

Tony Marteau

Dimitri Bernot

Marvin Jean

16 novembre 2015

# Compte Rendu du Solver de Ruzzle

## Projet d'Algo & Prog Avancé

L'objectif du projet d'Algo & Prog Avancé, le solver de ruzzle, est de faire un programme en C capable de trouver tout les mots d'une grille de jeu de quatre lignes par quatre colonnes renseignées par le joueur dans un temps imparti. Un mot est valable lorsque qu'il fait 2 lettres ou plus et moins de 16 lettres, sachant que chaque lettre de la grille ne peut être utilisée qu'une seule fois dans un mot.

Le programme est découpé en 2 grandes parties, la première qui consiste à faire initialiser la grille de jeu par le joueur et d'afficher ensuite la grille de jeu et la deuxième à chercher tout les mots d'une grille à l'aide d'un dictionnaire sérialisé.

Pour l'initialisation de la grille de jeu, le joueur doit renseigner la lettre, sa valeur en point et si elle possède un bonus. Toutes ces informations sont stockées dans une structure de données `t_donnee` avec un entier passage qui servira lorsque le programme cherche les mots dans la grille, lors de la saisie, si l'utilisateur se trompe en renseignant une lettre à la place des points par exemple, le programme redemande à l'utilisateur les points de la lettre, de même lorsqu'il renseigne une chiffre à la place d'une lettre. Concernant l'affichage de la matrice de jeu, il s'agit de deux boucles for imbriquées qui affichent les lettres de la grille de jeu.

Pour la deuxième partie, celle qui cherche les mots du dictionnaire dans la grille, la première fonction qui est appelé `chercheMot()`, elle s'occupe d'initialiser les valeurs de passage de la grille à 0, ensuite elle sort les mots du dictionnaire caractère par caractère et lorsque la boucle arrive à un `\n`, elle appelle la fonction `chercherMotGrille()` en passant en

paramètre le dit mot. Cette fonction `chercherMotGrille()` va dans un premier temps chercher dans la grille la première lettre du mot passé en paramètre, si la première lettre est trouvée, les coordonnées de cette dernière vont être mémorisées pour pouvoir ensuite chercher dans les lettres adjacentes la lettre suivante du mot, une boucle va ensuite s'occuper de chercher dans les lettres suivantes du mot recherché, les passages sont remis à 0 une fois le traitement terminé pour le mot suivant, si le mot est présent il est affiché avec son score.

L'algorithme est capable de trouver une majorité des mots dans la grille mais pas tous, en effet, lorsque une lettre possède plusieurs même lettres adjacentes, le programme va vérifier seulement la première lettre.

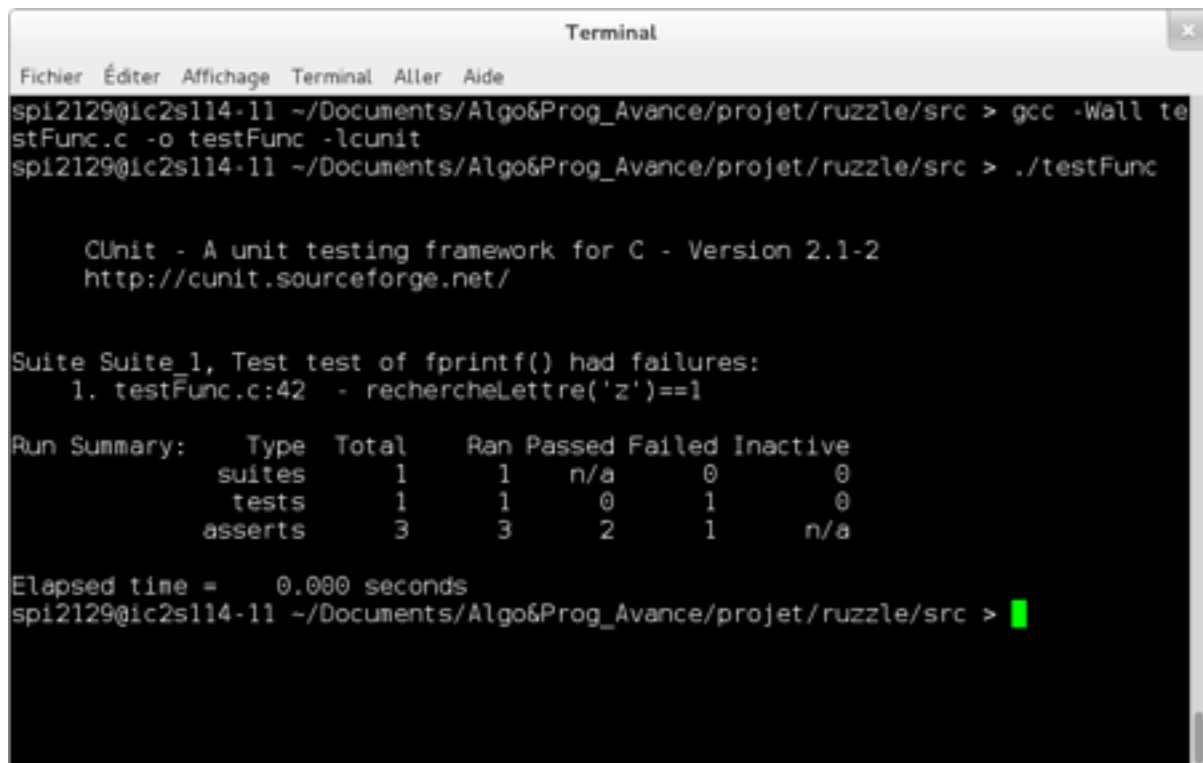
Dépôt GitHub : <https://github.com/Hactogeek/ruzzle>

# CUnit (Test Unitaire)

Test unitaire sur la recherche d'une lettre sur la grille de jeu, 3 tests, lettre c présente et z pas présente.

```
testFunc.c (/Documents/Algo&Prog_Avance/projet/ruzzle/src) - ge8it
Fichier Édition Affichage Rechercher Outils Documents Aide
Ouvrir Enregistrer Annuler
rechercheGrille.c testFunc.c main.c maxi_test.c makefile
1 #include "CUnit/CUnit.h"
2 #include "CUnit/Basic.h"
3 #include "../includes/general.h"
4
5 int rechercheLettre(char c)
6 {
7     int i, j;
8
9     bool trouve = false;
10
11     /* On parcourt les lettres adjacentes à la lettre précédente */
12
13     for(i=-1; i<=1 && !trouve; i++)
14     {
15         for(j=-1; j<=1 && !trouve; j++)
16         {
17
18             /* Vérification si les coordonnées sont bien dans la grille de jeu */
19
20             if(coordonnee.x+i>=0 && coordonnee.x+i<N && coordonnee.y+j>=0 && coordonnee.y+j<N)
21             {
22
23                 /* Vérification si la lettre correspond et si on n'est pas passé sur la case */
24
25                 if(c == grille[coordonnee.x+i][coordonnee.y+j].lettre && grille[coordonnee.x+i][coordonnee.y+j].passage == 0)
26                 {
27                     trouve = true;
28                     coordonnee.x=coordonnee.x+i;
29                     coordonnee.y=coordonnee.y+j;
30                     grille[coordonnee.x][coordonnee.y].passage=1;
31                     return 1;
32                 }
33             }
34         }
35     }
36     return 0;
37 }
38
39 void testRechercheLettre(void)
40 {
41     CU_ASSERT (rechercheLettre('c')==1);
42     CU_ASSERT (rechercheLettre('z')==1);
43     CU_ASSERT (rechercheLettre('x')==0);
44 }
45
46 int main()
47 {
48     grille[0][0].lettre = 'c';
49 }
```

Résultat des tests unitaire, 2 sont passées correctement et 1 a raté.



```
Terminal
Fichier Éditer Affichage Terminal Aller Aide
spi2129@ic2s114-11 ~/Documents/Algo&Prog_Avance/projet/ruzzle/src > gcc -Wall testFunc.c -o testFunc -lcunit
spi2129@ic2s114-11 ~/Documents/Algo&Prog_Avance/projet/ruzzle/src > ./testFunc

CUnit - A unit testing framework for C - Version 2.1-2
http://cunit.sourceforge.net/

Suite Suite_1, Test test of fprintf() had failures:
1. testFunc.c:42 - rechercheLettre('z')==1

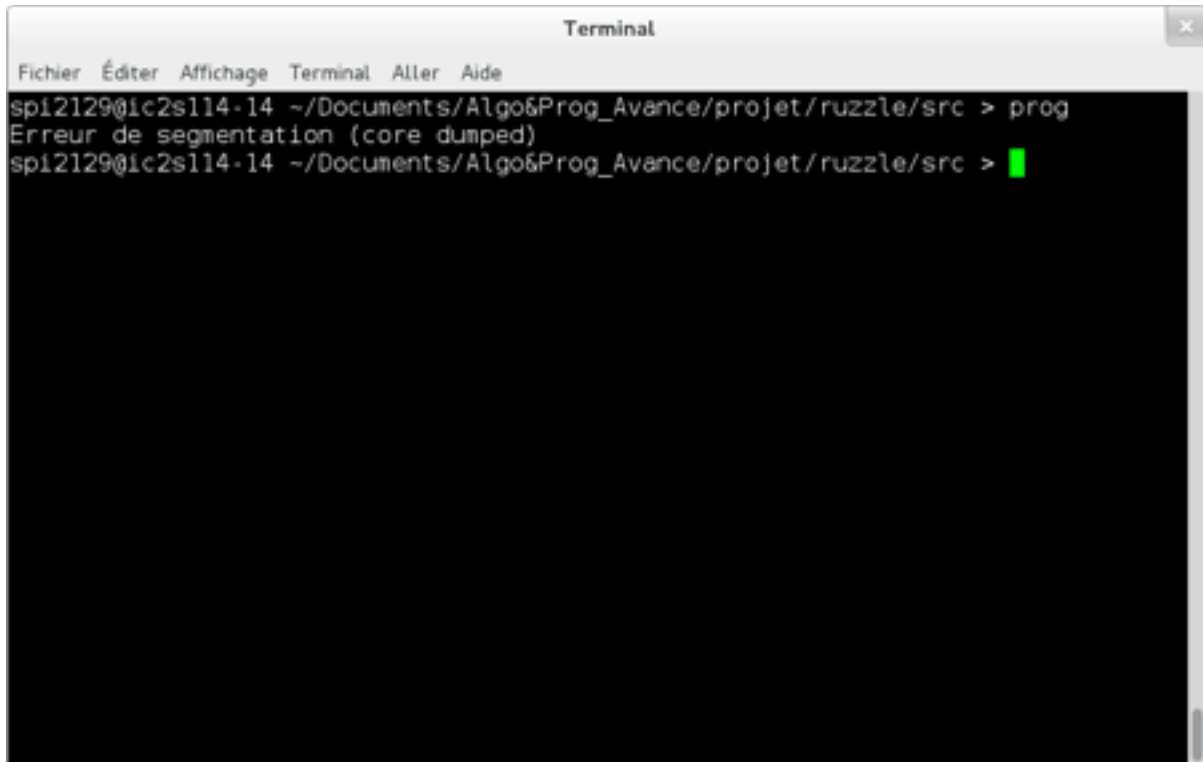
Run Summary:
  Type      Total   Ran Passed Failed Inactive
  suites      1      1    n/a      0        0
  tests       1      1     0       1        0
  asserts     3      3     2       1      n/a

Elapsed time = 0.000 seconds
spi2129@ic2s114-11 ~/Documents/Algo&Prog_Avance/projet/ruzzle/src > █
```

La recherche de lettre dans la grille passe le test unitaire facilement.

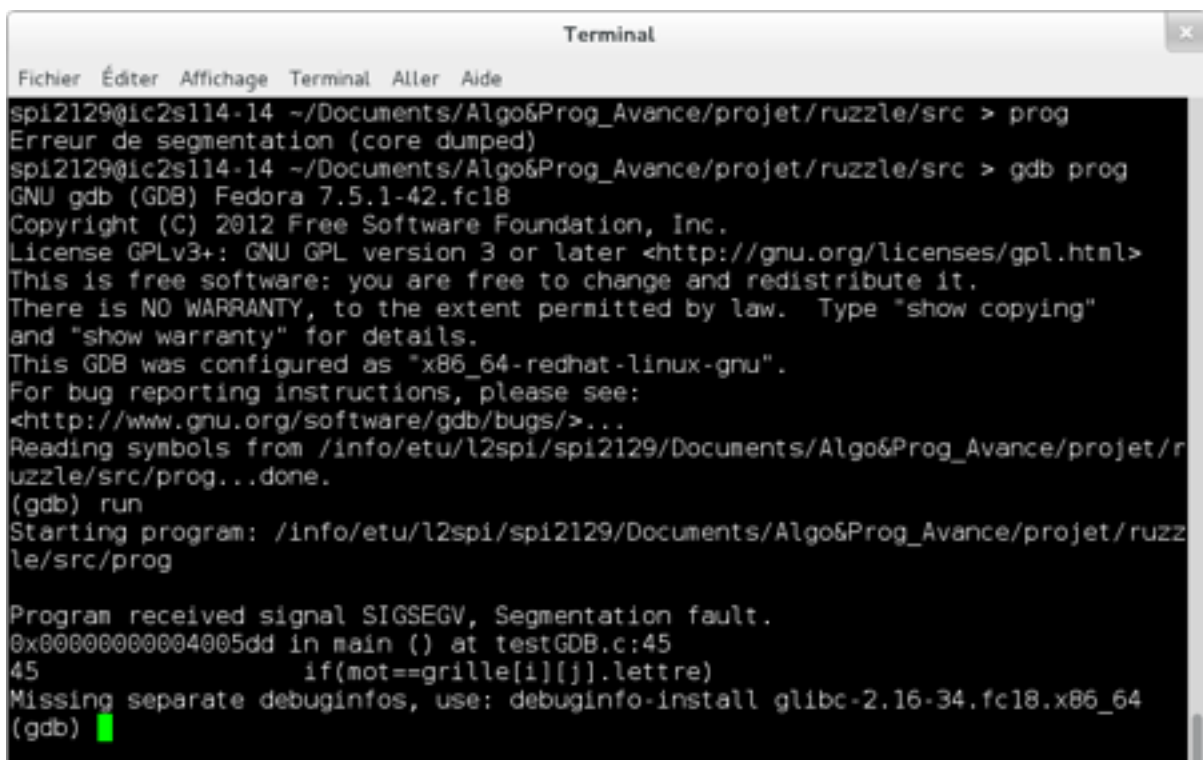
# Débogage(GDB)

On a une erreur dans le programme, une erreur de segmentation, nous allons chercher l'erreur.



```
Terminal
Fichier Éditer Affichage Terminal Aller Aide
spi2129@ic2s114-14 ~/Documents/Algo&Prog_Avance/projet/ruzzle/src > prog
Erreur de segmentation (core dumped)
spi2129@ic2s114-14 ~/Documents/Algo&Prog_Avance/projet/ruzzle/src > █
```

On lance GDB



```
Terminal
Fichier Éditer Affichage Terminal Aller Aide
spi2129@ic2s114-14 ~/Documents/Algo&Prog_Avance/projet/ruzzle/src > prog
Erreur de segmentation (core dumped)
spi2129@ic2s114-14 ~/Documents/Algo&Prog_Avance/projet/ruzzle/src > gdb prog
GNU gdb (GDB) Fedora 7.5.1-42.fc18
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /info/etu/l2spi/spi2129/Documents/Algo&Prog_Avance/projet/ruzzle/src/program...done.
(gdb) run
Starting program: /info/etu/l2spi/spi2129/Documents/Algo&Prog_Avance/projet/ruzzle/src/program

Program received signal SIGSEGV, Segmentation fault.
0x00000000004005dd in main () at testGDB.c:45
45             if(mot==grille[i][j].lettre)
Missing separate debuginfos, use: debuginfo-install glibc-2.16-34.fc18.x86_64
(gdb) █
```

On sait à peu près le lieu du problème, on place donc un point d'arrêt sur la ligne en question 45 pour l'exemple.

```
Terminal
Fichier Éditer Affichage Terminal Aller Aide
Erreur de segmentation (core dumped)
spi2129@ic2s114-14 ~/Documents/Algo&Prog_Avance/projet/ruzzle/src > gdb prog
GNU gdb (GDB) Fedora 7.5.1-42.fc18
Copyright (C) 2012 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <http://gnu.org/licenses/gpl.html>
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law. Type "show copying"
and "show warranty" for details.
This GDB was configured as "x86_64-redhat-linux-gnu".
For bug reporting instructions, please see:
<http://www.gnu.org/software/gdb/bugs/>...
Reading symbols from /info/etu/l2spi/spi2129/Documents/Algo&Prog_Avance/projet/ruzzle/src/prog...done.
(gdb) run
Starting program: /info/etu/l2spi/spi2129/Documents/Algo&Prog_Avance/projet/ruzzle/src/prog

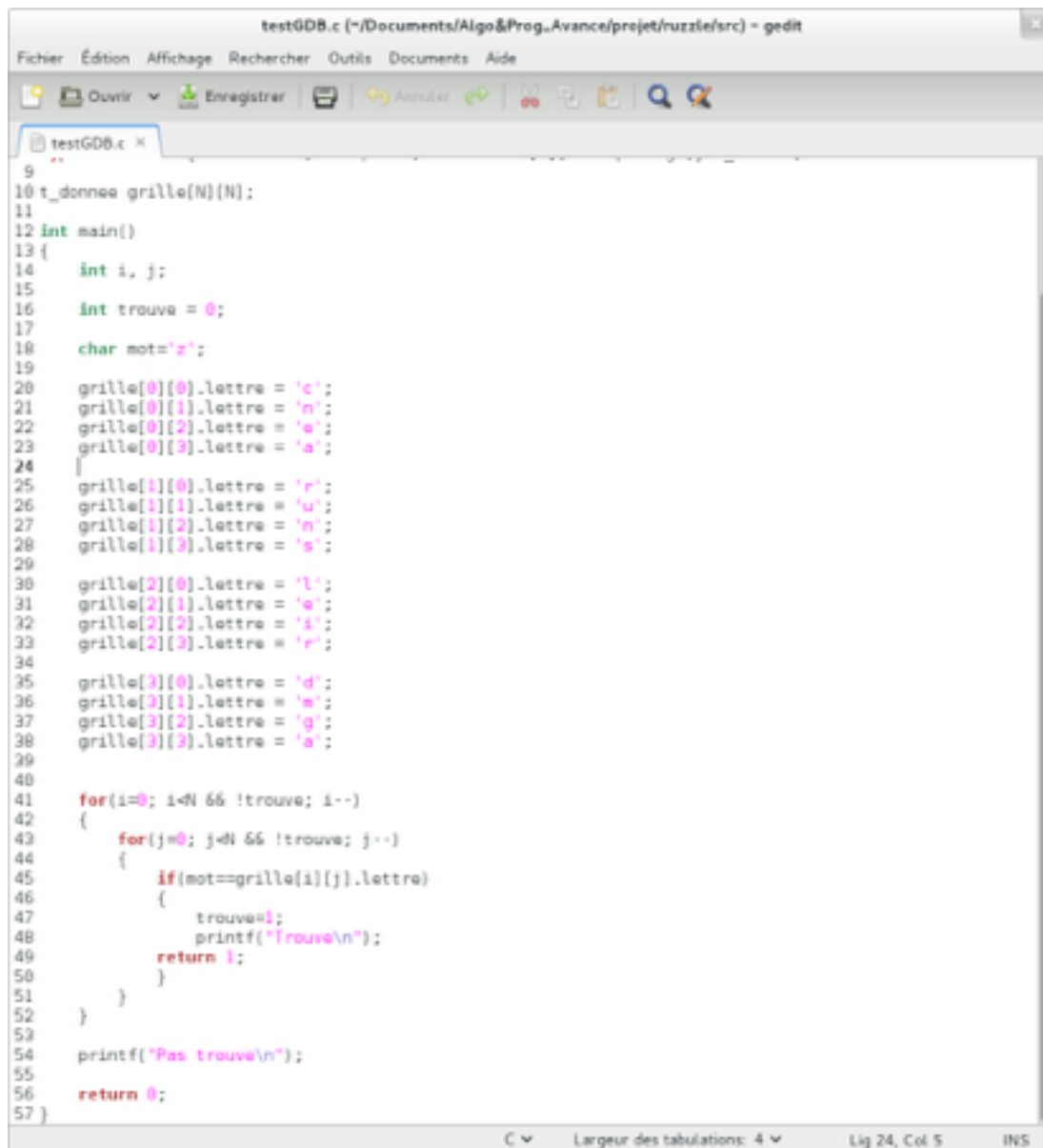
Program received signal SIGSEGV, Segmentation fault.
0x00000000004005dd in main () at testGDB.c:45
45         if(mot==grille[i][j].lettre)
Missing separate debuginfos, use: debuginfo-install glibc-2.16-34.fc18.x86_64
(gdb) break 45
Breakpoint 1 at 0x4005c1: file testGDB.c, line 45.
(gdb)
```

On regarde maintenant l'évolution des variables présente

```
Terminal
Fichier Éditer Affichage Terminal Aller Aide
le/src/prog

Breakpoint 1, main () at testGDB.c:45
45         if(mot==grille[i][j].lettre)
Missing separate debuginfos, use: debuginfo-install glibc-2.16-34.fc18.x86_64
(gdb) print i
$1 = 0
(gdb) print j
$2 = 0
(gdb) step
43         for(j=0; j<N && !trouve; j--)
(gdb) print i
$3 = 0
(gdb) print j
$4 = 0
(gdb) step
Breakpoint 1, main () at testGDB.c:45
45         if(mot==grille[i][j].lettre)
(gdb) print i
$5 = 0
(gdb) print j
$6 = -1
(gdb)
```

On constate qu'une variable qui doit toujours être supérieur ou égal à 0, prend la valeur -1, on a trouvé notre erreur.



```
testGDB.c (~Documents/Algo&Prog_Avance/projet/ruzzle/src) - gedit
Fichier  Édition  Affichage  Rechercher  Outils  Documents  Aide

testGDB.c x
..
9
10 t_donnee grille[N][N];
11
12 int main()
13 {
14     int i, j;
15
16     int trouve = 0;
17
18     char mot='x';
19
20     grille[0][0].lettre = 'c';
21     grille[0][1].lettre = 'n';
22     grille[0][2].lettre = 'a';
23     grille[0][3].lettre = 'a';
24
25     grille[1][0].lettre = 'r';
26     grille[1][1].lettre = 'u';
27     grille[1][2].lettre = 'n';
28     grille[1][3].lettre = 's';
29
30     grille[2][0].lettre = 'l';
31     grille[2][1].lettre = 'e';
32     grille[2][2].lettre = 'i';
33     grille[2][3].lettre = 'r';
34
35     grille[3][0].lettre = 'd';
36     grille[3][1].lettre = 'm';
37     grille[3][2].lettre = 'g';
38     grille[3][3].lettre = 'a';
39
40     for(i=0; i<N && !trouve; i++)
41     {
42         for(j=0; j<N && !trouve; j++)
43         {
44             if(mot==grille[i][j].lettre)
45             {
46                 trouve=1;
47                 printf("Trouve\n");
48                 return i;
49             }
50         }
51     }
52     printf("Pas trouve\n");
53
54     return 0;
55
56 }
57 }
```

Dans les boucles for, on enlève 1 à chaque itération au lieu d'ajouter 1.