Valgrind

Jakub Kúdela

Valgrind 3.8.1

- an instrumentation framework for dynamic analysis tools
- currently includes production-quality tools:

`memcheck`	memory-manag. Problems
`cachegrind`	cache profiler
`callgrind`	`cachegrind` extension, provides callgraphs
`massif`	heap profiler
`helgrind, DRD`	thread debuggers
`lackey, nulgrind`	demonstrative purposes

- others tools: http://valgrind.org/downloads/variants.html
- can be connected together with GDB (manual)

http://valgrind.org/
http://valgrind.org/info/tools.html
http://valgrind.org/downloads/variants.html

Preparing your program

- highly recommended compiler options:

`-g`	includes debugging information
`-Wall`	shows compiler's warning messages
`-O0`	code gets not optimized

- possible compiler options:

`-O1`	line numbers can be inaccurate
`-fno-inline`	suppresses all inlining

- not recommended compiler options:

`-O2`	`memcheck` false uninitialised value errors

http://valgrind.org/docs/manual/quick-start.html
http://valgrind.org/docs/manual/manual-core.html#manual-core.started

Running your program under Valgrind

- Works directly with existing executables:

```
valgrind [valgrind-options] <prog> [prog-options]
```

- Option that dictates which Valgrind tool will run:

```
--tool=<tool-to-run> [default: memcheck]
```

- Invocation example:

```
valgrind --tool=memcheck ls -l
```

http://valgrind.org/docs/manual/manual-core.html#manual-core.whatdoes

Memcheck

- memory error detector focused on problems such as:
  - Accessing memory you shouldn't
    - overrunning/underrunning heap blocks
    - overrunning the top of the stack
    - accessing memory after it has been freed
  - Using undefined values
    - that have not been initialised
    - that have been derived from other undefined values
  - Incorrect freeing of heap memory
    - double-freeing heap blocks
    - mismatched use of `malloc/new/new[]` vs. `free/delete/delete[]`
  - Overlapping source and destination blocks in `memcpy` and related functions.
  - Memory leaks

http://valgrind.org/docs/manual/mc-manual.html#mc-manual.overview

Interpreting Memcheck's output

- source:

```c
#include <stdlib.h>

void f(void)
{
    int * x = malloc(10 * sizeof(int));
    x[10] = 0;  // problem 1: heap block overrun
}               // problem 2: memory leak

int main(void)
{
    f();
    return 0;
}
```

- analysis:
  - PID
  - error types
  - stack traces
  - code adresses
  - memory adresses

```
==6351== Invalid write of size 4
==6351==    at 0x400552: f (interpreting_output.c:6)
==6351==    by 0x400562: main (interpreting_output.c:11)
==6351==  Address 0x51df068 is 0 bytes after a block of size 40 alloc'd
==6351==    at 0x4C2C66D: malloc (in /usr/lib64/valgrind/vgpreload_memcheck-amd6
4-linux.so)
==6351==    by 0x400545: f (interpreting_output.c:5)
==6351==    by 0x400562: main (interpreting_output.c:11)
==6351==
==6351==
==6351== HEAP SUMMARY:
==6351==     in use at exit: 40 bytes in 1 blocks
==6351==   total heap usage: 1 allocs, 0 frees, 40 bytes allocated
==6351==
==6351== LEAK SUMMARY:
==6351==    definitely lost: 40 bytes in 1 blocks
```

http://valgrind.org/docs/manual/mc-manual.html#mc-manual.errormsgs

Memcheck & illegal read/write

- source:

```c
#include <stdlib.h>

int main()
{
    int * a = malloc(10 * sizeof(int));
    int b = a[20];       // problem 1: illegal read
    a[30] = b;           // problem 2: illegal write
    return 0;
}
```

- analysis:
  - problem 1:
  - problem 2:

```
==1057== Invalid read of size 4
==1057==    at 0x400552: main (illegal_read_write.c:6)
==1057==  Address 0x51df090 is not stack'd, malloc'd or (recently) free'd
==1057==
==1057== Invalid write of size 4
==1057==    at 0x400562: main (illegal_read_write.c:7)
==1057==  Address 0x51df0b8 is not stack'd, malloc'd or (recently) free'd
```

http://valgrind.org/docs/manual/mc-manual.html#mc-manual.badrw

Memcheck & use of uninitialised values

- source:

```c
#include <stdio.h>

int main()
{
    int a;
    printf("%d", a);    // problem: use of uninitialised value
    return 0;
}
```

- analysis:
  - problem:

```
==886== Use of uninitialised value of size 8
==886==    at 0x4E794DB: _itoa_word (in /lib64/libc-2.15.so)
==886==    by 0x4E7BD38: vfprintf (in /lib64/libc-2.15.so)
==886==    by 0x4E84709: printf (in /lib64/libc-2.15.so)
==886==    by 0x400552: main (use_uninit.c:6)
```

http://valgrind.org/docs/manual/mc-manual.html#mc-manual.uninitvals

Memcheck & use of uninitialised or unaddressable values in system calls

- source:

```c
#include <stdlib.h>

int main( void )
{
    int * a = malloc(10 * sizeof(int));
    exit(a[0]);     // problem 1: use of uninitialised value in syscall
                    // problem 2: memory leak
}
```

- analysis:
  - problem:

```
==1097== Syscall param exit_group(status) contains uninitialised byte(s)
==1097==    at 0x4EED8A8: _Exit (in /lib64/libc-2.15.so)
==1097==    by 0x4E6C5FC: __run_exit_handlers (in /lib64/libc-2.15.so)
==1097==    by 0x4E6C6A4: exit (in /lib64/libc-2.15.so)
==1097==    by 0x4005A6: main (syscall_uninit.c:6)
```

http://valgrind.org/docs/manual/mc-manual.html#mc-manual.bad-syscall-args

Memcheck & illegal frees

- source:

```c
#include <stdlib.h>

int main()
{
    int * a = malloc(10 * sizeof(int));
    free(a);
    free(a);    // problem: repeated deallocation
    return 0;
}
```

- analysis:
  - problem:

```
==1141== Invalid free() / delete / delete[] / realloc()
==1141==    at 0x4C2B35C: free (in /usr/lib64/valgrind/vgpreload_memcheck-amd64-
linux.so)
==1141==    by 0x4005B1: main (illegal_free.c:7)
==1141==  Address 0x51df040 is 0 bytes inside a block of size 40 free'd
==1141==    at 0x4C2B35C: free (in /usr/lib64/valgrind/vgpreload_memcheck-amd64-
linux.so)
==1141==    by 0x4005A5: main (illegal_free.c:6)
```

http://valgrind.org/docs/manual/mc-manual.html#mc-manual.badfrees

Memcheck & freeing heap block with an inappropriate deallocation function

- source:

```
1  int main()
2  {
3      int * a = new int;
4      delete[] a;      // problem: inappropriate deallocation
5      return 0;
6  }
```

- analysis:
  - problem:

```
==1212== Mismatched free() / delete / delete []
==1212==    at 0x4C2A90E: operator delete[](void*) (in /usr/lib64/valgrind/vgpre
load_memcheck-amd64-linux.so)
==1212==    by 0x40062C: main (inappropriate_deallocation.c:4)
==1212==  Address 0x59f0040 is 0 bytes inside a block of size 4 alloc'd
==1212==    at 0x4C2C099: operator new(unsigned long) (in /usr/lib64/valgrind/vg
preload_memcheck-amd64-linux.so)
==1212==    by 0x400615: main (inappropriate_deallocation.c:3)
```

http://valgrind.org/docs/manual/mc-manual.html#mc-manual.rudefn

Memcheck & overlapping source and destination blocks

- source:

```c
#include <string.h>

int main()
{
    int a [10];
    memcpy (&a[0], &a[2], 5 * sizeof(int)); // problem: overlapping src, dst
    return 0;
}
```

- analysis:
  - problem:

```
==1270== Source and destination overlap in memcpy(0x7fefff670, 0x7fefff678, 20)
==1270==    at 0x4C2E1A0: memcpy@@GLIBC_2.14 (in /usr/lib64/valgrind/vgpreload_m
emcheck-amd64-linux.so)
==1270==    by 0x400577: main (overlapping_blocks_memcpy.c:6)
```

http://valgrind.org/docs/manual/mc-manual.html#mc-manual.overlap

Memcheck & memory leak detection

- source:

```c
#include <stdlib.h>

int main()
{
    int * a = malloc(10 * sizeof(int));
    a = malloc(10 * sizeof(int));    // problem 1: memory leak
    return 0;                        // problem 2: another memory leak
}
```

- analysis:
  - problem:

```
==1500== LEAK SUMMARY:
==1500==    definitely lost: 80 bytes in 2 blocks
==1500==    indirectly lost: 0 bytes in 0 blocks
==1500==      possibly lost: 0 bytes in 0 blocks
==1500==    still reachable: 0 bytes in 0 blocks
==1500==         suppressed: 0 bytes in 0 blocks
==1500== Rerun with --leak-check=full to see details of leaked memory
```

http://valgrind.org/docs/manual/mc-manual.html#mc-manual.leaks

Useful options

- `--leak-check=<no|summary|yes|full> [default: summary]`
  - if set to `summary`, it says how many leaks occurred
  - if set to `full` or `yes`, it also gives details of each individual leak

```
==6351== Invalid write of size 4
==6351==    at 0x400552: f (interpreting_output.c:6)
==6351==    by 0x400562: main (interpreting_output.c:11)
==6351==  Address 0x51df068 is 0 bytes after a block of size 40 alloc'd
==6351==    at 0x4C2C66D: malloc (in /usr/lib64/valgrind/vgpreload_memcheck-amd6
4-linux.so)
==6351==    by 0x400545: f (interpreting_output.c:5)
==6351==    by 0x400562: main (interpreting_output.c:11)
```

- `-v, --verbose`
  - gives extra information on various aspects of your program:
    - shared objects loaded
    - suppressions used
    - progress of the instrumentation and execution engines
    - warnings about unusual behaviour.

http://valgrind.org/docs/manual/mc-manual.html#opt.leak-check
http://valgrind.org/docs/manual/manual-core.html#opt.verbose

Useful options

- `--read-var-info=<yes|no> [default: no]`
  - gets variable types and locations from `DWARF3` debug info
  - causes slowdown
  - uses more memory
  - `memcheck`, `helgrind`, `DRD` show more precise error messages
- `--track-origins=<yes|no> [default: no]`
  - controls whether `memcheck` tracks the origin of uninitialised values
  - If set to `yes`, when an uninitialised value error is reported, `memcheck` will try to show the origin of the value.

```
==29553==  Uninitialised value was created by a stack allocation
==29553==    at 0x400968: main (a.c:5)
```

http://valgrind.org/docs/manual/manual-core.html#opt.read-var-info
http://valgrind.org/docs/manual/mc-manual.html#opt.track-origins

Valgrind & slowdown demo

- execution vs. execution under valgrind

```
u-pl22@64:~/BIG/SDMT/VALGRIND/memcheck$ time ./illegal_read_write

real    0m0.001s
user    0m0.000s
sys     0m0.000s
u-pl22@64:~/BIG/SDMT/VALGRIND/memcheck$ time valgrind ./illegal_read_write 2> /dev/null

real    0m0.281s
user    0m0.266s
sys     0m0.014s
```

http://valgrind.org/docs/manual/manual-core.html#manual-core.whatdoes

Valgrind & suppressing errors

- existence of unwanted errors such as:
  - false positives
  - errors found in system libraries

- `/usr/lib/valgrind/default.supp`
  - at startup valgrind reads a default list of errors to suppress
  - this file is created by the `./configure` script

- `--suppressions=<filename>`
  - reads additional supression files

- `--v`
  - Valgrind prints out one line for each used suppression
  - the line holds the suppression name and the number of its usage

http://valgrind.org/docs/manual/manual-core.html#manual-core.suppress
http://wiki.wxwidgets.org/Valgrind_Suppression_File_Howto

Valgrind & suppressing errors

- `--gen-suppressions=<yes|no|all> [default: no]`
  - if set to `all`, valgrind prints a supression after each reporter error

```
==27151== Invalid free() / delete / delete[] / realloc()
==27151==    at 0x4C2B35C: free (in /usr/lib64/valgrind/vgpreload_memcheck-amd64
-linux.so)
==27151==    by 0x4005B1: main (illegal_free.c:7)
==27151==  Address 0x51df040 is 0 bytes inside a block of size 40 free'd
==27151==    at 0x4C2B35C: free (in /usr/lib64/valgrind/vgpreload_memcheck-amd64
-linux.so)
==27151==    by 0x4005A5: main (illegal_free.c:6)
==27151==
{
   <insert_a_suppression_name_here>
   Memcheck:Free
   fun:free
   fun:main
}
```

http://valgrind.org/docs/manual/manual-core.html#manual-core.suppress
http://wiki.wxwidgets.org/Valgrind_Suppression_File_Howto

Valgrind & suppressing errors

- suppression example:

```
{
    __gconv_transform_ascii_internal/__mbrtowc/mbtowc
    Memcheck:Value4
    fun:__gconv_transform_ascii_internal
    fun:__mbr*toc
    fun:mbtowc
}
```

- interpretation:
  - for Memcheck only
  - suppress a use-of-uninitialised-value error
    - when the data size is 4 bytes
    - when it occurs in the function __gconv_transform_ascii_internal
    - when that is called from any function of name matching __mbr*toc
    - when that is called from mbtowc
  - identification string for user: __gconv_transform_ascii_internal/__mbrtowc/mbtowc.

http://valgrind.org/docs/manual/manual-core.html#manual-core.suppress
http://valgrind.org/docs/manual/mc-manual.html#mc-manual.suppfiles

Helgrind, DRD

- tools for finding race conditions and pthreads API misuses
  - model of program memory accesses helps to find missing locks
  - use of different models so the set of bugs may vary
  - false positives may occur

- invocations:

```
valgrind --tool=helgrind <prog> [prog-options]
valgrind --tool=DRD <prog> [prog-options]
```

http://valgrind.org/docs/manual/hg-manual.html#hg-manual.overview
http://valgrind.org/docs/manual/drd-manual.html#drd-manual.overview

Helgrind, DRD & data race

- source:

```
1   #include <pthread.h>
2
3   int var = 0;
4
5   void* child_fn ( void* arg ) {
6       var++; /* Unprotected relative to parent */
7       return NULL;
8   }
9
10  int main ( void ) {
11      pthread_t child;
12      pthread_create(&child, NULL, child_fn, NULL);
13      var++; /* Unprotected relative to child */
14      pthread_join(child, NULL);
15      return 0;
16  }
```

http://valgrind.org/docs/manual/hg-manual.html#hg-manual.data-races

Helgrind & data race

- analysis:

```
==15651== Thread #2 was created
==15651==    at 0x513E95E: clone (in /lib64/libc-2.15.so)
==15651==    by 0x4E3D05F: do_clone.clone.2 (in /lib64/libpthread-2.15.so)
==15651==    by 0x4E3E490: pthread_create@@GLIBC_2.2.5 (in /lib64/libpthread-2.1
5.so)
==15651==    by 0x4C2ED44: ??? (in /usr/lib64/valgrind/vgpreload_helgrind-amd64-
linux.so)
==15651==    by 0x400634: main (data_race.c:12)
==15651==
==15651== ----------------------------------------------------------------
==15651==
==15651== Possible data race during read of size 4 at 0x601038 by thread #1
==15651== Locks held: none
==15651==    at 0x400635: main (data_race.c:13)
==15651==
==15651== This conflicts with a previous write of size 4 by thread #2
==15651== Locks held: none
==15651==    at 0x400605: child_fn (data_race.c:6)
==15651==    by 0x4C2EEDD: ??? (in /usr/lib64/valgrind/vgpreload_helgrind-amd64-
linux.so)
==15651==    by 0x4E3DDA5: start_thread (in /lib64/libpthread-2.15.so)
==15651==
==15651== ----------------------------------------------------------------
==15651==
```

http://valgrind.org/docs/manual/hg-manual.html#hg-manual.data-races

DRD & data race

- analysis:

```
==15807== Conflicting load by thread 1 at 0x00601038 size 4
==15807==    at 0x400635: main (data_race.c:13)
==15807== Allocation context: BSS section of f.cuni.cz/u/k/kudelaja/BIG/SDMT/VAL
GRIND/helgrind_drd/data_race
==15807== Other segment start (thread 2)
==15807==    at 0x4C325FB: pthread_mutex_unlock (in /usr/lib64/valgrind/vgpreloa
d_drd-amd64-linux.so)
==15807==    by 0x4C2EEAE: ??? (in /usr/lib64/valgrind/vgpreload_drd-amd64-linux
.so)
==15807==    by 0x4E48DA5: start_thread (in /lib64/libpthread-2.15.so)
==15807== Other segment end (thread 2)
==15807==    at 0x51466F7: madvise (in /lib64/libc-2.15.so)
==15807==    by 0x4E48E80: start_thread (in /lib64/libpthread-2.15.so)
==15807==
==15807== Conflicting store by thread 1 at 0x00601038 size 4
==15807==    at 0x40063E: main (data_race.c:13)
==15807== Allocation context: BSS section of f.cuni.cz/u/k/kudelaja/BIG/SDMT/VAL
GRIND/helgrind_drd/data_race
==15807== Other segment start (thread 2)
==15807==    at 0x4C325FB: pthread_mutex_unlock (in /usr/lib64/valgrind/vgpreloa
d_drd-amd64-linux.so)
==15807==    by 0x4C2EEAE: ??? (in /usr/lib64/valgrind/vgpreload_drd-amd64-linux
.so)
==15807==    by 0x4E48DA5: start_thread (in /lib64/libpthread-2.15.so)
```

http://valgrind.org/docs/manual/drd-manual.html#drd-manual.data-races

Useful links

- http://valgrind.org/docs/manual/quick-start.html
- http://valgrind.org/docs/manual/manual.html
- http://valgrind.org/docs/manual/manual-core.html
- http://valgrind.org/docs/manual/mc-manual.html
- http://valgrind.org/docs/manual/hg-manual.html
- http://valgrind.org/docs/manual/drd-manual.html

Important notice

- somehow only 64-bit `valgrind` variant runs in labs
  - try using command `gentoo64` best before compiling sources

Q & A

thanks for your attention