Laurent LEQUIEVRE - Institut Pascal UMR6602 - 2017
laurent.lequievre@univ-bpclermont.fr

Une courte introduction à l'utilisation de Valgrind pour détecter les fuites de mémoire dans le cas d'allocation dynamique de mémoire

Valgrind est un logiciel dit «modulaire», c'est à dire qu'il dispose de plusieurs outils («modules»), dont **«memcheck»** qui va nous intéresser pour détecter :

- Que l'on n'utilise pas de valeurs ou de pointeurs non initialisés.
- Que l'on n'accède pas à des zones mémoire libérées ou non allouées.
- Que l'on ne libère pas deux fois une zone mémoire.
- Que l'on n'oublie pas de libérer la mémoire allouée.

Voici un programme **nommé 'fuite.c'** comportant une fuite de mémoire :

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
{
	int * pt = (int *)malloc(sizeof(int));
	return 0;
}
```

En effet dans ce petit programme, on alloue une mémoire sur le 'tas' en utilisant 'malloc', mais cette mémoire allouée n'est pas libérée par 'free' à la fin du programme !

Pour le compiler : (Ici on obtient un exécutable nommé 'fuite')

gcc -o fuite fuite.c

Pour le lancer avec valgrind : (Ici on lance valgrind avec comme paramètre le nom de l'exécutable)

valgrind ./fuite

Voici le retour de valgrind :

```
==4129== Memcheck, a memory error detector
==4129== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==4129== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright info
==4129== Command: ./fuite
==4129==
==4129==
==4129== HEAP SUMMARY:
==4129==     in use at exit: 4 bytes in 1 blocks
==4129==   total heap usage: 1 allocs, 0 frees, 4 bytes allocated
```

```
==4129==
==4129== LEAK SUMMARY:
==4129==    definitely lost: 4 bytes in 1 blocks
==4129==    indirectly lost: 0 bytes in 0 blocks
==4129==      possibly lost: 0 bytes in 0 blocks
==4129==    still reachable: 0 bytes in 0 blocks
==4129==         suppressed: 0 bytes in 0 blocks
==4129== Rerun with --leak-check=full to see details of leaked memory
==4129==
==4129== For counts of detected and suppressed errors, rerun with: -v
==4129== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

On peut voir que le module lancé par défaut de valgrind est 'memcheck'.

On peut s'apercevoir aussi qu'il y a eu allocation de 4 octets (dans notre cas un 'int') sur le 'tas' qui sont perdus. Valgrind nous suggère pour plus de détails de le lancer avec l'option '--leak-check=full'.

valgrind --leak-check=full ./fuite

Voici le retour de valgrind :

```
==4234== Memcheck, a memory error detector
==4234== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==4234== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright info
==4234== Command: ./fuite
==4234==
==4234==
==4234== HEAP SUMMARY:
==4234==     in use at exit: 4 bytes in 1 blocks
==4234==   total heap usage: 1 allocs, 0 frees, 4 bytes allocated
==4234==
==4234== 4 bytes in 1 blocks are definitely lost in loss record 1 of 1
==4234==    at 0x4C2AB80: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==4234==    by 0x40053E: main (in /datas/Cours_UBP/Cours_Langage_C/mes_solutions/tests/fuite)
==4234==
==4234== LEAK SUMMARY:
==4234==    definitely lost: 4 bytes in 1 blocks
==4234==    indirectly lost: 0 bytes in 0 blocks
==4234==      possibly lost: 0 bytes in 0 blocks
==4234==    still reachable: 0 bytes in 0 blocks
==4234==         suppressed: 0 bytes in 0 blocks
==4234==
==4234== For counts of detected and suppressed errors, rerun with: -v
==4234== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

On obtient plus d'information sur le pourquoi du comment … ici un soucis avec un 'malloc'.
Mais ou est ce 'malloc' dans le fichier ?

Pour obtenir encore plus de précision, on va compiler notre fichier 'fuite.c' avec l'option '-g' :
gcc -o fuite fuite.c -g

Cette option compilera en mode 'debug'.

valgrind --leak-check=full ./fuite

Voici le retour de valgrind :

```
==4323== Memcheck, a memory error detector
==4323== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==4323== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright info
==4323== Command: ./fuite
==4323==
==4323==
==4323== HEAP SUMMARY:
==4323==     in use at exit: 4 bytes in 1 blocks
==4323==   total heap usage: 1 allocs, 0 frees, 4 bytes allocated
==4323==
==4323== 4 bytes in 1 blocks are definitely lost in loss record 1 of 1
==4323==    at 0x4C2AB80: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==4323==    by 0x40053E: main (fuite.c:6)
==4323==
==4323== LEAK SUMMARY:
==4323==    definitely lost: 4 bytes in 1 blocks
==4323==    indirectly lost: 0 bytes in 0 blocks
==4323==      possibly lost: 0 bytes in 0 blocks
==4323==    still reachable: 0 bytes in 0 blocks
==4323==         suppressed: 0 bytes in 0 blocks
==4323==
==4323== For counts of detected and suppressed errors, rerun with: -v
==4323== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
```

Ouf !! ce coup-ci on nous indique le numéro de la ligne (ligne n° 6) du fichier 'fuite.c' ou le 'malloc' a été effectué.

Maintenant corrigeons notre fuite de mémoire :

```
#include <stdio.h>
#include <stdlib.h>

int main(void)
```

```
{
	int * pt = (int *)malloc(sizeof(int));
	free(pt) ;
	return 0;
}
```

gcc -o fuite fuite.c -g
valgrind --leak-check=full ./fuite

Voici le retour de valgrind :

```
==4427== Memcheck, a memory error detector
==4427== Copyright (C) 2002-2013, and GNU GPL'd, by Julian Seward et al.
==4427== Using Valgrind-3.10.1 and LibVEX; rerun with -h for copyright info
==4427== Command: ./fuite
==4427==
==4427==
==4427== HEAP SUMMARY:
==4427==     in use at exit: 0 bytes in 0 blocks
==4427==   total heap usage: 1 allocs, 1 frees, 4 bytes allocated
==4427==
==4427== All heap blocks were freed -- no leaks are possible
==4427==
==4427== For counts of detected and suppressed errors, rerun with: -v
==4427== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Plus de fuite !! 1 allocation de 4 octets et 1 libération, tout est OK.

Pour plus de détails :
https://openclassrooms.com/courses/debuguer-facilement-avec-valgrind