TEHARI Ayoub
L3 informatique
INF1

03 Février 2020

Compte Rendu MOCA – séance 3

Les modifications apporté aux fichiers :

“monpg.c”	#include "messages.h"
“outils.c”	#include "messages.h"

Explication des options de compilation :

gcc -Wall -c src/outils.c -Iinclude/ -I lib_messages/include/	l’option -c pour s’arrêter à l’étape compilation et générer le “.o” l’option -I pour indiquer au préprocesseur le répertoire pour chercher le “header” inclue. l’option -l pour indiquer le nom de la bibliothèque partagée, et -L permet de spécifier le répertoire où se trouve la bibliothèque.
gcc -Wall -c src/convertir.c -I include/	
gcc -Wall -c src/monpg.c -Iinclude/ -I lib_messages/include/	
gcc convertir.o outils.o monpg.o -o prog -lmessages -Llib_messages/lib/	

l’option -o : pour préciser le nom du fichier produit.
l’option -Wall : pour avoir le maximum des warning.

Pour l’exécution, il faudrait définir (ou étendre si elle existe) la variable d’environnement ‘LD_LIBRARY_PATH’
par cette commande :

```
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$(pwd)/lib_messages/lib/
```

Cette variable permet au shell de retrouver notre librairie, puisque nous lui indiquons le chemin où elle se trouve.

LE PROJET (la reprise)
PARTIE 1

1) Explication autour des warning :

le premier warning : utilisation de la fonction setCharnum avant sa declaration.
2eme warning : une variable locale non utilisée.
3eme warning : le compilateur ne sait pas si la fonction ’getSizeMaillon’ va retourner une valeur à la fin ou pas, puisque dans le ‘else’ il n’ya pas de ‘return’ alors que la fonction doit returner un type ‘int’

2)Les modifications faites dans les sources (#etape2) :

Warning 1 : échange de positionnement	void setCharnum(maillon_t* link, int k, char c)
Warning 2 : int j;	-------
Warning 3 : getSizeMaillon	return res;
Execution : Segmentation fault	if (f == NULL) { printf("Erreur d'ouverture %s\n",DICORES); f=stdout;}

Nom de fichier donné en paramètre	int main(int argc, char **argv)
	{ FILE* f = NULL;
	if (argc != 2) {
	printf("usage du %s est : <%s> <fichier_d_entree>\n",argv[0], argv[0]);
	}else {
	if ((f = fopen(argv[1], "r"))== NULL) {
	printf("Erreur de lecture fichier : %s\n", argv[1]);
	}else { ...

Partie 2

Le Makefile :

```
#les declarations des variables
CC=gcc
CFLAGS=-Wall
#test pour produire ou non la bibliotheque dynamique
ifdef N
CFLAGS_MAIN=$(CFLAGS) -fPIC
else
CFLAGS_MAIN=$(CFLAGS)
endif

PROG=projet2
PROG_S_LIB=projet2_S_LIB
PROG_D_LIB=projet2_D_LIB
PROGS=$(PROG_S_LIB) $(PROG_D_LIB) $(PROG)
OBJDIR=objs
INCDIR=include
SRCDIR=src

#les règles qui génèrent les programmes
all: $(PROG)
all_S_library: $(PROG_S_LIB)
	export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$(PWD)/
all_D_library: $(PROG_D_LIB)
	export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$(PWD)/

#règle pour générer les objets
$(OBJDIR)/%.o : $(SRCDIR)/%.c $(INCDIR)/%.h $(INCDIR)/types.h
	$(CC) $(CFLAGS_MAIN) -c $< -I $(INCDIR) -o $@

#fichier qui contient le main
$(OBJDIR)/dico.o : $(SRCDIR)/dico.c $(INCDIR)/words.h $(INCDIR)/maillons.h $(INCDIR)/display.h
	$(CC) $(CFLAGS) -c $< -I $(INCDIR) -o $@

#règle sans avoir besoin des bibliothèque
$(PROG) : $(OBJDIR)/dico.o $(OBJDIR)/words.o $(OBJDIR)/maillons.o $(OBJDIR)/display.o
	$(CC) -o $(PROG) $^
```

```
#règle de creation de la bibliothèque statique
libutilitaires.a : $(OBJDIR)/words.o $(OBJDIR)/maillons.o $(OBJDIR)/display.o
	ar r $@ $^
	ranlib $@

#règle de creation de la bibliothèque dynamique
libutilitairesD.so : $(OBJDIR)/words.o $(OBJDIR)/maillons.o $(OBJDIR)/display.o
	$(CC) -shared -o $@ $^

#règle de l'édition des liens au cas d'invocation des bibliothèques
$(PROG_S_LIB) : $(OBJDIR)/dico.o libutilitaires.a
	$(CC) -o $(PROG_S_LIB) $< -Wl,-Bstatic -l utilitaires -L .
#Option -Wl,-Bstatic pour forcer un link statique
#option -Wl,-Bdynamic pour forcer un link dynamique
$(PROG_D_LIB) : $(OBJDIR)/dico.o libutilitairesD.so
	$(CC) -o $(PROG_D_LIB) $< -Wl,-Bdynamic -l utilitairesD -L .

clean :
	@echo " deleting object files, libraries and programs"
	-rm $(OBJDIR)/*.o $(PROGS) *.so *.a
```

Pour compiler avec la bibliothèque dynamique :

$ make N= valeur all_D_library

les bibliothèques et les éxécutable seront mis à la racine du projet.

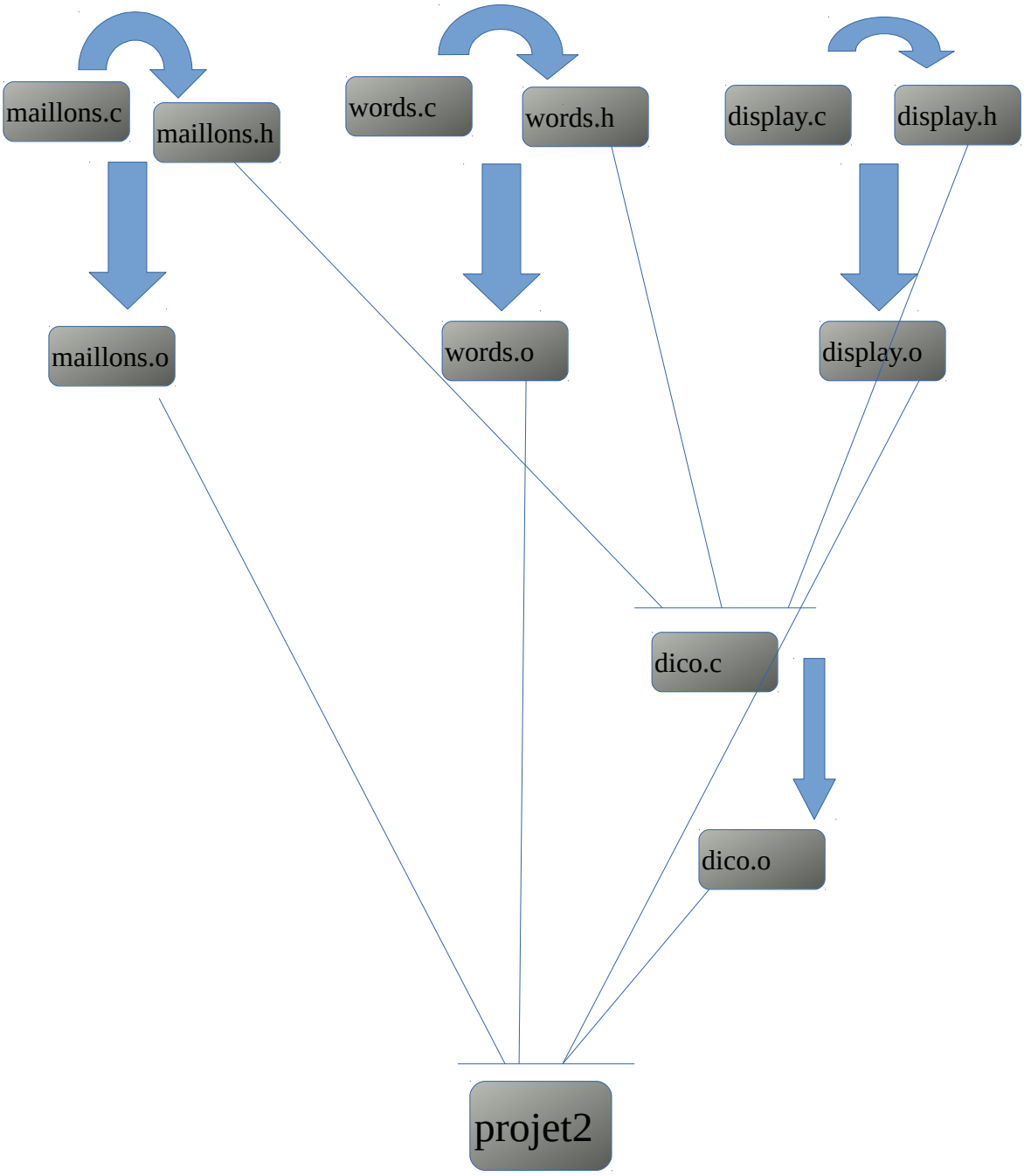
Explication de la modularité choisit :

1. maillons : ce module contient tout ce qui conserne la structure « maillon » et ses alentours : les conversions vers string etc ..
2. words : ce module contient les fonctions : d'ajout des mots aux dictionnaire, lire un nouveau mot , fct pour effectuer les comparaisons entre mots etc ..
3. display : ce module se base sur l'affichage à l'écran.
4. Dico : le program princial qui contient la fonction main.

le repertoire du projet contient :

1. le répertoire include : contient les headers.
2. Le répertoire objs : contient les fichiers objets.
3. Le répertoire src : contient les fichiers source.
4. Le répertoire tests : contient les fichiers de test.
5. Un fichier Makefile pout compiler.

Le shemas de la modularité :



TEHARI Ayoub
L3 informatique
INF1

11 Février 2020

Compte Rendu MOCA – séance 4

Objectif de la séance :

1. utilisation des outils CuTest et gcov.
2. Implémentation des tests / refactoring.
3. Les explications des choix.

Installation de CuTest :

les modification apportées au code :
le module dico : contient précedement la fonction main, qui elle renommé en dico, pour pouvoir créer un nouveau module principale(nommé main) qui contiendras la fonction main, et qui appel selon la valeur de FINAL soit au programme des tests, soit au programme principal.

Évolution du Makefile :

```
#règle pour générer les objets
$(OBJDIR)/%.o : $(SRCDIR)/%.c $(INCDIR)/%.h $(INCDIR)/types.h
	$(CC) $(CFLAGS_MAIN) -c $< -I $(INCDIR) -o $@
```

```
#fichier qui contient le main
$(OBJDIR)/main.o : $(SRCDIR)/main.c $(INCDIR)/dico.h $(INCDIR)/AllTests.h $(INCDIR)/CuTest.h
	$(CC) $(CFLAGS) -c $< -I $(INCDIR) -o $@
$(OBJDIR)/AllTests.o : $(SRCDIR)/AllTests.c $(INCDIR)/AllTests.h $(INCDIR)/SuiteDeTests.h $(INCDIR)/CuTest.h
#règle sans avoir besoin des bibliothèque
$(PROG) : $(OBJDIR)/main.o $(OBJDIR)/dico.o $(OBJDIR)/words.o $(OBJDIR)/maillons.o $(OBJDIR)/display.o $(OBJDIR)/AllTests.o $
(OBJDIR)/CuTest.o $(OBJDIR)/SuiteDeTests.o
	$(CC) -o $(PROG) $^
```

J’ai crée un fichier header pour le module SuiteDeTest, pour permettre la compilation avec la règle générique écrite ci-dessus, ce fichier contient :

```
#ifndef SUITEDETESTS_H
#define SUITEDETESTS_H
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <stdint.h>
#include "CuTest.h"
#include "maillons.h"
```

```
#include "words.h"

char *StrToUpper (char *str);

void PremierTestPourJouerAvecFrameworkCuTest (CuTest * tc);

void ConvesionMaillonStringTest (CuTest * tc);

void GetSizeMaillonTest (CuTest *tc);

void CompareWordTest (CuTest *tc) ;

CuSuite *MaTestSuite ();

#endif //ALLTEST_H
```

Les Tests unitaires :

void ConvesionMaillonStringTest (CuTest * tc)
cette fonction vérifie l'application successive de maillonToString et stringToMaillon
Cas 1: chaine vide, ce cas est important pour savoir si l'unité fonctionne avec taille égale à zero
Cas 2: chaine de taille qui occupe un seul maillon,
Cas 3: chaine de taille qui se repose sur plusieurs maillons chainnés

void GetSizeMaillonTest (CuTest *tc)
vérifie l'application de getSizeMaillon sur stringToMaillon
Cas 1: chaine vide, ce cas est important pour savoir si l'unité fonctionne avec taille égale à zero
Cas 2: chaine de taille qui occupe un seul maillon,
Cas 3: chaine de taille qui se repose sur plusieurs maillons chainnés

void CompareWordTest (CuTest *tc)
utilise generateMot_t pour générer les deux mots de type mot_t test des cas limites(mot=NULL || "" || taille(mot) > 6) en résonnant par la valeur retourné comme résultat.
Cas 1: résultat = -1 : mot1<mot2 ; mot1==NULL = 3 cas.
Cas 2: résultat = 0 : mot1=mot2 ; mot1==mot2==NULL = 3 cas.
Cas 3: résultat=1 : mot1>mot2 ; mot2==NULL = 3 cas.

Correction :

```
int compareWord(mot_t* w1, mot_t* w2) {

  if ((w1 == NULL)&&(w2 == NULL)) return 0;

  if (w1 == NULL) {

    return -1;

  } else if (w2 == NULL) {

    return 1;                    ...
```

Couverture de tests (gcov) :

modification du Makefile :

```
ifdef gcov
CFLAGS=-Wall -fprofile-arcs -ftest-coverage
else
CFLAGS=-Wall
endif
```

```
#règle sans avoir besoin des bibliothèque
$(PROG) : $(OBJDIR)/main.o $(OBJDIR)/dico.o $(OBJDIR)/words.o $(OBJDIR)/maillons.o $(OBJDIR)/display.o $(OBJDIR)/AllTests.o $
(OBJDIR)/CuTest.o $(OBJDIR)/SuiteDeTests.o
	$(CC) -fprofile-arcs -ftest-coverage -o $(PROG) $^
```

Pour l'utilisation, il vas falloir que je déplace les fichiers .gcno et .gcda dans le répertoire src/, pour pouvoir consulter les informations de couverture :
Exemple d'éxécusion sur hugo1.txt :

```
$ gcov -b -c *.c
```

```
File 'src/main.c'
Lines executed:100.00% of 3
No branches
Calls executed:100.00% of 1
Creating 'main.c.gcov'
Cannot open source file src/main.c
.
.
.
File 'src/words.c'
Lines executed:89.42% of 104
Branches executed:93.55% of 62
Taken at least once:74.19% of 62
Calls executed:94.12% of 17
Creating 'words.c.gcov'
Cannot open source file src/words.c

Lines executed:40.70% of 516
```

Test de l'application au niveau système :

TEHARI Ayoub
L3 informatique
INF1

19 Février 2020

Compte Rendu MOCA – séance 4 (suite)

Objectifs de la séance :

1. utilisation des outils CuTest et gcov.
2. Implémentation des tests / refactoring.
3. Les explications des choix.

Test de l'application au niveau système : (suite)

Exemple 1:

Entrée : hugo1.txt	Output du programme	Résultat attendu
je ne laisserai pas se faner les pervenches, sans aller ecouter ce qu'on dit sous les branches	je (1,1) laisse (1,7) les (1,30) (2,38) ne (1,4) pas (1,17) perven (1,34) qsgon (2,23) sans (2,1) se (1,21) sous (2,33)	aller (2,6) branches (2,42) ce (2,20) dit (2,29) ecouter (2,12) faner (1,24) je (1,1) laisserai (1,7) les (1,30) (2,38) ne (1,4) on (2,26) pas (1,17) pervenches (1,34) qu (2,23) sans (2,1) se (1,21) sous (2,33)

Défaut 1 : les mots sont coupé au 6ème caractère

Modification de la fonction generateMot_t () : en rajoutant le bout de code suivant permettant de faire le bon chaînage et ne pas « ignorer » les éventuelles bouts de mots qui suivent

```
while (newLinkWord->queue_mot->next != NULL)
        { newLinkWord->queue_mot = newLinkWord->queue_mot->next; }
```

Défaut 2 : l’apostrophe

j’ai modifié la MACRO SEP en rajoutant l’apostrophe, pour que le programme ne prend pas le bloc comme un seul mot,

```
#define SEP " ',.-"
```

Défaut 3 : l’ajout des mots au début du dictionnaire

On remarque que l’insertion au début du dictionnaire ne fonctionne pas, puisque la valeur du pointeur reste inchangé après l’appel des fonctions successives : addToDico() ; insertDico() ; et surtout dans la fonction addHeadWord() ;
Du coup, j’ai changé la signature de ces 3 fonction pour devoir retourner l’@ valide du début de dictionnaire, et en suite rajouter les return dictionary ; dans les fichiers sources correspondant, comme suit :

```
//Dans les headers

dico* addHeadWord(dico* dictionary, mot_t* linkWord) ;
dico* insertDico(dico* dictionary, mot_t* linkWord) ;
dico* addToDico(dico* dictionary, char* word, unsigned int* line, unsigned int* colonne) ;
```

```
// dans les fichier sources

return dictionary;
```

TEHARI Ayoub
L3 informatique UGA
19-02-2020

Option Méthodes et outils pour la conception avancée

Compte rendue TP5: Fuzzing et Recherche de Vulnérabilités

Introduction :

L'objectif du tp : utilisation de AFL, et savoir ses limites.
Ce compte rendue comprend les points suivants:

1. un programme contenant une erreur à l'exécution détectée (ou non détectée) par AFL (en expliquant alors pourquoi d'après-vous AFL ne l'a pas détectée …)
2. le résultat de l'exécution d'AFL sur le programme `vulnerable.c`, info : j'ai pas obtenu de crashes puisque j'ai laissé tourné pendant la nuit et que ma version de windows S ne support pas la désactivation de la mise en veille automatique.
3. le résultat de l'exécution de AFL sur le projet

J'ai repris le fichier exemple1.c, puis je l'ai modifié pour qu'il soit indétectable par le Fuzzing :

```
#include <stdio.h>
#include <stdlib.h>
#include <limits.h>
int main (int argc, char *argv[])
{
  int t[100] ;
  FILE *f = fopen(argv[1], "r") ;
  int i;
  int i1 ;
  int *p ;

  p = NULL ;
  if (argc != 2) return 1 ;
  fscanf(f, "%d", &i1) ;
  for (i=0;i<=i1;i++) {
         //t[i]= t[i] -1;
         //if (i>100) *p = 1 ;
         if (i==INT_MAX) *p = 1 ;
  }
  fclose (f) ;
  return 0 ;
}
```

AFL n'as pas détecté ce crash, car il lui faut générer une entrée qui est supérieur ou égale à INT_MAX, ce qui est presque impossible.

Par contre, dans le cas du fichier exemple1.c, AFL détecte, de manière relativement rapide, le crash et voici l'entrée généré : 506.

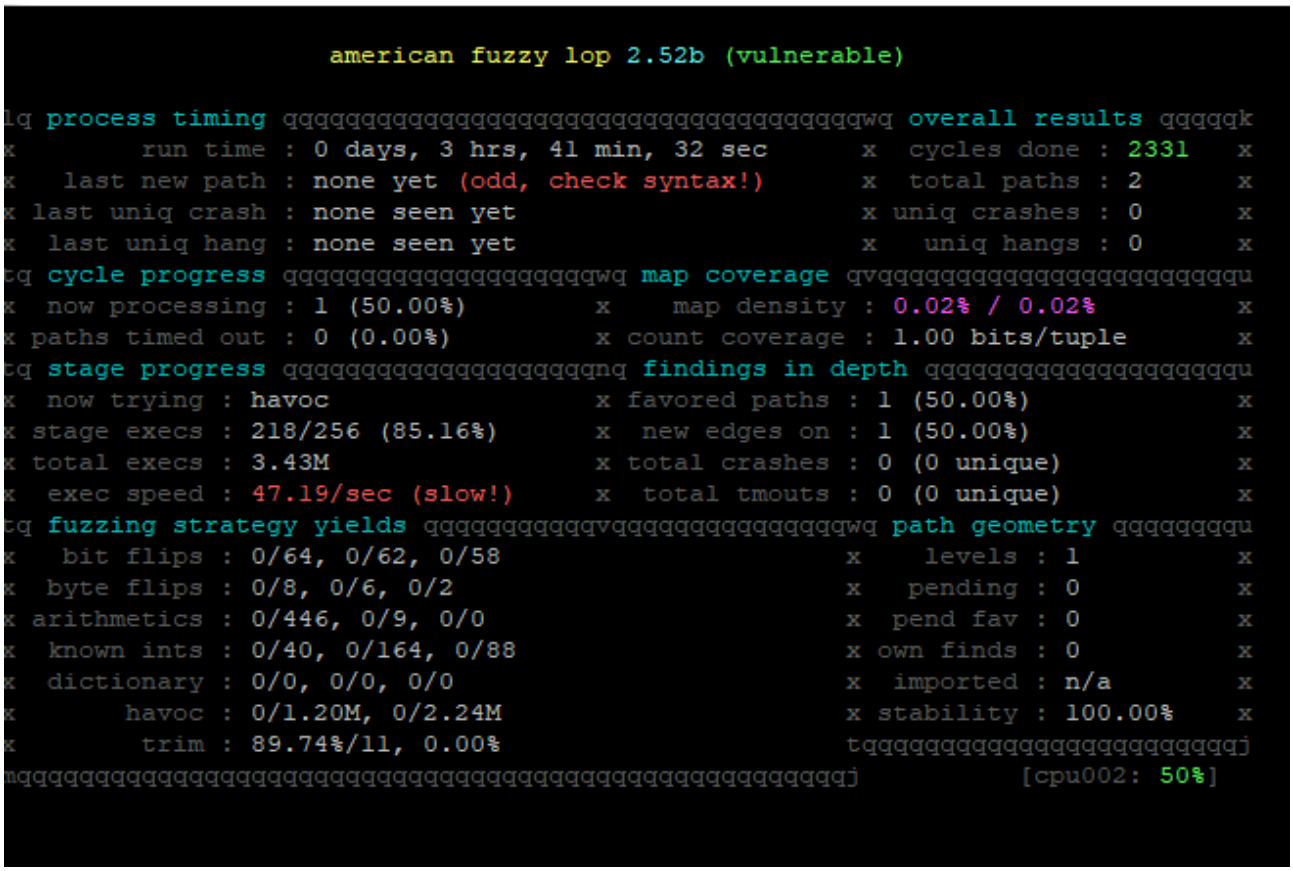
Cette dernière permet au programme d'essayer avoir l'accès à une case mémoire qui n'est pas alloué, ce qui engendre une erreur de type segmentation fault.

Le fichier exemple1.c contient deux possibilités d’accès memoire non autorisé :

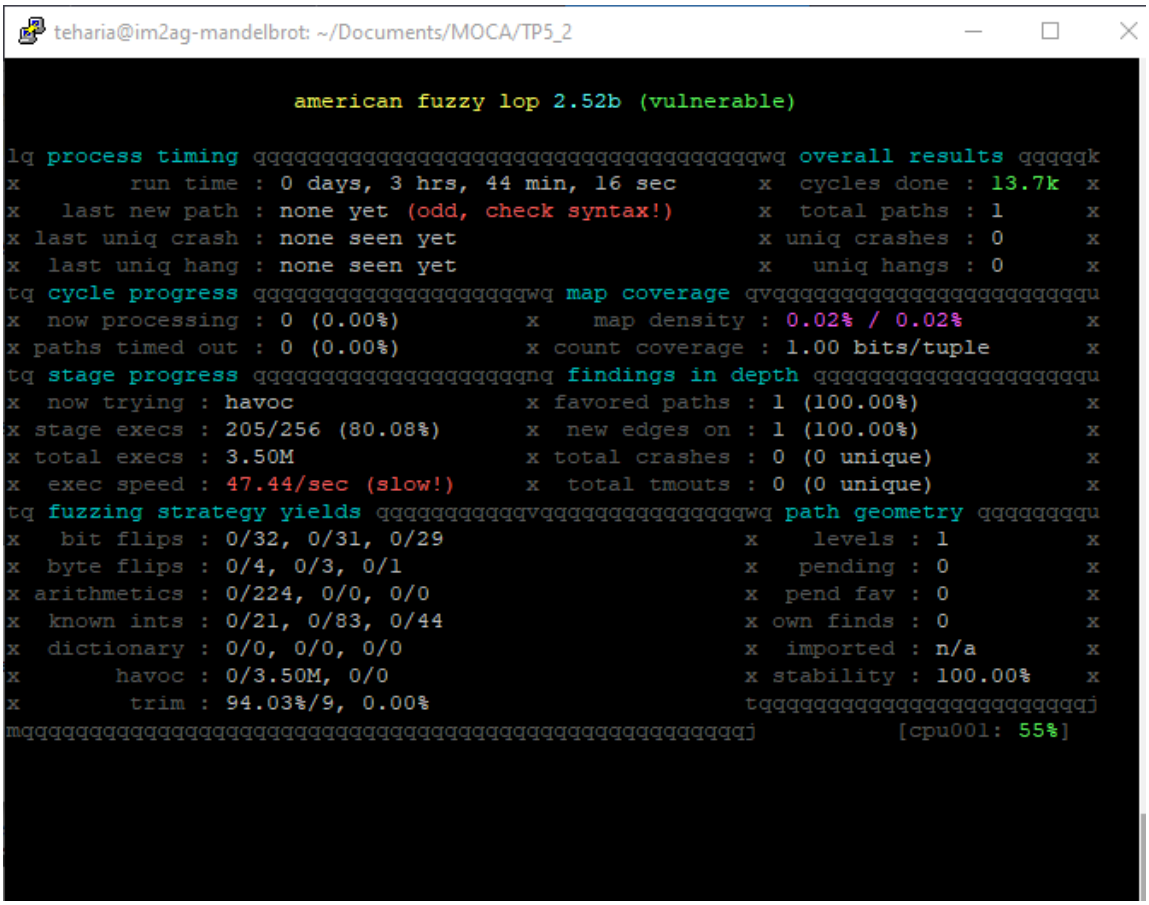
la première :	`for (i=0;i<=i1;i++) {` `  t[i]= t[i] -1;`
La deuxième :	`if (i>100) *p = 1 ;`

Exercice 3 : exécusion de ALF sur vulnerable

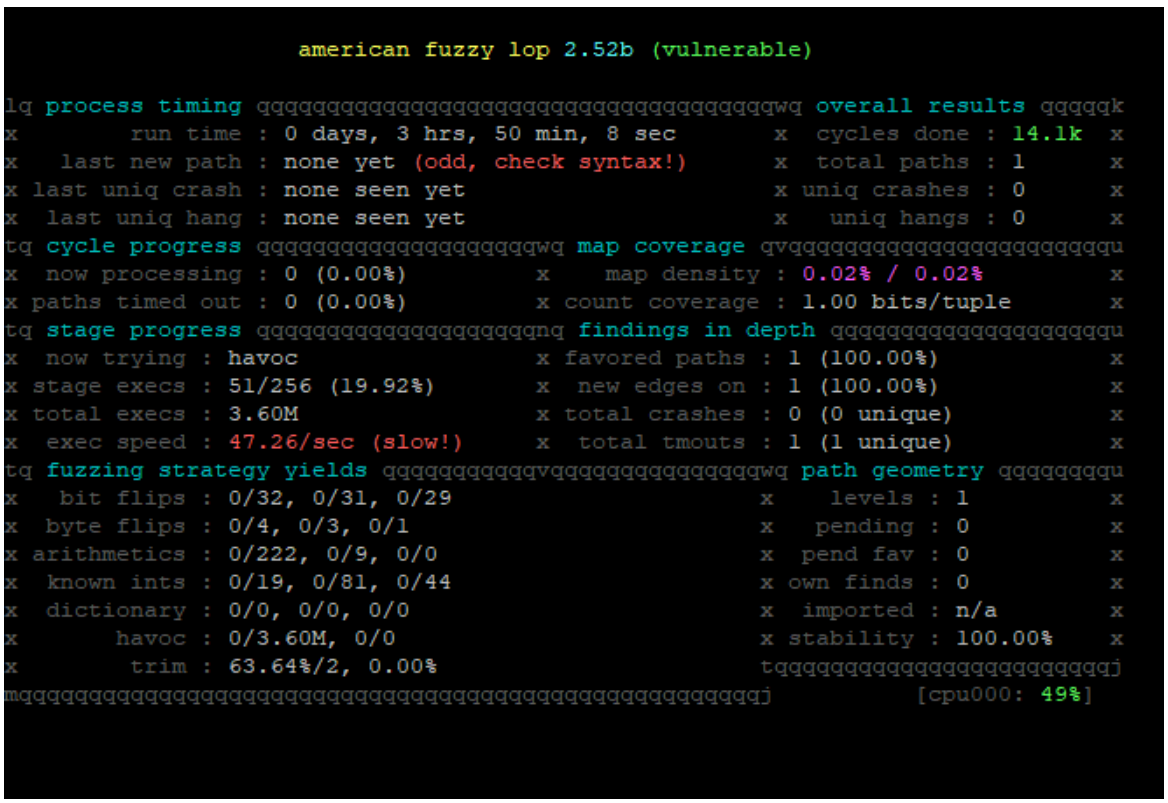
Exécusion sur les deux entrée données :



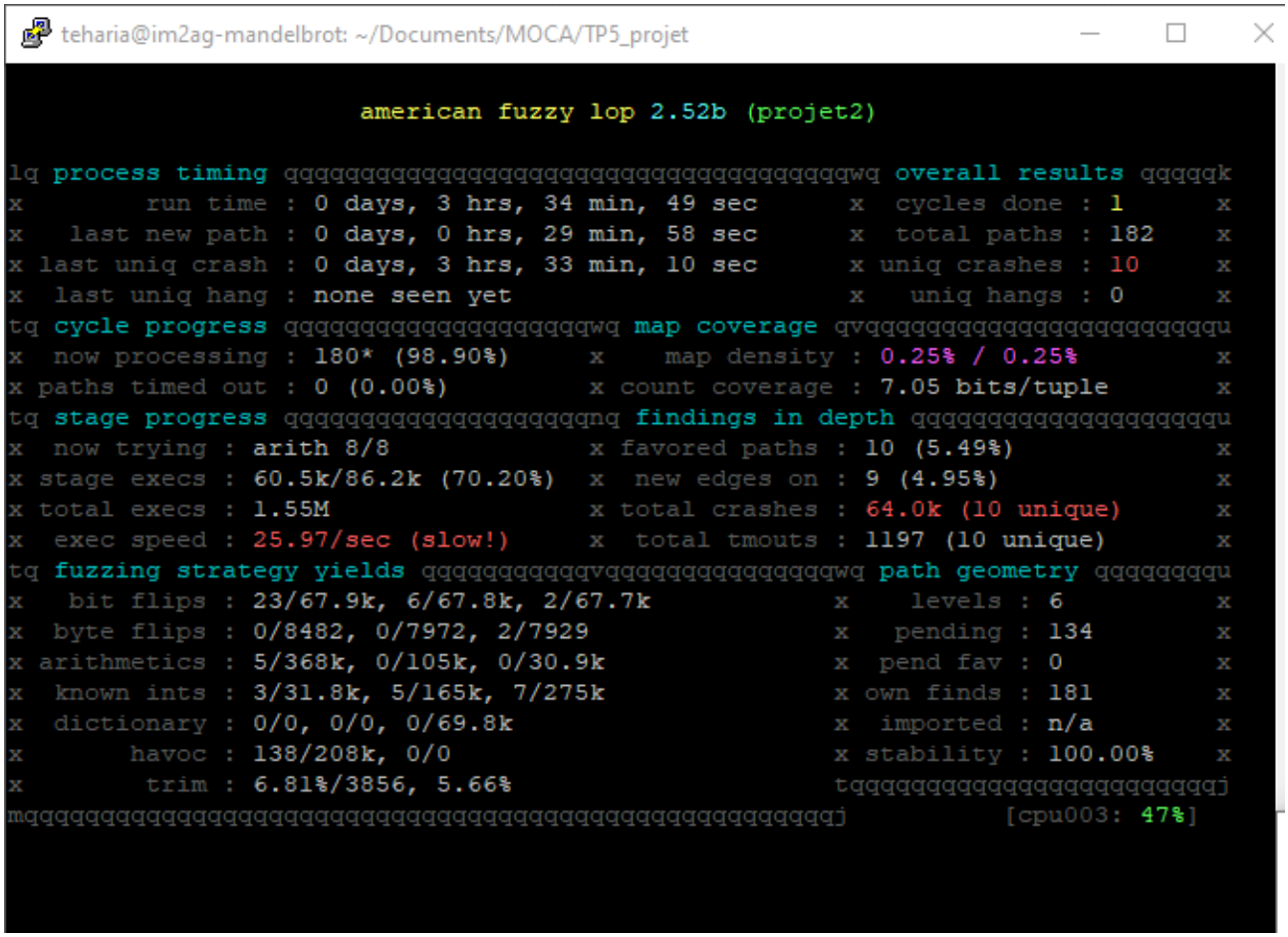
input1.txt :



input2.txt :



Le projet :



Les entrée qui provoquent les crashes :

je ne laisserai pas se faner les pervenches, sans aller ecouter ce qu'on dit sous les branches
je ne laisserai pas se faner les pervenches, sans aller ecouter ce qu'on dit sous les branch\00\FF
\00 je la\80sjjj\90j#jjjjjjjjjjIfj\00\00\00\80la\80sjljjj#la\80sjjjjj#jjjj\8Bjjjjjjjg\00#\FF\80j### se ######!#c\E8#n\00\A3 9\بAqj{\FF¨\FF\8B#Jr:ce\EF\99uo#][c\86eee la\80sjjjjj#jjjj\8Bjjjjjjj\80\FF#\FF\80jjjjjj[#
jV f\00\00\00\00##\00\00\FF\FF\8C\B4\00\A8##\8Cw\8C\8C\8C\B2\8E\8C\95i\8C\8C\8C\A0\8C\BB:#z;&&&##z;##>#####3######\95;\EC\FE\C0\C0\C0\C0\C0\8C\8C\8C\FF\FF\FF\80\8D\8C\87\A1\A1\A1\A1\AD\91\A1\A1\A1
\FBdd\00\006 ¨\FF¨qB\00\F2\F2\F1¨BB\F2\F2\F2\D2\F2\EA~BB;V\FF}\FF\F2\00\00¨BB;¨\FD¨\00\F3\00#\00\00\F2\F2\F2\F2\F2\FF\D2\F2\00¨\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00\00#\00\00\F2\F2\F2\F2\E5\F2\F2¨5B¨\FF\FF\D3a
¿尹꾲

敪粭揀l 颮砀嫿̂爛轏档托拤器ྲྀ
je ne la\8Bsschei* saOs aleeeeec\EFute\00\FF\A3e0n \00it sous les Qrancher\FF
ÿ
j anjrlaiseUlanerlaisai ld ` aÿer ` aÿer lIr equanerlaiseUlai l'on dlfô

On remarque dans toutes ces entrées, il existe des caractères spéciaux en dehors des caractères minuscules autorisés.

```
$ ./objs/projet2 id\:000000\,sig\:11\,src\:000000\,op\:flip1\,pos\:94
Erreur de segmentation (core dumped)
```

TEHARI Ayoub
L3 informatique
INF1

26 Février 2020

Compte Rendu MOCA – séance 6

Objectif de la séance :

1. Valgrind
2. Address Sanitizer
3. Stack protector
4. Klee

Déboguage avec Valgrind

2 Lancement de Valgrind Memcheck

La taille retourné par sizeof() appliqué au type char, est 1.
une chaine de caractère par exemple ‘’toto’’ contient un caractère de fin de chaine ce qui engendre un +1 si on veut vraiment copier cette chaine dans out, lors de l’appel du malloc.

```
$ valgrind ./string_analysis2 "toto"==8459== Memcheck, a memory error detector
==8459== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==8459== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info
==8459== Command: ./string_analysis2 toto
==8459==
==8459== Invalid write of size 1
==8459==    at 0x4C2C2B3: strcpy (vg_replace_strmem.c:458)
==8459==    by 0x4006A4: shazam (string_analysis2.c:8)
==8459==    by 0x4006E1: main (string_analysis2.c:14)
==8459==  Address 0x51e0044 is 0 bytes after a block of size 4 alloc'd
==8459==    at 0x4C28C20: malloc (vg_replace_malloc.c:296)
==8459==    by 0x40068D: shazam (string_analysis2.c:7)
==8459==    by 0x4006E1: main (string_analysis2.c:14)
==8459==
toto
==8459==
==8459== HEAP SUMMARY:
==8459==     in use at exit: 0 bytes in 0 blocks
==8459==   total heap usage: 1 allocs, 1 frees, 4 bytes allocated
==8459==
==8459== All heap blocks were freed -- no leaks are possible
==8459==
==8459== For counts of detected and suppressed errors, rerun with: -v
==8459== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

3 Passage sur votre application :

```
$ valgrind objs/projet2 tests/hugo1.txt
==9285== Memcheck, a memory error detector
==9285== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==9285== Using Valgrind-3.10.0 and LibVEX; rerun with -h for copyright info
==9285== Command: objs/projet2 tests/hugo1.txt
==9285==
==9285== Conditional jump or move depends on uninitialised value(s)
==9285==    at 0x40154C: addToDico (words.c:152)                    ERREUR1
==9285==    by 0x400E20: deco (dico.c:20)
==9285==    by 0x400D25: main (main.c:12)
==9285==
==9285== Conditional jump or move depends on uninitialised value(s)
==9285==    at 0x401404: insertDico (words.c:125)                   ERREUR2
==9285==    by 0x40156D: addToDico (words.c:155)
==9285==    by 0x400E20: deco (dico.c:20)
==9285==    by 0x400D25: main (main.c:12)
==9285==
==9285== Conditional jump or move depends on uninitialised value(s)
==9285==    at 0x4012CC: addTailWord (words.c:85)                   ERREUR3
==9285==    by 0x401418: insertDico (words.c:126)
==9285==    by 0x40156D: addToDico (words.c:155)
==9285==    by 0x400E20: deco (dico.c:20)
==9285==    by 0x400D25: main (main.c:12)
==9285==
Resultat existant dans le fichier resultat, on ecrase
==9285==
==9285== HEAP SUMMARY:
==9285==     in use at exit: 3,574 bytes in 326 blocks
==9285==   total heap usage: 344 allocs, 18 frees, 4,414 bytes allocated
==9285==
==9285== LEAK SUMMARY:
==9285==    definitely lost: 1,822 bytes in 268 blocks
==9285==    indirectly lost: 1,184 bytes in 57 blocks
==9285==      possibly lost: 0 bytes in 0 blocks
==9285==    still reachable: 568 bytes in 1 blocks
==9285==         suppressed: 0 bytes in 0 blocks
==9285== Rerun with --leak-check=full to see details of leaked memory
==9285==
==9285== For counts of detected and suppressed errors, rerun with: -v
==9285== Use --track-origins=yes to see where uninitialised values come from
==9285== ERROR SUMMARY: 3 errors from 3 contexts (suppressed: 2 from 1)
```

Analyse de l' ***ERREUR1 :***

```
149                     ...
150  } else if (dictionary->mot == NULL) {
151    dictionary->mot = newLinkWord;
152  } else {          ...
```

À la ligne 152 on trouve un else juste après un else ifcond2 ; (tel que cond2= dictionary->mot == NULL)

alors que lorsque on remonte dans les appels des fonctions on trouve dans le fichier dico.c (qui contient

Analyse de l' ***ERREUR2 & 3:le dictionare n'est pas initialisé***
On met dans le fichier dico.c :
dico* dictionary = NULL; à la place de
dico* dictionary = (dico*) malloc(sizeof(dico));
ou alors
dictionary→mot = NULL ;
dictionary→next=NULL;

Déboguage avec Address sanitizer et associés

Modification du make :

```
ifdef sanitize
CFLAGS=-fsanitize=$(sanitize) -Wall -g
LDFLAGS =-fsanitize=$(sanitize)
else
CFLAGS=-Wall -g
LDFLAGS =
endif
```

Exécusion sur la version après avoir corriger les erreurs de valgrind :(meme chose avec sanitize=address) SANS ERREURS

```
$ make sanitize= undefined
gcc -fsanitize=undefined -Wall -g -c src/main.c -I include -o objs/main.o
gcc -fsanitize=undefined -Wall -g -c src/dico.c -I include -o objs/dico.o
gcc -fsanitize=undefined -Wall -g -c src/words.c -I include -o objs/words.o
gcc -fsanitize=undefined -Wall -g -c src/maillons.c -I include -o objs/maillons.o
gcc -fsanitize=undefined -Wall -g -c src/display.c -I include -o objs/display.o
gcc -fsanitize=undefined -Wall -g -c src/AllTests.c -I include -o objs/AllTests.o
gcc -fsanitize=undefined -Wall -g -c src/CuTest.c -I include -o objs/CuTest.o
gcc -fsanitize=undefined -Wall -g -c src/SuiteDeTests.c -I include -o
objs/SuiteDeTests.o
gcc -fsanitize=undefined -o objs/projet2 objs/main.o objs/dico.o objs/words.o
objs/maillons.o objs/display.o objs/AllTests.o objs/CuTest.o objs/SuiteDeTests.o
teharia@im2ag-mandelbrot:~/Documents/MOCA/TP6/sanitize$ ./objs/projet2
tests/jeu_test.txt
Resultat existant dans le fichier resultat, on ecrase
teharia@im2ag-mandelbrot:~/Documents/MOCA/TP6/sanitize$ ./objs/projet2 tests/orwell.txt
Resultat existant dans le fichier resultat, on ecrase
teharia@im2ag-mandelbrot:~/Documents/MOCA/TP6/sanitize$ ./objs/projet2 tests/wells.txt
Resultat existant dans le fichier resultat, on ecrase
teharia@im2ag-mandelbrot:~/Documents/MOCA/TP6/sanitize$ ./objs/projet2 tests/hugo1.txt
Resultat existant dans le fichier resultat, on ecrase
teharia@im2ag-mandelbrot:~/Documents/MOCA/TP6/sanitize$ ./objs/projet2 tests/hugo.txt
Resultat existant dans le fichier resultat, on ecrase
```

Le jeu_test.txt est constitué d'entreé généré aléatoirement :

```
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <sys/types.h>
```

```
int main(){
       srandom(getpid());
       int i, j;
       double r, d ;
       for (i=0; i<1000; i++) {
          d = (double) random() * 20.0 / (double) RAND_MAX;
          for (j = 0 ; j< d ; j++) {
              r = (double) random() * 25.0 / (double) RAND_MAX;
              char c = (char) r + 'a';
              printf("%c",c);
          }
          printf("\n");
       }
       return 0;
}
```

Exécusion sur la version sans avoir corriger les erreurs de valgrind :

```
$ ./objs/projet2 tests/jeu_test.txt
ASAN:SIGSEGV
=================================================================
==11841==ERROR: AddressSanitizer: SEGV on unknown address 0x000000000000 (pc
0x000000401e88 sp 0x7ffd5f0c87c0 bp 0x7ffd5f0c8800 T0)
    #0 0x401e87 in compareWord src/words.c:68
    #1 0x402888 in insertDico src/words.c:122
    #2 0x402b0e in addToDico src/words.c:154
    #3 0x40160c in deco src/dico.c:23
    #4 0x4014a5 in main src/main.c:12
    #5 0x7f07e6b46b44 in __libc_start_main (/lib/x86_64-linux-
gnu/libc.so.6+0x21b44)
    #6 0x4013b8
(/home/t/teharia/Documents/MOCA/TP6/sanitize/objs/projet2+0x4013b8)

AddressSanitizer can not provide additional info.
SUMMARY: AddressSanitizer: SEGV src/words.c:68 compareWord
==11841==ABORTING
```

Protection de pile

2 Exemple d'écrasement de pile

l’option -O dans gcc permet de contrôler le niveau d’optimisation,
On a -O0 le niveau par défaut ça permet d’optimiser le temps de compilation et l’utilisation de la memoire, mais par contre la taille du code généré est volumineuse par rapport à d’autre niveau, du coup le temps d’éxécusion aussi, ceci peut améliorer les informations de débogage.
-O2 permet d’optimiser la taille du code et son temps d’execusion, ceci rend le temps de compilation quelque peu plus long, et peut nécessiter plus de mémoire

Déboguage avec Klee

2 Premiers lancement de Klee

```
#include <stdio.h>
int main () {
      printf("Hello, world !\n");
      return 0;
}
```

```
$ clang hello.c -Wall -o hello          Aucun message d'erreur,
$ ./hello                               ni à la compilation ni à l'execusion
Hello, world !
```

3 Application à notre cas d'étude

KLEE: ERROR: /home/t/teharia/Documents/MOCA/TP6/klee/dico.c:208: memory error: out of bound pointer

```
202   int compareWord(mot_t* w1, mot_t* w2) {
203     if (w1 == NULL) {
204       return 1;
205     } else if (w2 == NULL) {
206       return -1;
207     } else {
208       char* word1 = maillonToString(w1->tete_mot);
   ...
```

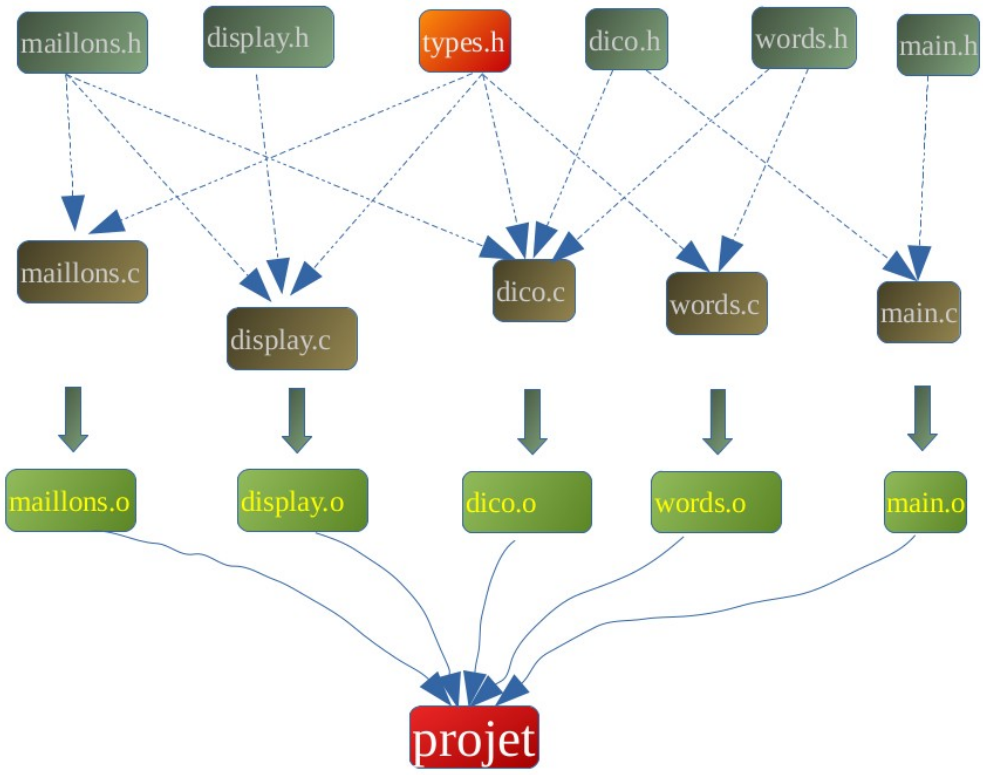
Ce message d'erreur est du que w1→tete_mot est null ;

5 Retour sur le code du Projet

Partie Rappel :

Explication de la modularité choisit :

1. types.h : contien toutes les types utilisés dans l'application.
2. maillons : ce module contient tout ce qui conserne la structure « maillon » et ses alentours : les conversions vers string etc ..
3. words : ce module contient les fonctions : d'ajout des mots aux dictionnaire, lire un nouveau mot , fct pour effectuer les comparaisons entre mots etc ..
4. display : ce module se base sur l'affichage à l'écran.
5. Dico : le program princial qui contient la fonction deco (main précédemment).
6. Main.c : choix entre exécusion des test (CuTest) et l'application deco.



Modification du fichier dico.c :

```
#ifdef KLEE
#include <klee/klee.h>
int deco (int argc, char **argv)
{
	unsigned int* line = (unsigned int*) malloc(sizeof(int));
	klee_make_symbolic(line, sizeof(int), "line");
	unsigned int* colonne = (unsigned int*) malloc(sizeof(int));
	klee_make_symbolic(colonne, sizeof(int), "colonne");
	char* word = (char*) malloc(sizeof(char)*maxSizeWord);
	klee_make_symbolic(word, sizeof(char)*maxSizeWord, "word");
	for (int i =0; i<maxSizeWord; i++){
		klee_assume('a' <= word[i]);
		klee_assume(word[i] <= 'z');
	}
	dico* dictionary = (dico*) malloc(sizeof(dico));
	dictionary->mot=NULL;
	dictionary->next=NULL;
	dictionary = addToDico(dictionary,word,line,colonne);

	displayDico(dictionary);
	return 0;
}

#else
...
```

D’abord, on symbolise les entrées suivants : le mot, sa ligne et sa colonne,

puis on pose des contraintes sur le mot, tel que chaque caractère de la chaine doit être valide.

Ensuite, on fait appel à notre application.

Modification du Makefile :

le repertoire du projet contient :

1. le répertoire include : contient les headers.
2. Le répertoire objs : contient les fichiers objets.
3. Le répertoire src : contient les fichiers source.
4. Le répertoire tests : contient les fichiers de test.
5. Un fichier Makefile pout compiler.

Pour la compilation, j'ai choisit d'utiliser le compilateur wllvm, à la place de gcc,
la commande make klee=1, active ce mode de compilation tel que, toute les fichiers objets sont générés normalement sauf pour deco.o, la compilation se fait en definissant la macro KLEE avec l'option `-DKLEE,` qui active à son tour la symbolisation du mot en entrée,
l'option `-I${KLEE}/include`d permet d'indiquer au compilateur le dossier où se trouve le header klee.h, qui définit la fonction de symbolisation des variable

```
#les declarations des variables
ifdef klee
CC=wllvm
CFLAGS=-Wall -g
LDFLAGS =-L${KLEE}/lib -lkleeRuntest
else
CC=gcc


ifdef gcov
CFLAGS=-Wall -fprofile-arcs -ftest-coverage
LDFLAGS = -fprofile-arcs -ftest-coverage
else
ifdef sanitize
CFLAGS=-fsanitize=$(sanitize) -Wall -g
LDFLAGS =-fsanitize=$(sanitize)
else
CFLAGS=-Wall -g
LDFLAGS =
endif
endif
#test pour produire ou non la bibliotheque dynamique
ifdef N
CFLAGS_MAIN=$(CFLAGS) -fPIC
else
CFLAGS_MAIN=$(CFLAGS)
endif
endif

OBJDIR=objs
INCDIR=include
SRCDIR=src
PROG=$(OBJDIR)/projet2
PROG_S_LIB=projet2_S_LIB
PROG_D_LIB=projet2_D_LIB
PROGS=$(PROG_S_LIB) $(PROG_D_LIB) $(PROG)
MAIN_TEST=prog_tests

#les règles qui génèrent les programmes
all: $(PROG)
all_S_library: $(PROG_S_LIB)
       export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$(PWD)/
all_D_library: $(PROG_D_LIB)
       export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$(PWD)/

all_tests : $(MAIN_TEST)

#règle pour générer les objets
```

```
$(OBJDIR)/%.o : $(SRCDIR)/%.c $(INCDIR)/%.h $(INCDIR)/types.h
	$(CC) $(CFLAGS_MAIN) -c $< -I $(INCDIR) -o $@

ifdef klee
#règle pour générer les objets
$(OBJDIR)/dico.o : $(SRCDIR)/dico.c $(INCDIR)/dico.h $(INCDIR)/types.h $
{KLEE}/include/klee/klee.h
	$(CC) $(CFLAGS_MAIN) -c $< -I${KLEE}/include -I $(INCDIR) -DKLEE -o $@
endif

#fichier qui contient le main
$(OBJDIR)/main.o : $(SRCDIR)/main.c $(INCDIR)/dico.h $(INCDIR)/AllTests.h $
(INCDIR)/CuTest.h
	$(CC) $(CFLAGS) -c $< -I $(INCDIR) -o $@

$(OBJDIR)/AllTests.o : $(SRCDIR)/AllTests.c $(INCDIR)/AllTests.h $
(INCDIR)/SuiteDeTests.h $(INCDIR)/CuTest.h


#règle sans avoir besoin des bibliothèque
$(PROG) : $(OBJDIR)/main.o $(OBJDIR)/dico.o $(OBJDIR)/words.o $(OBJDIR)/maillons.o $
(OBJDIR)/display.o $(OBJDIR)/AllTests.o $(OBJDIR)/CuTest.o $(OBJDIR)/SuiteDeTests.o
	$(CC) $(LDFLAGS) -o $(PROG) $^
ifdef klee
	extract-bc $@
endif

#règle de creation de la bibliothèque statique
libutilitaires.a : $(OBJDIR)/words.o $(OBJDIR)/maillons.o $(OBJDIR)/display.o
	ar r $@ $^
	ranlib $@

#règle de creation de la bibliothèque dynamique
libutilitairesD.so : $(OBJDIR)/words.o $(OBJDIR)/maillons.o $(OBJDIR)/display.o
	$(CC) -shared -o $@ $^

#règle de l'édition des liens au cas d'invocation des bibliothèques
$(PROG_S_LIB) : $(OBJDIR)/dico.o libutilitaires.a
	$(CC) -o $(PROG_S_LIB) $< -Wl,-Bstatic -l utilitaires -L .
#Option -Wl,-Bstatic pour forcer un link statique
#option -Wl,-Bdynamic pour forcer un link dynamique
$(PROG_D_LIB) : $(OBJDIR)/dico.o libutilitairesD.so
	$(CC) -o $(PROG_D_LIB) $< -Wl,-Bdynamic -l utilitairesD -L .


clean :
	@echo " deleting object files, libraries and programs"
	-rm $(OBJDIR)/*.o $(PROGS) *.so *.a $(OBJDIR)/*.gcno $(OBJDIR)/*.gcda
	-rm $(OBJDIR)/$(PROG).bc
```

L’execusion :

```
$ make klee=1wllvm -Wall -g  -c src/main.c -I include -o objs/main.o
wllvm  -c src/dico.c -I/home/m/monniaud/packages/klee/2017-09-26_c7a1f9d//include -I
include -DKLEE -o objs/dico.o
wllvm  -c src/words.c -I include -o objs/words.o
wllvm  -c src/maillons.c -I include -o objs/maillons.o
wllvm  -c src/display.c -I include -o objs/display.o
wllvm  -c src/AllTests.c -I include -o objs/AllTests.o
wllvm  -c src/CuTest.c -I include -o objs/CuTest.o
wllvm  -c src/SuiteDeTests.c -I include -o objs/SuiteDeTests.o
wllvm -L/home/m/monniaud/packages/klee/2017-09-26_c7a1f9d//lib -lkleeRuntest -o
objs/projet2 objs/main.o objs/dico.o objs/words.o objs/maillons.o objs/display.o
objs/AllTests.o objs/CuTest.o objs/SuiteDeTests.o
```

```
extract-bc objs/projet2
$ klee objs/projet2.bc
KLEE: output directory is "/home/t/teharia/Documents/MOCA/TP6/klee_projet/objs/klee-out-
0"
KLEE: Using STP solver backend
KLEE: WARNING: undefined reference to function: _setjmp
KLEE: WARNING: undefined reference to function: fabs
KLEE: WARNING: undefined reference to function: feof
KLEE: WARNING: undefined reference to function: fflush
KLEE: WARNING: undefined reference to function: fgetc
KLEE: WARNING: undefined reference to function: fopen
KLEE: WARNING: undefined reference to function: fprintf
KLEE: WARNING: undefined reference to function: longjmp
KLEE: WARNING: undefined reference to function: printf
KLEE: WARNING: undefined reference to function: sprintf
KLEE: WARNING: undefined reference to variable: stdout
KLEE: WARNING: undefined reference to function: strcat
KLEE: WARNING: undefined reference to function: strchr
KLEE: WARNING: undefined reference to function: strcmp
KLEE: WARNING: undefined reference to function: strcpy
KLEE: WARNING: undefined reference to function: strdup
KLEE: WARNING: undefined reference to function: strlen
KLEE: WARNING: undefined reference to function: ungetc
KLEE: WARNING: undefined reference to function: vsprintf
KLEE: WARNING ONCE: Alignment of memory from call "malloc" is not modelled. Using
alignment of 8.
KLEE: WARNING ONCE: calling external: strlen(46706944) at [no debug info]
KLEE: WARNING ONCE: calling external: fopen(47434240, 45761056) at [no debug info]
KLEE: WARNING ONCE: calling external: feof(44639936) at [no debug info]
KLEE: WARNING ONCE: calling external: printf(46592896) at [no debug info]
Resultat existant dans le fichier resultat, on ecrase
KLEE: WARNING ONCE: calling external: fprintf(44639936, 45761024, 45762064) at [no debug
info]
KLEE: WARNING ONCE: calling external: fflush(140225135096480) at [no debug info]
KLEE: ERROR: (location information missing) external call with symbolic argument:
fprintf
KLEE: NOTE: now ignoring this error at this location

KLEE: done: total instructions = 1135
KLEE: done: completed paths = 1
KLEE: done: generated tests = 1
```

Dictionnaires.txt : contient 971 mots triés

```
$ head dictionnaires.txt
a (937,1)
aaihnehpahi (492,1)
ab (974,1)
abaaqcwwxmfdcuh (709,1)
abyarhoebohpmb (955,1)
aclwkhfsmiwjdnqtr (12,1)
ad (814,1)
afrfxjnpmaqaj (847,1)
agv (734,1)
agviif (556,1)
$ sort dictionnaires.txt > sortedDictionnaire.txt
$ diff dictionnaires.txt sortedDictionnaire.txt
$
```

Ces mots sont généré lors de l'appel de la fonction `klee_make_symbolic()`.