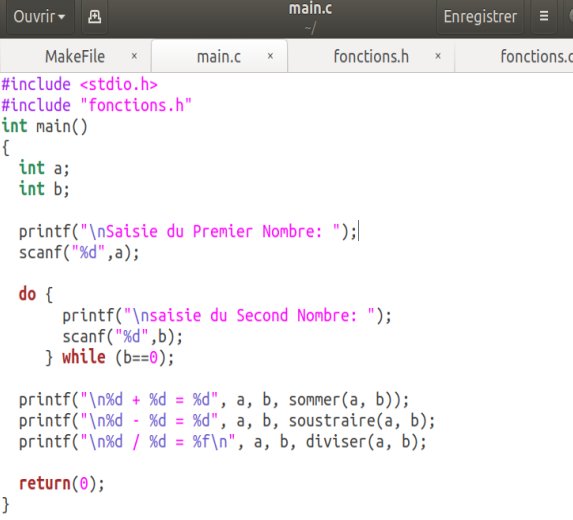
**7. Automatisation de la Compilation sous Linux :**

**7.1. Création du Fichier MakeFile :**

**touch MakeFile nano MakeFile**



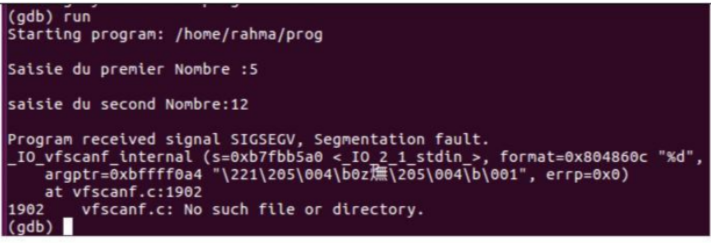
**8. Exécution et Débogage :**



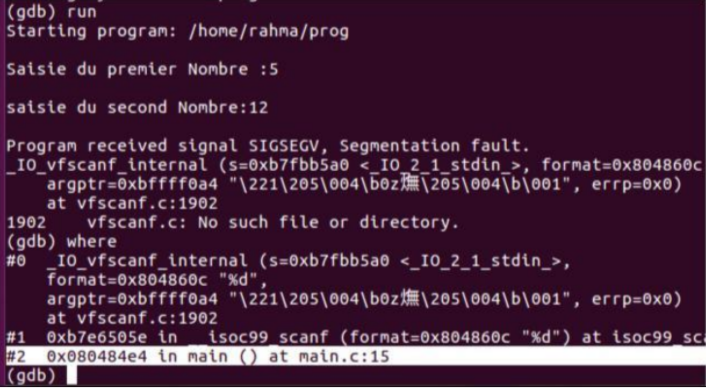
**Figure 9: Bug d'exécution**

**gdb prog 🡪** La seule façon de trouver les erreurs en cours d'exécution est de débogueur le binaire.

(gdb) run



**Figure 10: Opération de Dédogage**

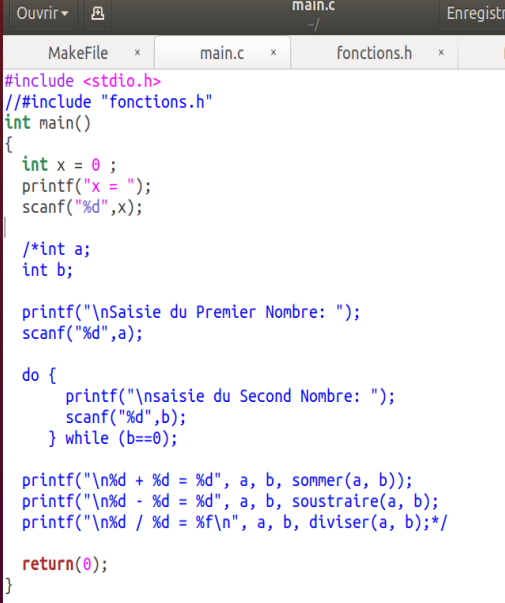
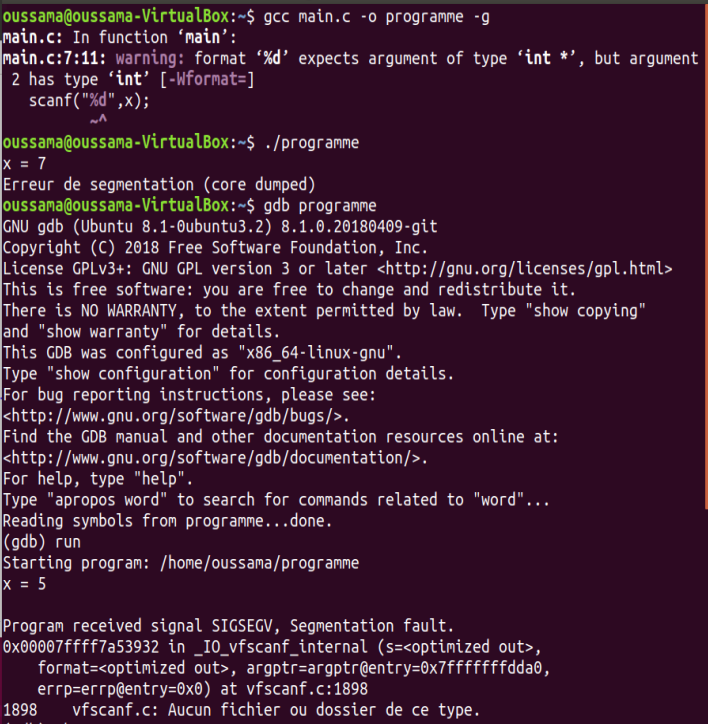
(gdb) where

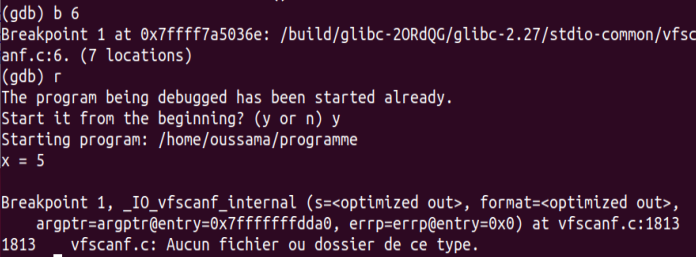
**Figure 11: Localisation du Bug**

**Breakpoints :**

**(gdb) break 3** 🡪 permettra d'arrêter l'exécution à la ligne 3.

**(gdb) b Somme: 6** : permet de faire un breakpoint dans la fonction Somme, à la ligne 6.





**Figure 12: Exemple d'Utilisation des Breappoints:**

Commande 2: **gcc main.c –o programme –g**

Commande 3: **gdb programme**

Commande 4: **(gdb) b 6**

Commande 5: **(gdb) r**

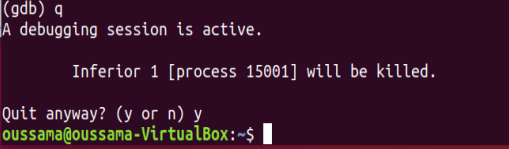
Commande 6**: (gdb) p x 🡪** Examiner l’état de la mémoire à ce moment là.

commande **list 🡪** Remémorer l’endroit dans le code où l’exécution a été interrompue.

**Effacer un Breakpoint:**

Commande 7: **(gdb) clear 6 🡪** indiquant un numéro de ligne ou un nom de fonction permet d'effacer le breakpoint.

La commande **delete** permet d'effacer un breakpoint en indiquant son numéro.

Commande 8: **(gdb) quit.**

Résumé: Quelques commandes importantes sous gdb

(gdb) run ou « r »: lancer le débogage.

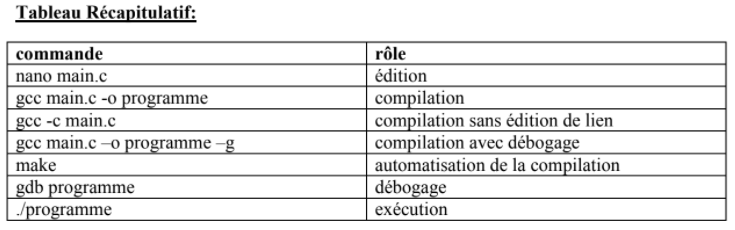
(gdb) quit ou « q »: quitter gdb.

(gdb) break ou « b »: placer des points d'arrêt en précisant le numéro de la ligne ou le nom de

la fonction.

(gdb) print ou « p » : afficher le continue d’une variable.

(gdb) next ou « n » : passer à la ligne suivante.

(gdb) continue ou « c »: relance l’exécution jusqu’au prochain point d’arrêt.