Московский авиационный институт
(национальный исследовательский университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу «Дискретный анализ»

Студент: Т. Д. Голубев
Преподаватель: А. А. Кухтичев
Группа: М8О-206Б-22
Дата:
Оценка:
Подпись:

Москва, 2024

Лабораторная работа №3

Задача: Для реализации словаря из предыдущей лабораторной работы, необходимо провести исследование скорости выполнения и потребления оперативной памяти. В случае выявления ошибок или явных недочётов, требуется их исправить. Результатом лабораторной работы является отчёт, состоящий из:

- Дневника выполнения работы, в котором отражено что и когда делалось, какие средства использовались и какие результаты были достигнуты на каждом шаге выполнения лабораторной работы.
- Выводов о найденных недочётах.
- Сравнение работы исправленной программы с предыдущей версией.
- Общих выводов о выполнении лабораторной работы, полученном опыте.

Минимальный набор используемых средств должен содержать утилиту gprof и библиотеку dmalloc, однако их можно заменять на любые другие аналогичные или более развитые утилиты (например, Valgrind или Shark) или добавлять к ним новые (например, gcov).

1 Valgrind

Утилита Valgrind – мощное средство поиска ошибок работы с памятью. В его составе имеется некоторое количество дополнительных утилит, предназначенных для профилирования программ, анализа потребления памяти и поиска ошибок связанных с синхронизацией в многопоточных программах [1].

Запустив утилиту на своей программе, я обнаружил ошибку, связанную с чтением из файла.

```
Running main() from /home/cat_mood/programming/mai-da-labs/lab02/build/_deps/googletes
[==========] Running 52 tests from 2 test suites.
[----------] Global test environment set-up.
[----------] 15 tests from binary_string_test
[ RUN      ] binary_string_test.bitdifftest01
[       OK ] binary_string_test.bitdifftest01 (11 ms)
[ RUN      ] binary_string_test.bitdifftest02
[       OK ] binary_string_test.bitdifftest02 (1 ms)
[ RUN      ] binary_string_test.bitdifftest03
[       OK ] binary_string_test.bitdifftest03 (1 ms)
...
==19221== Invalid read of size 8
==19221==    at 0x486218A: TPatriciaTrie::SaveToFile(std::basic_ofstream<char,std::cha
const (in /home/cat_mood/programming/mai-da-labs/lab02/build/liblab02_lib.so)
==19221==    by 0x131E51: patricia_test_file01_Test::TestBody() (in /home/cat_mood/pro
==19221==    by 0x17F936: void testing::internal::HandleSehExceptionsInMethodIfSupport
(testing::Test::*)(),char const*) (in /home/cat_mood/programming/mai-da-labs/lab02/bui
==19221==    by 0x176A20: void testing::internal::HandleExceptionsInMethodIfSupported
(testing::Test::*)(),char const*) (in /home/cat_mood/programming/mai-da-labs/lab02/bui
==19221==    by 0x14F3AB: testing::Test::Run() (in /home/cat_mood/programming/mai-da-l
==19221==    by 0x14FF14: testing::TestInfo::Run() (in /home/cat_mood/programming/mai-
==19221==    by 0x150874: testing::TestSuite::Run() (in /home/cat_mood/programming/mai
==19221==    by 0x160CE2: testing::internal::UnitTestImpl::RunAllTests() (in
/home/cat_mood/programming/mai-da-labs/lab02/build/lab02_test)
==19221==    by 0x180FB1: bool testing::internal::HandleSehExceptionsInMethodIfSupport
(testing::internal::UnitTestImpl::*)(),char const*) (in /home/cat_mood/programming/mai
==19221==    by 0x177C54: bool testing::internal::HandleExceptionsInMethodIfSupported
(testing::internal::UnitTestImpl::*)(),char const*) (in /home/cat_mood/programming/mai
==19221==    by 0x15F25E: testing::UnitTest::Run() (in /home/cat_mood/programming/mai-
==19221==    by 0x13B5E1: RUN_ALL_TESTS() (in /home/cat_mood/programming/mai-da-labs/l
==19221==  Address 0x4e75e70 is 0 bytes after a block of size 64 alloc'd
==19221==    at 0x4849013: operator new(unsigned long) (in /usr/libexec/valgrind/vgpre
```

```
==19221==    by 0x486189A: TPatriciaTrie::Insert(TPair<std::__cxx11::basic_string<char
long>const&) (in /home/cat_mood/programming/mai-da-labs/lab02/build/liblab02_lib.so)
==19221==    by 0x131E0B: patricia_test_file01_Test::TestBody() (in /home/cat_mood/pro
==19221==    by 0x17F936: void testing::internal::HandleSehExceptionsInMethodIfSupport
(testing::Test::*)(),char const*) (in /home/cat_mood/programming/mai-da-labs/lab02/bui
==19221==    by 0x176A20: void testing::internal::HandleExceptionsInMethodIfSupported<
(testing::Test::*)(),char const*) (in /home/cat_mood/programming/mai-da-labs/lab02/bui
==19221==    by 0x14F3AB: testing::Test::Run() (in /home/cat_mood/programming/mai-da-l
==19221==    by 0x14FF14: testing::TestInfo::Run() (in /home/cat_mood/programming/mai-
==19221==    by 0x150874: testing::TestSuite::Run() (in /home/cat_mood/programming/mai
==19221==    by 0x160CE2: testing::internal::UnitTestImpl::RunAllTests() (in
/home/cat_mood/programming/mai-da-labs/lab02/build/lab02_test)
==19221==    by 0x180FB1: bool testing::internal::HandleSehExceptionsInMethodIfSupport
(testing::internal::UnitTestImpl::*)(),char const*) (in /home/cat_mood/programming/mai
==19221==    by 0x177C54: bool testing::internal::HandleExceptionsInMethodIfSupported<
(testing::internal::UnitTestImpl::*)(),char const*) (in /home/cat_mood/programming/mai
==19221==    by 0x15F25E: testing::UnitTest::Run() (in /home/cat_mood/programming/mai-
...
==19221== HEAP SUMMARY:
==19221==     in use at exit: 0 bytes in 0 blocks
==19221==   total heap usage: 4,683 allocs,4,683 frees,686,483 bytes allocated
==19221==
==19221== All heap blocks were freed --no leaks are possible
==19221==
==19221== Use --track-origins=yes to see where uninitialised values come from
==19221== For lists of detected and suppressed errors,rerun with: -s
==19221== ERROR SUMMARY: 3700 errors from 109 contexts (suppressed: 0 from
0)
```

Дело в том, что я считываю из файла в `struct TSaveData`. Так как структура в языке C++ значительно отличается от структуры в C (в C++ структура ялвяется классом с модификатором public по умолчанию), считывание в неё напрямую из файла вызывает ошибки. Так же я использовал фукнцию `memcpy` для копирования информации из строки. Она тоже вызывала ошибки.

Результат выполнения утилиты valgrind после исправления ошибок:

```
==42079== Memcheck,a memory error detector
==42079== Copyright (C) 2002-2017,and GNU GPL'd,by Julian Seward et al.
==42079== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==42079== Command: ./lab02_test
==42079==
```

```
Running main() from /home/cat_mood/programming/mai-da-labs/lab02/build/_deps/googletes
[==========] Running 52 tests from 2 test suites.
[----------] Global test environment set-up.
[----------] 15 tests from binary_string_test
[ RUN      ] binary_string_test.bitdifftest01
[       OK ] binary_string_test.bitdifftest01 (11 ms)
[ RUN      ] binary_string_test.bitdifftest02
[       OK ] binary_string_test.bitdifftest02 (1 ms)
[ RUN      ] binary_string_test.bitdifftest03
[       OK ] binary_string_test.bitdifftest03 (1 ms)
[ RUN      ] binary_string_test.bitdifftest04
[       OK ] binary_string_test.bitdifftest04 (1 ms)
[ RUN      ] binary_string_test.bitdifftest05
[       OK ] binary_string_test.bitdifftest05 (1 ms)
[ RUN      ] binary_string_test.bitdifftest06
[       OK ] binary_string_test.bitdifftest06 (1 ms)
[ RUN      ] binary_string_test.bitdifftest07
[       OK ] binary_string_test.bitdifftest07 (1 ms)
[ RUN      ] binary_string_test.bitdifftest08
[       OK ] binary_string_test.bitdifftest08 (1 ms)
[ RUN      ] binary_string_test.bitdifftest09
[       OK ] binary_string_test.bitdifftest09 (1 ms)
[ RUN      ] binary_string_test.bitindex01
[       OK ] binary_string_test.bitindex01 (2 ms)
[ RUN      ] binary_string_test.bitindex02
[       OK ] binary_string_test.bitindex02 (1 ms)
[ RUN      ] binary_string_test.bitindex03
[       OK ] binary_string_test.bitindex03 (1 ms)
[ RUN      ] binary_string_test.bitindex04
[       OK ] binary_string_test.bitindex04 (1 ms)
[ RUN      ] binary_string_test.bitindex05
[       OK ] binary_string_test.bitindex05 (20 ms)
[ RUN      ] binary_string_test.bitindex06
[       OK ] binary_string_test.bitindex06 (1 ms)
[----------] 15 tests from binary_string_test (57 ms total)

[----------] 37 tests from patricia_test
[ RUN      ] patricia_test.modifier01
[       OK ] patricia_test.modifier01 (4 ms)
[ RUN      ] patricia_test.modifier02
[       OK ] patricia_test.modifier02 (2 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 (3 ms)
[ RUN      ] patricia_test.modifier06
[       OK ] patricia_test.modifier06 (5 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 (4 ms)
[ RUN      ] patricia_test.modifier13
[       OK ] patricia_test.modifier13 (3 ms)
[ RUN      ] patricia_test.modifier14
[       OK ] patricia_test.modifier14 (4 ms)
[ RUN      ] patricia_test.modifier15
[       OK ] patricia_test.modifier15 (83 ms)
[ RUN      ] patricia_test.modifier16
[       OK ] patricia_test.modifier16 (65 ms)
[ RUN      ] patricia_test.modifier17
[       OK ] patricia_test.modifier17 (41 ms)
[ RUN      ] patricia_test.modifier18
[       OK ] patricia_test.modifier18 (12 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 (4 ms)
[ RUN      ] patricia_test.insert04
[       OK ] patricia_test.insert04 (2 ms)
[ RUN      ] patricia_test.erase01
```

```
[       OK ] patricia_test.erase01 (1 ms)
[ RUN      ] patricia_test.erase02
[       OK ] patricia_test.erase02 (3 ms)
[ RUN      ] patricia_test.erase03
[       OK ] patricia_test.erase03 (3 ms)
[ RUN      ] patricia_test.erase04
[       OK ] patricia_test.erase04 (7 ms)
[ RUN      ] patricia_test.erase05
[       OK ] patricia_test.erase05 (7 ms)
[ RUN      ] patricia_test.erase06
[       OK ] patricia_test.erase06 (6 ms)
[ RUN      ] patricia_test.erase07
[       OK ] patricia_test.erase07 (6 ms)
[ RUN      ] patricia_test.erase08
[       OK ] patricia_test.erase08 (4 ms)
[ RUN      ] patricia_test.erase09
[       OK ] patricia_test.erase09 (5 ms)
[ RUN      ] patricia_test.erase10
[       OK ] patricia_test.erase10 (5 ms)
[ RUN      ] patricia_test.file01
[       OK ] patricia_test.file01 (22 ms)
[ RUN      ] patricia_test.file02
[       OK ] patricia_test.file02 (3 ms)
[ RUN      ] patricia_test.file03
[       OK ] patricia_test.file03 (4 ms)
[ RUN      ] patricia_test.file04
[       OK ] patricia_test.file04 (83 ms)
[ RUN      ] patricia_test.file05
[       OK ] patricia_test.file05 (2 ms)
[----------] 37 tests from patricia_test (437 ms total)

[----------] Global test environment tear-down
[==========] 52 tests from 2 test suites ran. (525 ms total)
[  PASSED  ] 52 tests.
==42079==
==42079== HEAP SUMMARY:
==42079==     in use at exit: 0 bytes in 0 blocks
==42079==   total heap usage: 4,683 allocs,4,683 frees,683,411 bytes allocated
==42079==
==42079== All heap blocks were freed --no leaks are possible
==42079==
```

```
==42079== For lists of detected and suppressed errors,rerun with: -s
==42079== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

2 GProf

Gprof – это инструмент анализа производительности для приложений Unix [2]. Профилировщик собирает информацию о количестве времени, необходимом для выполнения определённой функции.

Диагностика программы:

```
cat_mood@nuclear-box:~/programming/mai-da-labs/lab02/build$ gprof ./lab02_exe
<../tests/e2e/01.t -p ./gmon.out
Flat profile:

Each sample counts as 0.01 seconds.
%   cumulative   self              self     total
time   seconds   seconds    calls  ns/call  ns/call  name
100.00      0.01     0.01   542932    18.42    18.42  std::__cxx11::basic_string<char
const
0.00      0.01     0.00   512787     0.00     0.00  std::__is_constant_evaluated()
0.00      0.01     0.00   188389     0.00     0.00  std::__cxx11::basic_string<char,s
const
0.00      0.01     0.00   164558     0.00     0.00  std::__cxx11::basic_string<char,s
0.00      0.01     0.00   145516     0.00     0.00  std::__ptr_traits_ptr_to<char*,cha
0.00      0.01     0.00   145516     0.00     0.00  std::__cxx11::basic_string<char,s
0.00      0.01     0.00   145516     0.00     0.00  char* std::__addressof<char>(char&
0.00      0.01     0.00   145516     0.00     0.00  char* std::addressof<char>(char&)
0.00      0.01     0.00   142312     0.00     0.00  std::__cxx11::basic_string<char,s
const
0.00      0.01     0.00   142312     0.00     0.00  std::allocator_traits<std::alloca
0.00      0.01     0.00   114397     0.00    18.42  std::__cxx11::basic_string<char,s
const
0.00      0.01     0.00   114397     0.00     0.00  std::__cxx11::basic_string<char,s
const
0.00      0.01     0.00   114397     0.00     0.00  std::__ptr_traits_ptr_to<char
const*,char const,false>::pointer_to(char const&)
0.00      0.01     0.00   114397     0.00     0.00  char const* std::__addressof<char
const>(char const&)
0.00      0.01     0.00   114397     0.00     0.00  char const* std::addressof<char
const>(char const&)
0.00      0.01     0.00   104711     0.00     0.00  std::char_traits<char>::assign(cha
const&)
0.00      0.01     0.00   104321     0.00    18.42  std::__cxx11::basic_string<char,s
long)
```

```
0.00      0.01     0.00   104321     0.00     0.00  std::__cxx11::basic_string<char,st
long)
0.00      0.01     0.00   102598     0.00    30.87  std::__cxx11::basic_string<char,st
0.00      0.01     0.00    95189     0.00     0.00  std::__new_allocator<char>::_M_max
const
0.00      0.01     0.00    95189     0.00     0.00  std::__new_allocator<char>::alloca
long,void const*)
0.00      0.01     0.00    95189     0.00     0.00  std::allocator_traits<std::allocat
long)
0.00      0.01     0.00    93016     0.00     0.00  std::__cxx11::basic_string<char,st
const
0.00      0.01     0.00    93016     0.00    18.42  std::__cxx11::basic_string<char,st
const*) const
0.00      0.01     0.00    93016     0.00     0.00  std::char_traits<char>::length(cha
const*)
0.00      0.01     0.00    93016     0.00     0.00  std::char_traits<char>::compare(ch
const*,char const*,unsigned long)
0.00      0.01     0.00    93016     0.00     0.00  unsigned long const& std::min<unsi
long>(unsigned long const&,unsigned long const&)
0.00      0.01     0.00    93016     0.00    18.42  bool std::operator==<char,std::cha
const*)
0.00      0.01     0.00    92522     0.00     0.00  std::__cxx11::basic_string<char,st
0.00      0.01     0.00    92522     0.00     0.00  std::__cxx11::basic_string<char,st
0.00      0.01     0.00    92522     0.00    30.87  std::__cxx11::basic_string<char,st
0.00      0.01     0.00    92522     0.00     0.00  std::remove_reference<std::allocat
std::move<std::allocator<char>&>(std::allocator<char>&)
0.00      0.01     0.00    69369     0.00     0.00  std::__new_allocator<char>::deallo
long)
0.00      0.01     0.00    69369     0.00     0.00  std::allocator_traits<std::allocat
long)
0.00      0.01     0.00    69369     0.00    18.42  std::__cxx11::basic_string<char,st
long)
0.00      0.01     0.00    57876     0.00     0.00  std::__cxx11::basic_string<char,st
const
0.00      0.01     0.00    57876     0.00     0.00  std::__cxx11::basic_string<char,st
const*,unsigned long)
0.00      0.01     0.00    57486     0.00     0.00  std::char_traits<char>::copy(char*
const*,unsigned long)
0.00      0.01     0.00    52994     0.00     0.00  std::is_constant_evaluated()
0.00      0.01     0.00    49604     0.00     0.00  std::__cxx11::basic_string<char,st
long)
```

```
0.00      0.01     0.00    49604     0.00     0.00  std::__cxx11::basic_string<char,s
0.00      0.01     0.00    49604     0.00     0.00  std::__cxx11::basic_string<char,s
long&,unsigned long)
0.00      0.01     0.00    46445     0.00    18.42  std::__cxx11::basic_string<char,s
0.00      0.01     0.00    46444     0.00    46.11  TPair<std::__cxx11::basic_string<
long>::~TPair()
0.00      0.01     0.00    46077     0.00     0.00  __gnu_cxx::__alloc_traits<std::al
0.00      0.01     0.00    46077     0.00     0.00  std::allocator_traits<std::alloca
0.00      0.01     0.00    46077     0.00    36.84  void std::__cxx11::basic_string<c
0.00      0.01     0.00    46077     0.00     0.00  std::__cxx11::basic_string<char,s
0.00      0.01     0.00    46077     0.00     0.00  std::iterator_traits<char*>::diff
std::__distance<char*>(char*,char*,std::random_access_iterator_tag)
0.00      0.01     0.00    46077     0.00     0.00  std::iterator_traits<char*>::itera
std::__iterator_category<char*>(char* const&)
0.00      0.01     0.00    46077     0.00     0.00  std::iterator_traits<char*>::diff
std::distance<char*>(char*,char*)
0.00      0.01     0.00    46077     0.00     0.00  std::__cxx11::basic_string<char,s
0.00      0.01     0.00    46077     0.00     0.00  std::__cxx11::basic_string<char,s
0.00      0.01     0.00    23286     0.00    18.42  TPair<std::__cxx11::basic_string<
long>::TPair()
0.00      0.01     0.00    23158     0.00   146.59  TPair<std::__cxx11::basic_string<
long>::TPair(std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator<cha
long const&)
0.00      0.01     0.00    23158     0.00     0.00  std::__cxx11::basic_string<char,s
long,unsigned long)
0.00      0.01     0.00    23158     0.00   146.59  std::__cxx11::basic_string<char,s
0.00      0.01     0.00    11799     0.00   100.04  TPair<std::__cxx11::basic_string<
long>::operator=(TPair<std::__cxx11::basic_string<char,std::char_traits<char>,std::all
long>const&)
0.00      0.01     0.00    11799     0.00     0.00  __gnu_cxx::__alloc_traits<std::al
0.00      0.01     0.00    11799     0.00    18.42  std::__cxx11::basic_string<char,s
const
0.00      0.01     0.00    11799     0.00   100.04  std::__cxx11::basic_string<char,s
0.00      0.01     0.00    11799     0.00   100.04  std::__cxx11::basic_string<char,s
0.00      0.01     0.00    11799     0.00   100.04  std::__cxx11::basic_string<char,s
0.00      0.01     0.00    11799     0.00     0.00  std::__cxx11::basic_string<char,s
std::__addressof<std::__cxx11::basic_string<char,std::char_traits<char>,std::allocator
0.00      0.01     0.00        1     0.00     0.00  __static_initialization_and_destru

%         the percentage of the total running time of the
time       program used by this function.
```

```
cumulative a running sum of the number of seconds accounted
seconds   for by this function and those listed above it.

self      the number of seconds accounted for by this
seconds    function alone.  This is the major sort for this
listing.

calls      the number of times this function was invoked,if
this function is profiled,else blank.

self      the average number of milliseconds spent in this
ms/call    function per call,if this function is profiled,
else blank.

total     the average number of milliseconds spent in this
ms/call    function and its descendents per call,if this
function is profiled,else blank.

name       the name of the function.  This is the minor sort
for this listing. The index shows the location of
the function in the gprof listing. If the index is
in parenthesis it shows where it would appear in
the gprof listing if it were to be printed.


Copyright (C) 2012-2022 Free Software Foundation,Inc.

Copying and distribution of this file,with or without modification,
are permitted in any medium without royalty provided the copyright
notice and this notice are preserved.
```

Как видно из отчёта профилировщика, большую часть времени занимают операции со `std::string`. Это связано с тем, что любое действие с деревом требует выполнение поиска по ключу, который является строкой. Отсюда и частые сравнения строк (найденной с текущей).

3 Дневник отладки

1. Использовал утилиту Valgrind, получил отчёт с ошибками
2. Исправил ошибки, убедился в этом, использовав Valgrind повторно.
3. Ознакомился с утилитой GProf.
4. Использовал GProf на реализации словаря. Сделал выводы.

4 Выводы

Выполнив третью лабораторную работу по курсу «Дискретный анализ», я выполнил отладку и профилирование своей программы, написанной в ходе второй лабораторной работы. Для этого я использовал такие утилиты, как Valgrind и GProf. Вторая утилита мне была ранее не знакома.

С помощью Valgrind я убедился в отсутствии утечек памяти в моей программе и нашёл некоторые ошибки при чтении из файла, которые я успешно исправил.

С помощью утилиты GProf я получил информацию о количестве времени выполнения каждой из функций. В моей лабораторной работе больше всего времени занимают операции со строками.

Вся эта информация полезна для поиска возможных ошибок в ходе разработки, а также для оптимизации уже отлаженной программы.

Список литературы

[1] OpenNET: статья – Выявление ошибок работы с памятью при помощи valgrind (memory debug profile gcc valgrind) // OpenNET URL: `https://www.opennet.ru/base/dev/valgrind_memory.txt.html` (дата обращения: 08.05.2024).

[2] Gprof – Wikipedia // Wikipedia URL: `https://en.wikipedia.org/wiki/Gprof` (дата обращения: 08.05.2024).