Valgrind

Concurrent Programming

Introduction

- What is Valgrind?
- Installing Valgrind
- How to use?
- Practice

What is Valgrind?

- A framework for heavyweight dynamic binary instrumentation(DBI)
- Detecting memory management and thread bugs, and profile program
  - a memory error detector (Memcheck)
  - two thread error detectors (Helgrind)
  - a cache and branch-prediction profiler (Cachegrind)
  - a call-graph generating cache and branch-prediction profiler (Callgrind)
  - a heap profiler (Massif)
  - …
- Supporting various system platform
- Open Source / Freeware Software GNU GPL V2

What is Valgrind? - Memcheck

- What can we detect with Memcheck?
  - Accessing memory you shouldn’t
  - Using undefined values
  - Incorrect freeing of heap memory
  - Overlapping src and dst pointers in memcpy and related functions
  - Passing a fishy value to the size parameter of a memory allocation function
  - Memory leaks

What is Valgrind? - Helgrind

- What can we detect with Helgrind?
  - Misuses of the POSIX pthreads API
  - Potential deadlocks arising from lock ordering problems
  - Data races
    - Accessing memory without adequate locking or synchronization

Installing Valgrind

```
$ sudo apt-get install valgrind
```

How to use?

- valgrind --tool=name [options] prog-and-args
  - tool = [memcheck, cachegrind, callgrind, massif, helgrind, drd, nulgrind, etc..]

- Memcheck

```
$ valgrind --tool=memcheck --leak-check=yes ./a.out
```

- Helgrind

```
$ valgrind --tool=helgrind ./a.out
```

* Compile with -g option, if you want to see line numbers that occur problem

Test code for Memcheck

< memcheck_test.cpp >

```c
#include <stdio.h>
#include <stdlib.h>

int main(void) {
    int p, t;
    char *pc;
    if (p == 5) {
        t = p + 1;
    }

    printf("%d is not initialized\n", p);

    pc = (char *)malloc(sizeof(char)* 10);

    return 0;
}
```

Memcheck result

```
$ g++ -g –o memcheck_test memcheck_test.cpp
```

```
$ valgrind --tool=memcheck --leak-check=full ./memcheck_test
```

```
==30070== Use of uninitialised value of size 8
==30070==    at 0x4E80A4B: _itoa_word (_itoa.c:179)
==30070==    by 0x4E846F6: vfprintf (vfprintf.c:1660)
==30070==    by 0x4E8B498: printf (printf.c:33)
==30070==    by 0x4005A7: main (memcheck_test.cpp:11)
==30070==
```

```
==30070== HEAP SUMMARY:
==30070==     in use at exit: 10 bytes in 1 blocks
==30070==   total heap usage: 1 allocs, 0 frees, 10 bytes allocated
==30070==
==30070== 10 bytes in 1 blocks are definitely lost in loss record 1 of 1
==30070==    at 0x4C2AB80: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.so)
==30070==    by 0x4005B4: main (memcheck_test.cpp:13)
==30070==
==30070== LEAK SUMMARY:
==30070==    definitely lost: 10 bytes in 1 blocks
==30070==    indirectly lost: 0 bytes in 0 blocks
==30070==      possibly lost: 0 bytes in 0 blocks
==30070==    still reachable: 0 bytes in 0 blocks
==30070==         suppressed: 0 bytes in 0 blocks
```

Memcheck - memory leak type

- “Definitely lost”
  - Your program is leaking meory – fix those leaks!
- ”Indirectly lost”
  - Your program is leaking memory in a pointer-based structure.
- ”Possibly lost”
  - your program is leaking memory, unless you're doing unusual things with pointers that could cause them to point into the middle of an allocated block
- ”Still reachable”
  - your program is probably ok
- “Suppressed”
  - Those are leaks outside of your code

Test code for Helgrind

< helgrind_test.cpp >

```cpp
#include <pthread.h>

int var = 0;

void *thread_func(void *arg) {
    var++;
    return NULL;
}

int main(void) {
    pthread_t child;
    pthread_create(&child, NULL, thread_func, NULL);
    var++;
    pthread_join(child, NULL);
    return 0;
}
```

Test code for Helgrind

```
$ g++ -g -o helgrind_test helgrind_test.cpp -lpthread
```

```
$ valgrind --tool=helgrind ./helgrind_test
```

```
==32496== ----------------------------------------------------------------
==32496==
==32496== Possible data race during read of size 4 at 0x60104C by thread #1
==32496== Locks held: none
==32496==    at 0x40068E: main (helgrind_test.cpp:13)
==32496==
==32496== This conflicts with a previous write of size 4 by thread #2
==32496== Locks held: none
==32496==    at 0x40065E: thread_func(void*) (helgrind_test.cpp:6)
==32496==    by 0x4C30FA6: ??? (in /usr/lib/valgrind/vgpreload_helgrind-amd64-linux.so)
==32496==    by 0x4E45181: start_thread (pthread_create.c:312)
==32496==    by 0x515547C: clone (clone.S:111)
==32496==  Address 0x60104c is 0 bytes inside data symbol "var"
==32496==
==32496== ----------------------------------------------------------------
```

Practice

- Practice on the prime_cond_err.cpp uploaded at Piazza page

- With valgrind, find three problems
  - 2 memory problems - use memcheck
  - 1 race condition problem - use helgrind

Thank You