The Valgrind Memory Checker. (i.e., Your best friend.)

Mike Closson

Dept of Computing Science
University of Alberta
modifications by Stef Nychka

October 26, 2009

Attribution.

Some of the material in this presentation was shamelessly stolen from the Valgrind website.

What is Valgrind?

- Valgrind is a "program-execution monitoring framework".
- Valgrind comes with many tools, the tool you will use most often is the *memcheck* tool.
- Memcheck will detect and report the following types of memory errors:
  - Use of uninitialized memory.
  - Reading/writing to memory after it has been freed.
  - Reading/writing off the end of malloc'd blocks.
  - Reading/writing inappropriate areas on the stack.
  - Overlapping src and dest pointers in memcpy() and related functions.
  - other stuff...

How do you pronounce Valgrind?

- The "Val" as in the world "value". The "grind" is pronounced with a short 'i' – ie. "grinned" (rhymes with "tinned") rather than "grind" (rhymes with "find").
- Valgrind comes from Nordic mythology. Valgrind is the name of the main entrance to Valhalla (the Hall of the Chosen Slain in Asgard). Over this entrance there resides a wolf and over it there is the head of a boar and on it perches a huge eagle, whose eyes can see to the far regions of the nine worlds. Only those judged worthy by the guardians are allowed to pass through Valgrind. All others are refused entrance.

Why should I use Valgrind?

- Valgrind will tell you about tough to find bugs early!
- Valgrind is very thorough.
- You may be tempted to think that Valgrind is too picky, since your program may seem to work even when valgrind complains. It is the author's experience that fixing *ALL* Valgrind complaints will save you time in the long run.

Is there a downside to using Valgrind?

- Valgrind is kind-of like a virtual x86 interpreter. So your program will run 10 to 30 times slower than normal. But for the kind of programs we write in 201, this won't be a problem.

What doesn't Valgrind do?

- Valgrind won't check static arrays: from the Valgrind FAQ:

  Unfortunately, Memcheck doesn't do bounds checking on static or stack arrays. We'd like to, but it's just not possible to do in a reasonable way that fits with how Memcheck works. Sorry.

- (There is now an experimental `exp-ptrcheck` valgrind tool which does this. It doesn't seem useful currently.)

What is the difference between Valgrind and GDB?

What is the difference between Valgrind and GDB?

- GDB is a debugger, Valgrind is a memory checker (among other things).
- Valgrind won't let you step interactively through a program.
- GDB doesn't check for use of uninitialized values, or over/underflowing dynamic memory.
- Both GDB and Valgrind will show you the line number of a segfault.
- Valgrind often shows the cause of a segfault, too.

It is the author's experience that most bugs are found and fixed faster using Valgrind than GDB.

"Valgrind is producing too much output."

Often Valgrind produces so much output that the important sections disappear off the top of the terminal. Here are three suggestions to cope:

1. If in a Terminal, use the scroll bar or Shift-PageUp and Shift-PageDown.
2. In an xterm, use Shift-PageUp and Shift-PageDown to scroll up and down.
3. If you are using xterm, enable the scrollbar by clicking Ctrl-Middle mouse button and select "Enable Scrollbar". Use the middle mouse button to select the ScrollBar.
4. Re-direct the output to a file, and view the file with less.
   `valgrind .... > out 2>&1`

Example #1

Download valgrind.tar.gz from the Presentations page, and extract its contents.
What is the error in ov.c?

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
int main() {
  char *buf = malloc( 3 );
  strcpy( buf, "012345" );
  printf("buf: %s\n", buf );
  return 0;
}
```

Make sure you compile with -g.

```
$ gcc -g -o ov ov.c
$
```

Here is valgrind's output. (Yours may differ slightly.)

```
$ valgrind ./ov
==2334== Memcheck, a memory error detector.
==2334== Copyright (C) 2002-2005, and GNU GPL'd, by Julian Seward et al.
==2334== Using LibVEX rev 1471, a library for dynamic binary translation.
==2334== Copyright (C) 2004-2005, and GNU GPL'd, by OpenWorks LLP.
==2334== Using valgrind-3.1.0, a dynamic binary instrumentation framework.
==2334== Copyright (C) 2000-2005, and GNU GPL'd, by Julian Seward et al.
==2334== For more details, rerun with: -v
==2334==
==2334== Invalid write of size 1
==2334==    at 0x401D131: strcpy (in ....
==2334==    by 0x8048423: main (ov.c:6)
==2334==  Address 0x414A02B is 0 bytes after a block of size 3 alloc'd
==2334==    at 0x401B477: malloc (in ....
==2334==    by 0x804840D: main (ov.c:5)
$
....
```

What did valgrind tell us?

- There was an invalid write (i.e., overflow) on line 6.
- The overflow occurred after a malloc'd block of 3 bytes. (Notice we malloc'd 3 bytes on line 5.)

```
5:  char *buf = malloc( 3 );
6:  strcpy( buf, "012345" );
7:  printf("buf: %s\n", buf );
8:  return 0;
```

Example #2

Compile and run index.c. No error is reported, but the program is incorrect.
Now run valgrind. (Your output may differ slightly.)

```
==29789== Invalid write of size 4
==29789==    at 0x804841B: main (index.c:11)
==29789==  Address 0x1BA570D0 is 0 bytes after a block of size 40 alloc'd
==29789==    at 0x1B9008A9: malloc (vg_replace_malloc.c:149)
==29789==    by 0x80483D9: main (index.c:8)
```

What is the error?

Other Valgrind resources.

- For another intoductory tutorial on valgrind, including a little more explanation on how to understand valgrind's output, see: http://www.valgrind.org/docs/

Conclusion

- For these small examples, the error may be obvious, but for big programs it's nice that valgrind will just tell us the error.
- Note understanding the errors is helpful (see the valgrind website), yet not always needed. Knowing the line number of a problem is often enough information on its own.
- Valgrind's –leak-check option, and perhaps others, don't get along with memwatch. Just use –tool=memcheck (done by default when no arguments), and avoid other options, or at least use them with care, when using memwatch.