

# TOI — TP Débogage (noté)

Rendu avant le 2024-12-XXT20:00,

à john.gliksberg@uvsq.fr ou max.hoffer@uvsq.fr

## Travail attendu

Dans ce TP nous attendons la production d'un rapport au format PDF. Soyez concis et efficaces mais ne renvoyez pas non plus un document en texte brut — trouvez le juste milieu. Le travail peut être fait seul ou en binôme.

Récupération des exercices :

```
git clone https://github.com/trosh/ATOI24_C3TP_debug.git
```

## Débogage mémoire

### Exercice 1 : un bug dans mon Laplacien

Dans le répertoire du cours ex1/ examinez le code fourni et tentez de le lancer. Il semble que cela soit du traitement d'image.

- a. Que fait ce code ?
  - i. Expliquez main.c et les différentes images créées.
  - ii. Expliquez le principe de la convolution et du laplacien dans le traitement d'image.
- b. Écrivez un Makefile simple (un seul binaire) pour ce code.
- c. Le code est-il fonctionnel ?
- d. Quel est le bug et comment l'avez-vous trouvé ?
- e. Corrigez le bug.

### Exercice 2 : le Segfault

Ce code ex2/ semble incorrect. En fait, chacune des fonctions process\_data contient une erreur. Cependant elles ne plantent pas forcément de manière identique.

- a. Expliquer en quoi chacune des fonctions process\_dataXX est fausse.
- b. En commentant et décommentant les différentes fonctions (1,2,3), expliquer l'ampleur des dégâts et les erreurs rencontrées pour ces trois fonctions :

- i. `process_data` : plantage ? explication.
- ii. `process_data1` : plantage ? explication.
- iii. `process_data3` : plantage ? explication.
- c. Quel outil vous permet de vous prémunir des erreurs de `process_data` ?  
Inversement, quel autre outil utilisez vous pour l'erreur de `process_data3` ?
- d. Une corruption mémoire mène-t-elle toujours au plantage et pourquoi ?

### **Exercice 3 : la fuite**

Ce code `ex3/` semble avoir un problème de fuite mémoire.

- a. Expliquez ce qu'est une fuite mémoire et les conséquences possibles.
- b. Trouvez la fuite dans le code avec un outil.
- c. Corrigez la fuite.

## **Débogage logique**

### **Exercice 4 : la somme**

- a. Expliquez ce que fait ce programme.
- b. Le programme fonctionne-t-il ?
- c. Utilisez GDB pour afficher la valeur de `sum` à chaque coup de boucle.
- d. Expliquez le bug et corrigez-le ; donnez la valeur finale de `sum`.

### **Exercice 5 : Quand ça dépasse**

Le code de `ex5/` contient une fonction vulnérable à un overflow.

- a. Expliquez pourquoi ce code est vulnérable
- b. Dessinez l'état de la pile lors de l'appel à `vulnerable_function`.
- c. Comment pouvez-vous déclencher l'erreur ?
- d. Que devez-vous taper dans le buffer pour changer la valeur de la variable `password_is_good` afin d'afficher « Vous avez cassé le MDP ! » ?

### **Exercice 6 : Condition d'arrêt**

Le code de `ex6/` contient une erreur qui mène à un crash.

- a. Utilisez GDB pour trouver l'erreur.
- b. Quelle est l'erreur ?
- c. Corrigez l'erreur.

## Exercice 7 : Ça n'avance plus ?

Le code ex7/ ne se termine jamais.

- a. Expliquez ce que fait ce code :
  - i. Qu'est-ce qu'un thread ?
  - ii. Qu'est-ce qu'un mutex ?
  - iii. Que fait le code ?
- b. Écrivez un makefile simple pour ce code.
  - i. Dépend-il d'une bibliothèque système ? Laquelle ?
- c. Pourquoi le code ne se termine-t-il jamais ?
  - i. Comment afficher l'état du programme avec GDB ?
  - ii. Mettre des captures/copies dans le rapport de cet état.
  - iii. Que fait la commande `thread apply all bt` ?
- d. Comment corriger ce code ?