Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №2 по курсу «Дискретный анализ»

Студент: И. В. Кочкожаров
Группа: М8О-208Б-22
Дата:
Оценка:
Подпись:

Москва, 2024

1 Дневник отладки

Работа разрабатывалась через тестирование. Сначла были написаны тесты отдельных методов Patricia Trie (около 40 штук) с использованием библиотеки gtest, а только потом писались реализации этих методов. Таким образом получалось избежать проблем с утечками памяти, это всегда легко обнаруживалось и справлялось с помощью тестов, запускаемых через утилиту Valgrind (*valgrind –leak-check=full*)

Однако одну проблему, которую обнаруживал Valgrind долго не удавалось решить:

```
==139533== Syscall param write(buf) points to uninitialised byte(s)
==139533==    at 0x4CD8D30: write (write.c:26)
==139533==    by 0x4C59164: _IO_file_write@@GLIBC_2.2.5 (fileops.c:1181)
==139533==    by 0x4C5726E: new_do_write (fileops.c:449)
==139533==    by 0x4C58148: _IO_do_write@@GLIBC_2.2.5 (fileops.c:426)
==139533==    by 0x4C5797F: _IO_file_close_it@@GLIBC_2.2.5 (fileops.c:135)
==139533==    by 0x4C4B442: fclose@@GLIBC_2.2.5 (iofclose.c:53)
==139533==    by 0x109BBC: TFile::~TFile() (in /home/ivan/cs/da-labs/build/lab2/lab2)
==139533==    by 0x109664: main (in /home/ivan/cs/da-labs/build/lab2/lab2)
==139533==  Address 0x4dd3b7c is 12 bytes inside a block of size 4,096 alloc'd
==139533==    at 0x4841814: malloc (vg_replace_malloc.c:431)
==139533==    by 0x4C4B258: _IO_file_doallocate (filedoalloc.c:101)
==139533==    by 0x4C5A6E8: _IO_doallocbuf (genops.c:347)
==139533==    by 0x4C5A6E8: _IO_doallocbuf (genops.c:342)
==139533==    by 0x4C58787: _IO_file_overflow@@GLIBC_2.2.5 (fileops.c:745)
==139533==    by 0x4C592DE: _IO_new_file_xsputn (fileops.c:1244)
==139533==    by 0x4C592DE: _IO_file_xsputn@@GLIBC_2.2.5 (fileops.c:1197)
==139533==    by 0x4C4C93D: fwrite (iofwrite.c:39)
==139533==    by 0x10A522: TPatriciaTrie::SaveToFile(_IO_FILE*) const (in /home/ivan/c
==139533==    by 0x10961F: main (in /home/ivan/cs/da-labs/build/lab2/lab2)
==139533==
```

Подобные ошибки происходили, каждый раз, когда происходила запись структуры узла в специальном виде в бинарный файл. Эта структура содержала в себе буфер из 257 *char*, и даже инициализация этой структуры нулями не помогала. Выход был найден такой - структура была максимально упрощена до обычного C-шного POD-а, к которому применялся системный вызов *memset* после создания. После такого ошибка перестала появляться, предполагается, что проблема была в том, что запимсывались неинииализированные байты, используемые для выравнивания большой структуры.

Так же была использована утилита *gprof* для построения наглядной таблицы вызовов.

```
time  self seconds calls  name
52.65   0.10    15433800 GetBitByIndex(TCaseInsensitiveString const&,int)
10.53   0.02    21515816 TCaseInsensitiveString::size() const
10.53   0.02       89173 TPatriciaTrie::FindPreviousNode(TCaseInsensitiveString
const&,int)
5.26   0.01    21396508 GetBitSize(TCaseInsensitiveString const&)
5.26   0.01      149100 TCaseInsensitiveString::TCaseInsensitiveString(char
const*)
5.26   0.01       59654 GetBitDifference
(TCaseInsensitiveString const&,TCaseInsensitiveString const&)
5.26   0.01             TPatriciaTrie::LoadFromFile(_IO_FILE*)
5.26   0.01             std::remove_reference<unsigned long&>::
type&& std::move<unsigned long&>(unsigned long&)
0.00   0.00    15463615 TCaseInsensitiveString::CStr() const
0.00   0.00      119285 operator==(
TCaseInsensitiveString const&,TCaseInsensitiveString const&)
0.00   0.00       89496 TCaseInsensitiveString::Scan(_IO_FILE*)
0.00   0.00       89173 TPatriciaTrie::FindNode
(TCaseInsensitiveString const&,int)
0.00   0.00       89173 TPair<TPatriciaTrie::TNode*,int>::
TPair(TPatriciaTrie::TNode* const&,int const&)
0.00   0.00       29841 TCaseInsensitiveString::TCaseInsensitiveString()
0.00   0.00       29840 TPatriciaTrie::Insert(
TPair<TCaseInsensitiveString,unsigned long>const&)
0.00   0.00       29840 TPair<TCaseInsensitiveString,unsigned long>::TPair
(TCaseInsensitiveString const&,unsigned long const&)
0.00   0.00       29815 TPatriciaTrie::Find(TCaseInsensitiveString const&)
0.00   0.00       29520 TPatriciaTrie::TNode::TNode
(TPair<TCaseInsensitiveString,unsigned long>const&)
0.00   0.00       29520 TPair<TCaseInsensitiveString,unsigned long>::TPair
(TPair<TCaseInsensitiveString,unsigned long>const&)
0.00   0.00           1 TPatriciaTrie::DestroyTrie(TPatriciaTrie::TNode*)
0.00   0.00           1 TPatriciaTrie::TPatriciaTrie()
0.00   0.00           1 TPatriciaTrie::~TPatriciaTrie()
```

Видно, что больше всего процессорного времени на вызовы функции *GetBitByIndex*.

2 Тесты

Тесты написаны с использованием библиотеки *gtest*

```
1
2  TEST(patricia_test, modifier01) {
3      TPatriciaTrie p;
4      p.Insert({"a", 1});
5      EXPECT_EQ(p.Find("a")->value, 1);
6      EXPECT_EQ(p.Size(), 1);
7  }
8
9  TEST(patricia_test, modifier02) {
10     TPatriciaTrie p;
11     p.Insert({"a", 1});
12     EXPECT_FALSE(p.Insert({"A", 2}));
13 }
14
15 TEST(patricia_test, modifier03) {
16     TPatriciaTrie p;
17     p.Insert({"abc", 10});
18     EXPECT_EQ(p.Size(), 1);
19     EXPECT_FALSE(p.Insert({"abc", 20}));
20 }
21
22 TEST(patricia_test, modifier04) {
23     TPatriciaTrie p;
24     p.Insert({"ab", 10});
25     p.Insert({"abc", 20});
26     EXPECT_EQ(p.Size(), 2);
27     EXPECT_EQ(p.Find("ab")->value, 10);
28     EXPECT_EQ(p.Find("abc")->value, 20);
29 }
```

Вывод в консоль после запуска тестов через valgrind

```
==149268== Memcheck,a memory error detector
==149268== Copyright (C) 2002-2022,and GNU GPL'd,by Julian Seward et al.
==149268== Using Valgrind-3.21.0 and LibVEX; rerun with -h for copyright info
==149268== Command: ./lab2_tests
==149268==
Running main() from /var/tmp/portage/dev-cpp/gtest-1.13.0/work/
googletest-1.13.0/googletest/src/gtest_main.cc
[==========] Running 38 tests from 2 test suites.
[----------] Global test environment set-up.
[----------] 2 tests from binary_string_test
[ RUN      ] binary_string_test.bitdifftest01
```

```
[       OK ] binary_string_test.bitdifftest01 (8 ms)
[ RUN      ] binary_string_test.bitdifftest02
[       OK ] binary_string_test.bitdifftest02 (1 ms)
[----------] 2 tests from binary_string_test (13 ms total)

[----------] 36 tests from patricia_test
[ RUN      ] patricia_test.modifier01
[       OK ] patricia_test.modifier01 (5 ms)
[ RUN      ] patricia_test.modifier02
[       OK ] patricia_test.modifier02 (3 ms)
[ RUN      ] patricia_test.modifier03
[       OK ] patricia_test.modifier03 (2 ms)
[ RUN      ] patricia_test.modifier04
[       OK ] patricia_test.modifier04 (3 ms)
[ RUN      ] patricia_test.modifier05
[       OK ] patricia_test.modifier05 (2 ms)
[ RUN      ] patricia_test.modifier06
[       OK ] patricia_test.modifier06 (4 ms)
[ RUN      ] patricia_test.modifier07
[       OK ] patricia_test.modifier07 (3 ms)
[ RUN      ] patricia_test.modifier08
[       OK ] patricia_test.modifier08 (1 ms)
[ RUN      ] patricia_test.modifier09
[       OK ] patricia_test.modifier09 (1 ms)
[ RUN      ] patricia_test.modifier10
[       OK ] patricia_test.modifier10 (4 ms)
[ RUN      ] patricia_test.modifier11
[       OK ] patricia_test.modifier11 (3 ms)
[ RUN      ] patricia_test.modifier12
[       OK ] patricia_test.modifier12 (7 ms)
[ RUN      ] patricia_test.modifier13
[       OK ] patricia_test.modifier13 (6 ms)
[ RUN      ] patricia_test.modifier14
[       OK ] patricia_test.modifier14 (6 ms)
[ RUN      ] patricia_test.modifier15
[       OK ] patricia_test.modifier15 (1867 ms)
[ RUN      ] patricia_test.modifier16
[       OK ] patricia_test.modifier16 (1209 ms)
[ RUN      ] patricia_test.modifier17
[       OK ] patricia_test.modifier17 (1147 ms)
[ RUN      ] patricia_test.modifier18
```

```
[       OK ] patricia_test.modifier18 (8 ms)
[ RUN      ] patricia_test.insert01
[       OK ] patricia_test.insert01 (1 ms)
[ RUN      ] patricia_test.insert02
[       OK ] patricia_test.insert02 (2 ms)
[ RUN      ] patricia_test.insert03
[       OK ] patricia_test.insert03 (3 ms)
[ RUN      ] patricia_test.insert04
[       OK ] patricia_test.insert04 (1 ms)
[ RUN      ] patricia_test.erase01
[       OK ] patricia_test.erase01 (1 ms)
[ RUN      ] patricia_test.erase02
[       OK ] patricia_test.erase02 (2 ms)
[ RUN      ] patricia_test.erase03
[       OK ] patricia_test.erase03 (2 ms)
[ RUN      ] patricia_test.erase04
[       OK ] patricia_test.erase04 (5 ms)
[ RUN      ] patricia_test.erase05
[       OK ] patricia_test.erase05 (5 ms)
[ RUN      ] patricia_test.erase06
[       OK ] patricia_test.erase06 (5 ms)
[ RUN      ] patricia_test.erase07
[       OK ] patricia_test.erase07 (4 ms)
[ RUN      ] patricia_test.erase08
[       OK ] patricia_test.erase08 (3 ms)
[ RUN      ] patricia_test.erase09
[       OK ] patricia_test.erase09 (4 ms)
[ RUN      ] patricia_test.erase10
[       OK ] patricia_test.erase10 (4 ms)
[ RUN      ] patricia_test.file01
[       OK ] patricia_test.file01 (20 ms)
[ RUN      ] patricia_test.file02
[       OK ] patricia_test.file02 (7 ms)
[ RUN      ] patricia_test.file03
[       OK ] patricia_test.file03 (4 ms)
[ RUN      ] patricia_test.file04
[       OK ] patricia_test.file04 (1189 ms)
[----------] 36 tests from patricia_test (5568 ms total)

[----------] Global test environment tear-down
[==========] 38 tests from 2 test suites ran. (5610 ms total)
```

```
[  PASSED  ] 38 tests.
==149268==
==149268== HEAP SUMMARY:
==149268==     in use at exit: 311,335 bytes in 5 blocks
==149268==   total heap usage: 2,043 allocs,2,038 frees,928,529 bytes allocated
==149268==
==149268== LEAK SUMMARY:
==149268==    definitely lost: 0 bytes in 0 blocks
==149268==    indirectly lost: 0 bytes in 0 blocks
==149268==      possibly lost: 0 bytes in 0 blocks
==149268==    still reachable: 311,335 bytes in 5 blocks
==149268==         suppressed: 0 bytes in 0 blocks
==149268== Rerun with --leak-check=full to see details of leaked memory
==149268==
==149268== For lists of detected and suppressed errors,rerun with: -s
==149268== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Этот вывод valgrind говорит о том, что в программе нет утечек памяти.

3 Выводы

Для реализации словаря из предыдущей лабораторной работы, было проведено исследование скорости выполнения и потребления оперативной памяти. Были обнаружен недочеты и они были справлены. Valgrind очередной раз подтвердил своё удобство и полезность, а так же были получены навыки использования утилиты gprof для анализа вызовов функций в программе. Очевидно, что подход разработки через тестирование позволил написать код без критических ошибок..