

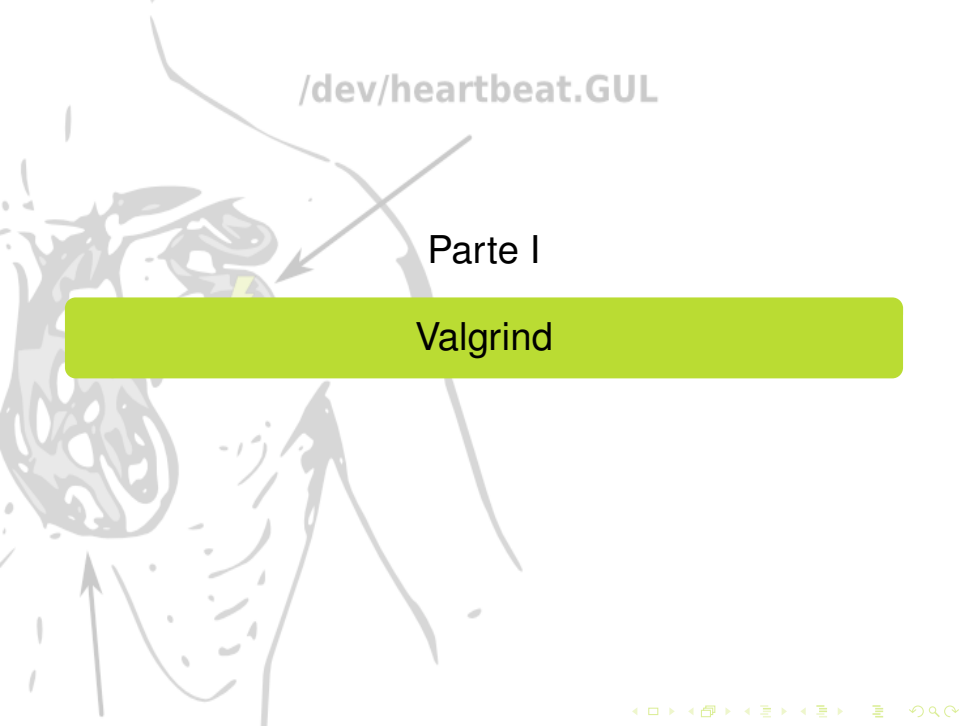
/dev/heartbeat.GUL

Aprendiendo a depurar código

Borja Bergua Guerra

Grupo de Usuarios de Linux de la UC3M

13 de Marzo de 2008



`/dev/heartbeat.GUL`

Parte I

Valgrind

Tipos de error

- Lecturas/escrituras ilegales
- Uso de valores sin inicializar
- Liberaciones de memoria ilegales
- Cuando se libera un bloque con una llamada incorrecta
- Pasar parámetros a llamadas al sistema con permisos de lectura/escritura incorrectos
- Solapamiento de bloques origen y destino
- Detección de pérdidas de memoria

Valgrind - Tipos de error

/dev/heartbeat.GUL

Lecturas/escrituras ilegales

```
Invalid read of size 4
  at 0x40F6BBCC: (within /usr/lib/libpng.so.2.1.0.9)
  by 0x40F6B804: (within /usr/lib/libpng.so.2.1.0.9)
  by 0x40B07FF4: read_png_image(QImageIO *) (kernel/qpngio.cpp:326)
  by 0x40AC751B: QImageIO::read() (kernel/qimage.cpp:3621)
Address 0xBFFFF0E0 is not stack'd, malloc'd or free'd
```

Valgrind - Tipos de error

/dev/heartbeat.GUL

Uso de valores sin inicializar

```
Conditional jump or move depends on uninitialised value(s)  
  at 0x402DFA94: _IO_vfprintf (_itoa.h:49)  
  by 0x402E8476: _IO_printf (printf.c:36)  
  by 0x8048472: main (tests/manuell.c:8)
```

Liberaciones de memoria ilegales

```
Invalid free()
  at 0x4004FFDF: free (vg_clientmalloc.c:577)
  by 0x80484C7: main (tests/doublefree.c:10)
Address 0x3807F7B4 is 0 bytes inside a block of size 177 free'd
  at 0x4004FFDF: free (vg_clientmalloc.c:577)
  by 0x80484C7: main (tests/doublefree.c:10)
```

Valgrind - Tipos de error

/dev/heartbeat.GUL

Cuando se libera un bloque con una llamada incorrecta

```
Mismatched free() / delete / delete []
  at 0x40043249: free (vg_clientfuncs.c:171)
  by 0x4102BB4E: QGArray::~~QGArray(void) (tools/qgarray.cpp:149)
  by 0x4C261C41: PptDoc::~~PptDoc(void) (include/qmemarray.h:60)
  by 0x4C261F0E: PptXml::~~PptXml(void) (pptxml.cc:44)
Address 0x4BB292A8 is 0 bytes inside a block of size 64 alloc'd
  at 0x4004318C: operator new[](unsigned int) (vg_clientfuncs.c:152)
  by 0x4C21BC15: KLaola::readSBStream(int) const (klaola.cc:314)
  by 0x4C21C155: KLaola::stream(KLaola::OLENode const *) (klaola.cc:4
  by 0x4C21788F: OLEFilter::convert(QCString const &) (olefilter.cc:2
```

Valgrind - Tipos de error

/dev/heartbeat.GUL

Pasar parámetros a llamadas al sistema con permisos de lectura/escritura incorrectos

```
Syscall param write(buf) points to uninitialised byte(s)
  at 0x25A48723: __write_nocancel (in /lib/tls/libc-2.3.3.so)
  by 0x259AFAD3: __libc_start_main (in /lib/tls/libc-2.3.3.so)
  by 0x8048348: (within /auto/homes/njn25/grind/head4/a.out)
Address 0x25AB8028 is 0 bytes inside a block of size 10 alloc'd
  at 0x259852B0: malloc (vg_replace_malloc.c:130)
  by 0x80483F1: main (a.c:5)
```

```
Syscall param exit(error_code) contains uninitialised byte(s)
  at 0x25A21B44: __GI__exit (in /lib/tls/libc-2.3.3.so)
  by 0x8048426: main (a.c:8)
```

Valgrind - Tipos de error

/dev/heartbeat.GUL

Solapamiento de bloques origen y destino

```
Source and destination overlap in memcpy(0xbffff294, 0xbffff280, 21)
at 0x40026CDC: memcpy (mc_replace_strmem.c:71)
by 0x804865A: main (overlap.c:40)
```

Detección de pérdidas de memoria

```
8 bytes in 1 blocks are definitely lost in loss record 1 of 14
  at 0x.....: malloc (vg_replace_malloc.c:...)
  by 0x.....: mk (leak-tree.c:11)
  by 0x.....: main (leak-tree.c:39)

88 (8 direct, 80 indirect) bytes in 1 blocks are definitely lost
    in loss record 13 of 14
  at 0x.....: malloc (vg_replace_malloc.c:...)
  by 0x.....: mk (leak-tree.c:11)
  by 0x.....: main (leak-tree.c:25)
```

Valgrind - Uso

/dev/heartbeat.GUL

Cómo se usa

- Uso normal

```
./myprog arg1 arg2
```

- Uso con valgrind

```
valgrind ./myprog arg1 arg2
```

- Con detección de pérdidas de memoria

```
--leak-check=full
```

- Con más detección de (posible) memoria pérdida/malgastada

```
--show-reachable=yes
```

Valgrind - Ejemplo

/dev/heartbeat.GUL

Código: ejemplo-valgrind.c

```
1: #include <stdlib.h>
2:
3: void f(void)
4: {
5:     int* x = malloc(10 * sizeof(int));
6:     x[10] = 0; // problem 1: heap block overrun
7: }              // problem 2: memory leak -- x not freed
8:
9: int main(void)
10: {
11:     f();
12:     return 0;
13: }
```

Valgrind - Ejemplo

/dev/heartbeat.GUL

Salida: valgrind ./ejemplo-valgrind

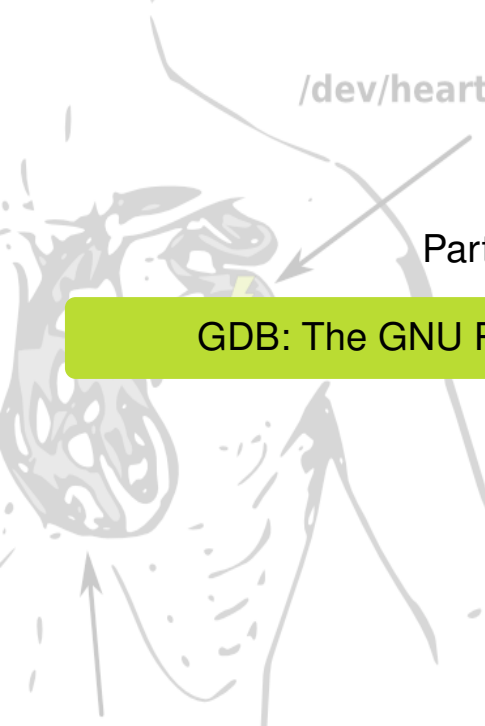
```
Invalid write of size 4
  at 0x804838F: f (ejemplo-valgrind.c:6)
  by 0x80483AC: main (ejemplo-valgrind.c:11)
Address 0x4183050 is 0 bytes after a block of size 40 alloc'd
  at 0x4022765: malloc (vg_replace_malloc.c:149)
  by 0x8048385: f (ejemplo-valgrind.c:5)
  by 0x80483AC: main (ejemplo-valgrind.c:11)
```

Valgrind - Ejemplo

/dev/heartbeat.GUL

Salida: `valgrind --leak-check=full ./ejemplo-valgrind`

```
40 bytes in 1 blocks are definitely lost in loss record 1 of 1
  at 0x4022765: malloc (vg_replace_malloc.c:149)
  by 0x8048385: f (ejemplo-valgrind.c:5)
  by 0x80483AC: main (ejemplo-valgrind.c:11)
```



`/dev/heartbeat.GUL`

Parte II

GDB: The GNU Project Debugger

Usando GDB

- Compilando el programa

```
gcc -g
```

- Iniciando GDB

```
gdb program
```

- Ejecutando el programa

```
(gdb) run [args]
```

- Terminando GDB

```
(gdb) quit
```

GDB - Comandos básicos

Ejecutar el programa

- Argumentos

```
(gdb) set args
```

```
(gdb) show args
```

- Entorno

```
(gdb) set environment varname [=value]
```

```
(gdb) show environment [varname]
```

- Directorio de trabajo

```
(gdb) cd directory
```

```
(gdb) pwd
```

- Entada y salida estándar

```
(gdb) run > outfile
```

Parando y continuando

- Parando el programa
 - Breakpoints
 - Watchpoints
 - Catchpoints
- Continuando el programa
 - `continue`
 - `step`
 - `next`

Breakpoints

- Un breakpoint para el programa cuando se alcanza un punto concreto.

```
(gdb) break [location]
```

```
(gdb) info breakpoints
```

- location puede ser:
 - `linenum`
 - `filename:linenum`
 - `function`
 - `filename:function`
 - `etc.`

Watchpoints

- Un `watchpoint` para el programa cuando el valor de una expresión cambia, sin necesidad de predecir el lugar concreto en donde puede suceder.

```
(gdb) watch expr
```

```
(gdb) info watchpoints
```

Catchpoints

- Un `catchpoint` para el programa cuando se produzcan ciertos tipos de eventos, como excepciones C++ o carga de bibliotecas dinámicas.

```
(gdb) catch event
```

```
(gdb) info breakpoints
```

```
(gdb) info watchpoints
```

- `event` puede ser:

- `throw`
- `catch`
- `exec`
- `fork`
- `etc.`

GDB - Comandos básicos

/dev/heartbeat.GUL

Examinando el código fuente

- `list` muestra el código fuente alrededor de la línea en la que nos encontramos.

```
(gdb) list
```

```
(gdb) list -
```

```
(gdb) list linenum
```

```
(gdb) list function
```

GDB - Comandos básicos

/dev/heartbeat.GUL

Examinando los datos

- `print` muestra el valor de una variable o expresión.
`(gdb) print expr`
- `display` muestra el valor de una variable o expresión cada vez que el programa se para.
`(gdb) display expr`

Examinando la pila de llamadas

- `backtrace` muestra la pila de llamadas, un resumen de cómo el programa llegó ahí.

```
(gdb) backtrace
```

```
(gdb) bt
```

GDB - Comandos básicos

Core dumps

- En primer lugar debemos conocer el tamaño máximo permitido para ficheros core.

```
ulimit -c
```

- Si es 0, no generará ficheros core. Debemos ampliarlo según nuestras necesidades.

```
ulimit -c unlimited
```

- Para examinar un fichero core debemos abrir el programa junto con el fichero core generado.

```
gdb program core
```

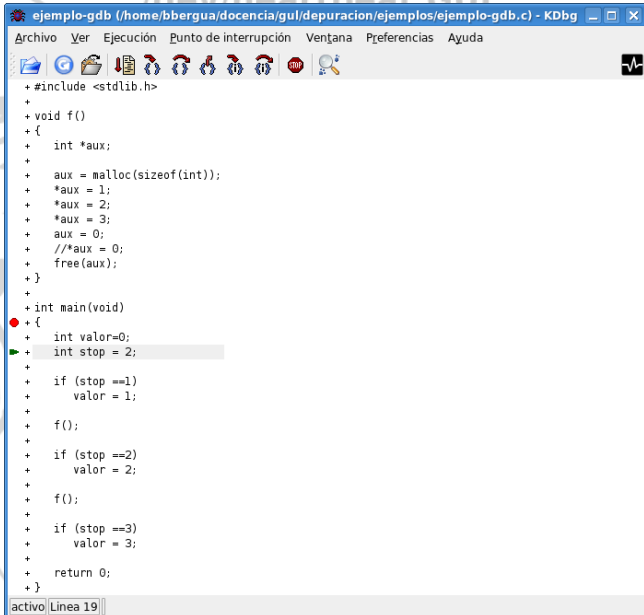
- Finalmente mostramos la pila de llamadas.

```
(gdb) backtrace
```

Continuando el programa

- `continue` continúa ejecutando el programa hasta que termine.
- `step` ejecuta la siguiente línea de código fuente (entra dentro de funciones).
- `next` ejecuta la siguiente línea de código de una vez (no entra dentro de funciones).
- `finish` continúa ejecutando el programa hasta que termine la función actual.

Interfaz gráfica - Kdbg



```
+ #include <stdlib.h>
+
+ void f()
+ {
+     int *aux;
+
+     aux = malloc(sizeof(int));
+     *aux = 1;
+     *aux = 2;
+     *aux = 3;
+     aux = 0;
+     /*aux = 0;
+     free(aux);
+ }
+
+ int main(void)
+ {
+     int valor=0;
+     int stop = 2;
+
+     if (stop ==1)
+         valor = 1;
+
+     f();
+
+     if (stop ==2)
+         valor = 2;
+
+     f();
+
+     if (stop ==3)
+         valor = 3;
+
+     return 0;
+ }
```

activo Linea 19

Enlaces

/dev/heartbeat.GUL

Valgrind

- <http://www.valgrind.org>

GDB

- <http://sourceware.org/gdb/documentation/>
- <http://bulma.net/body.phtml?nIdNoticia=1805>