CS2720 Practical Software Development

Valgrind Tutorial Spring 2011

Instructor: Rex Forsyth
Office: C-558
E-mail: forsyth@cs.uleth.ca
Tel: 329-2496

Tutorial Web Page: http://www.cs.uleth.ca/~forsyth/cs2720/lab/lab.html

Memory Errors

- segmentation fault
- bus error
- abort
- memory leaks
- unintialized variables
- illegal frees, mismatched frees

It would be nice to have a tool that would check for these errors and help identify the problem.

There is such a tool : **valgrind**

This is a suite of memory checking and profiling tools, but primarily used for its memory checking ability

How does it work?

- creates a virtual machine and simulates your program
- slows execution time down by a factor of 100 or more
- BUT it checks!!
- code must be compiled so that the virtual machine can be constructed and errors can be identified (where and what)

To use valgrind :

- compile your program with the **-g** option
- type

  ```
  valgrind --tool=memcheck progname [args]
  ```

- The default tool is memcheck so the more usual way to start is just

  ```
  valgrind progname [args]
  ```

- your program will execute under valgrinds control
- valgrind will report any errors
- if you need more information, valgrind usually tells you what options to include to obtain this information.

valgrind has an extensive memory checker and will check for several things:

- illegal reads and writes
  - out of bounds array access on the heap
  - accessing *deleted* memory

  If a segmentation fault would happen, it still happens but valgrind provides a message first.
- using uninitialized variables
  - on the stack
  - on the heap
- Illegal frees
  - trying to free from the stack
  - already freed
- mismatched frees
  - **new** with **delete []**
  - **new []** with **delete**

- system calls with invalid data : *write*, *exit*
- overlapping source and destination blocks
- leak checking
  1. still reachable
     - pointer to the block but not freed when program exited
  2. definitely lost
     - direct – no pointer
     - indirect – pointed at by a pointer from leaked memory
  3. possibly lost
     - pointer to the middle of a block