COMPTE-RENDU TP4 RÉSEAU

GDB

<u>Objectif</u> : Prise en main du logiciel GDB

<u>Exercice 1</u> :

3) Même sen le logiciel gdb on peut régler les problème de compilation. En compilant le programme avec les commandes de compilation suivante :
-std=c18 -Wall -Wconversion -Werror -Wextra -Wpedantic -Wfatal-errors -Wwrite-strings -g

On voit que pour les lignes 12, 16, 39 et 59 les formats des variables i et j doivent être déclarées unsigned int au lieu de juste int.

4) Commande de compilation : gcc -o td4_produit td4_produit.c
./ td4_produit < entree.txt
résultat : que des 0 au début et après un core dump, c'ets un résultat inattendue donc il faut debugger le programme.

<u>Exercice 2</u> :
6) Il faut compiler avec -g
Le logiciel gdb nous permet d'éviter de faire des printf pour débugger, on va voir dans l'exécutable ce qu'il se passe
Commande : gdb td4_produit

r < entree.txt => pour lancer le programme
d => supprime les breakpoint
d 8 => supprime le breakpoint n°8
i b => voir la liste des breakpoint
k => arrête le programme
p nb_lignes => affiche la valeur de la variable nb_lignes
u => pour remonter dans le programme principal
d => descendre dans le programme

display lecture_matrice::nb_col ce qui donne le résultat suivant lecture_matrice::nb_col = 0x2

Pour exécuter un programme pas à pas on peut utiliser les commandes step et next. La commande step exécture le programme instruction par instruction et la commande next exécute le programme en sautant les fonctions appelées.

7) Pour voir comment est organiser un exécutable en c, on peut utiliser la commande objdump mais elle ne donne pas les en-têtes, pour cela on utilise la commande readelf --header td4_produit

Les en-têtes de segment sont chargées directement sur la mémoire vive de notre machine : elles permet de savoir les points d'entrée de l'exécutable, les vraies adresses
Les en-tetes de section (.o): dans chaque segment on va regrouper des sections, détail des regroupements des sections dans les segments

Détail de l'en-tête d'un segment principale :
7f 45 4c 46 identifie un fichier au format ELF, 45 4c 46 corresponds au code ASCII de ELF

```
habibah@ummhabibah:~/Documents/Cours/L3/réseau/tp/4$ readelf --header td4_produit
En-tête ELF:
  Magique:   7f 45 4c 46 02 01 01 00 00 00 00 00 00 00 00 00
  Classe:                            ELF64
  Données:                           complément à 2, système à octets de poids faible d'abord (little endian)
  Version:                           1 (actuelle)
  OS/ABI:                            UNIX - System V
  Version ABI:                       0
  Type:                              DYN (fichier objet partagé)
  Machine:                           Advanced Micro Devices X86-64
  Version:                           0x1
  Adresse du point d'entrée:               0x1120
  Début des en-têtes de programme :          64 (octets dans le fichier)
  Début des en-têtes de section :          18568 (octets dans le fichier)
  Fanions:                           0x0
  Taille de cet en-tête:             64 (octets)
  Taille de l'en-tête du programme:  56 (octets)
  Nombre d'en-tête du programme:     13
  Taille des en-têtes de section:    64 (octets)
  Nombre d'en-têtes de section:      36
  Table d'index des chaînes d'en-tête de section: 35
```

Si on fait la commande file td4_produit, regarde d’abord l’identifiant pour savoir quel est le type du fichier

```
habibah@ummhabibah:~/Documents/Cours/L3/réseau/tp/4$ file td4_produit
td4_produit: ELF 64-bit LSB shared object, x86-64, version 1 (SYSV), dynamically linked, interpreter /lib64/ld-linux-x86-64.so.2, B
ildID[sha1]=0d3305a09b569dbf5ef5873986ee4a835938a05a, for GNU/Linux 3.2.0, with debug_info, not stripped
```