Tutorial Valgrind

Valgrind es un conjunto de herramientas de código abierto que permite la depuración y optimización de programas. Dentro de Valgrind la herramienta más utilizada es *Memcheck.* Dicha herramienta ayuda a la identificación de los errores más comunes en la gestión de memoria dinámica como son: lagunas de memoria, errores en la liberación de punteros, acceso a memoria del heap no reservada previamente o no inicializada. Un programa no se puede dar por válido hasta que su ejecución bajo un amplio rango de circunstancias no produce ningún error como los mencionados. El siguiente programa en C presenta todos los tipos de errores mencionados previamente:

```
01  #include <stdlib.h>
02  #include <stdio.h>
03
04  int* f(int numero_celdas)
05  {
06    int* x;
07    int* y;
08
09    y = malloc(numero_celdas * sizeof(int));
10    x = malloc(numero_celdas * sizeof(int));
11    x[numero_celdas] = 0; // Error 1: acceso fuera de memoria reservada
12
13    return x;
14  } // Error 2: No se libera la memoria de y: Laguna de memoria
15
16  int main(void)
17  {
18    int* y = f(10);
19
20    printf("Valor celda 1: %d\n", y[0]); // Error 3: valor no inicializado
21
22    free(y);
23    free(y); //Error 4: No se puede liberar dos veces el mismo bloque
24
25    return 0;
26  }
```

Para analizar este programa bajo la herramienta Valgrind hay que seguir los siguientes pasos:

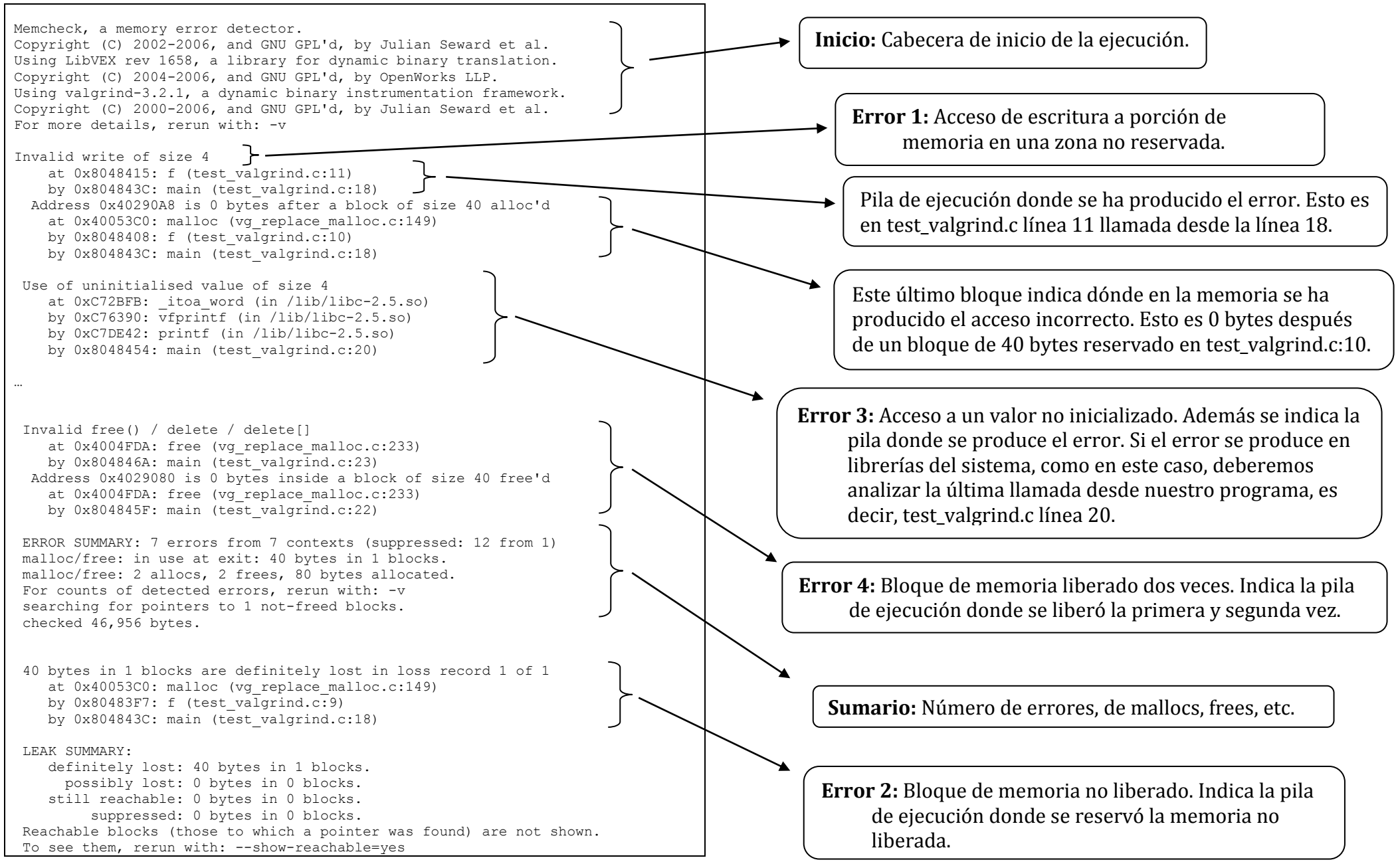
1. Compilar con la opción `-g` y no usar opción de optimización de código (`-O1`, `-O2`, `-O3`, etc.). Si guardamos el código en un fichero con nombre test_valgrind.c, podemos compilarlo como:

   ```
   gcc -g test_valgrind.c -o test_valgrind
   ```

2. Ejecutar el programa dentro de Valgrind como

   ```
   valgrind --leak-check=full test_valgrind
   ```

3. Analizar la salida. La salida de Valgrind se estructura de la siguiente forma. Primero se imprime una cabecera, luego para cada error encontrado Valgrind muestra un bloque descriptivo. El bloque indica el tipo de error producido más la pila de ejecución donde se ha producido. Finalmente se indica un resumen de la ejecución con el número de errores y de reservas realizadas así como la cantidad de memoria que queda sin liberar. En la siguiente página se puede ver el resultado de Valgrind aplicado al programa anterior:


Memcheck, a memory error detector.
Copyright (C) 2002-2006, and GNU GPL'd, by Julian Seward et al.
Using LibVEX rev 1658, a library for dynamic binary translation.
Copyright (C) 2004-2006, and GNU GPL'd, by OpenWorks LLP.
Using valgrind-3.2.1, a dynamic binary instrumentation framework.
Copyright (C) 2000-2006, and GNU GPL'd, by Julian Seward et al.
For more details, rerun with: -v
Invalid write of size 4
at 0x8048415: f (test_valgrind.c:11)
by 0x804843C: main (test_valgrind.c:18)
Address 0x40290A8 is 0 bytes after a block of size 40 alloc'd
at 0x40053C0: malloc (vg_replace_malloc.c:149)
by 0x8048408: f (test_valgrind.c:10)
by 0x804843C: main (test_valgrind.c:18)
Use of uninitialised value of size 4
at 0xC72BFB: _itoa_word (in /lib/libc-2.5.so)
by 0xC76390: vfprintf (in /lib/libc-2.5.so)
by 0xC7DE42: printf (in /lib/libc-2.5.so)
by 0x8048454: main (test_valgrind.c:20)
…
Invalid free() / delete / delete[]
at 0x4004FDA: free (vg_replace_malloc.c:233)
by 0x804846A: main (test_valgrind.c:23)
Address 0x4029080 is 0 bytes inside a block of size 40 free'd
at 0x4004FDA: free (vg_replace_malloc.c:233)
by 0x804845F: main (test_valgrind.c:22)
ERROR SUMMARY: 7 errors from 7 contexts (suppressed: 12 from 1)
malloc/free: in use at exit: 40 bytes in 1 blocks.
malloc/free: 2 allocs, 2 frees, 80 bytes allocated.
For counts of detected errors, rerun with: -v
searching for pointers to 1 not-freed blocks.
checked 46,956 bytes.
40 bytes in 1 blocks are definitely lost in loss record 1 of 1
at 0x40053C0: malloc (vg_replace_malloc.c:149)
by 0x80483F7: f (test_valgrind.c:9)
by 0x804843C: main (test_valgrind.c:18)
LEAK SUMMARY:
definitely lost: 40 bytes in 1 blocks.
possibly lost: 0 bytes in 0 blocks.
still reachable: 0 bytes in 0 blocks.
suppressed: 0 bytes in 0 blocks.
Reachable blocks (those to which a pointer was found) are not shown.
To see them, rerun with: --show-reachable=yes
Inicio: Cabecera de inicio de la ejecución.
Error 1: Acceso de escritura a porción de memoria en una zona no reservada.
Pila de ejecución donde se ha producido el error. Esto es en test_valgrind.c línea 11 llamada desde la línea 18.
Este último bloque indica dónde en la memoria se ha producido el acceso incorrecto. Esto es 0 bytes después de un bloque de 40 bytes reservado en test_valgrind.c:10.
Error 3: Acceso a un valor no inicializado. Además se indica la pila donde se produce el error. Si el error se produce en librerías del sistema, como en este caso, deberemos analizar la última llamada desde nuestro programa, es decir, test_valgrind.c línea 20.
Error 4: Bloque de memoria liberado dos veces. Indica la pila de ejecución donde se liberó la primera y segunda vez.
Sumario: Número de errores, de mallocs, frees, etc.
Error 2: Bloque de memoria no liberado. Indica la pila de ejecución donde se reservó la memoria no liberada.