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: А. А. Боглаев
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.

Используемые утилиты: Valgrind, gprof.

1 Valgrind

Согласно [1], Valgrind — инструментальное программное обеспечение, предназначенное для отладки использования памяти, обнаружения утечек памяти, а также профилирования. Для поиска ошибок я использовал инструмент *memcheck*, а для поиска утечек использовал флаг $--leak-check=full$. С помощью данного флага можно получить более подробную информацию об утечках.

При реализации B-дерева я работал с памятью, выделял и освобождал ее. При попытке сдать работу в тестирующую систему, я получил ошибку времени выполнения. Стало понятно, что где-то есть проблема при работе с памятью. Чтобы найти эту проблему, я использовал Valgrind.

```
alex@wega:~/Рабочий стол/вуз/2course/da_labs/Lab_03$ valgrind --tool=memcheck
--leak-check=full ./lab2
==10805== Memcheck,a memory error detector
==10805== Copyright (C) 2002-2017,and GNU GPL'd,by Julian Seward et al.
==10805== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==10805== Command: ./lab2
==10805==
-dbfdfb
NoSuchWord
snfjsnf
NoSuchWord
==10805== Invalid read of size 1
==10805==    at 0x10BA10: TBTree::DeleteTree(TBTree::TNode*) (btree.hpp:103)
==10805==    by 0x10BAA9: TBTree::~TBTree() (btree.hpp:116)
==10805==    by 0x10E15B: main (main.cpp:78)
==10805==  Address 0x18 is not stack'd,malloc'd or (recently) free'd
==10805==
==10805==
==10805== Process terminating with default action of signal 11 (SIGSEGV)
==10805==  Access not within mapped region at address 0x18
==10805==    at 0x10BA10: TBTree::DeleteTree(TBTree::TNode*) (btree.hpp:103)
==10805==    by 0x10BAA9: TBTree::~TBTree() (btree.hpp:116)
==10805==    by 0x10E15B: main (main.cpp:78)
==10805==  If you believe this happened as a result of a stack
==10805==  overflow in your program's main thread (unlikely but
==10805==  possible),you can try to increase the size of the
==10805==  main thread stack using the --main-stacksize= flag.
==10805==  The main thread stack size used in this run was 8388608.
==10805==
```

```
==10805== HEAP SUMMARY:
==10805==     in use at exit: 74,752 bytes in 3 blocks
==10805==   total heap usage: 3 allocs,0 frees,74,752 bytes allocated
==10805==
==10805== LEAK SUMMARY:
==10805==    definitely lost: 0 bytes in 0 blocks
==10805==    indirectly lost: 0 bytes in 0 blocks
==10805==      possibly lost: 0 bytes in 0 blocks
==10805==    still reachable: 74,752 bytes in 3 blocks
==10805==         suppressed: 0 bytes in 0 blocks
==10805== Reachable blocks (those to which a pointer was found) are not shown.
==10805== To see them,rerun with: --leak-check=full --show-leak-kinds=all
==10805==
==10805== For lists of detected and suppressed errors,rerun with: -s
==10805== ERROR SUMMARY: 1 errors from 1 contexts (suppressed: 0 from 0)
Ошибка сегментирования (образ памяти сброшен на диск)
```

Утилита показала, что проблема возникает при удалении пустого дерева. Если первой командой программе поступает удаление или поиск, То память не выделяется и в результате деструктор пытается удалить пустое дерево. Для решения данной проблемы нужно добавить проверку на пустоту дерева.

```
alex@wega:~/Рабочий стол/вуз/2course/da_labs/Lab_03$ valgrind --tool=memcheck
--leak-check=full ./lab2
==11795== Memcheck,a memory error detector
==11795== Copyright (C) 2002-2017,and GNU GPL'd,by Julian Seward et al.
==11795== Using Valgrind-3.18.1 and LibVEX; rerun with -h for copyright info
==11795== Command: ./lab2
==11795==
djfndjf
NoSuchWord
-dsjfnjdsf
NoSuchWord
==11795==
==11795== HEAP SUMMARY:
==11795==     in use at exit: 0 bytes in 0 blocks
==11795==   total heap usage: 3 allocs,3 frees,74,752 bytes allocated
==11795==
==11795== All heap blocks were freed --no leaks are possible
==11795==
==11795== For lists of detected and suppressed errors,rerun with: -s
==11795== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

После исправления ошибки я смог сдать программу в тестирующую систему. Также можно заметить, что Valgrind вывел меньше информации после исправления.

2 GPROF

Согласно [2], gprof — это профайлер GNU, измеряющий производительность программы. Он измеряет производительность программ, написанных на Fortran, C++, ассемблере и C. Результаты, генерируемые командой Linux, помогают оптимизировать код для более быстрого выполнения и повышения эффективности за счет отображения частей программы, которые занимают больше всего времени выполнения.

Тестирование проводилось на тесте из $5 * 10^6$ строк.

```
alex@wega:~/Рабочий стол/вуз/2course/da_labs/Lab_03$ cat tests/01.t | gprof
./lab2 >profile
Flat profile:

Each sample counts as 0.01 seconds.
%                self     total
time      calls  ms/call  ms/call  name
11.15   3334454     0.00     0.00  TBTree::TNode::Search(TBTree::TNode*,std::string
key)
6.98   1665546     0.00     0.00  TBTree::DeleteFromTNode(TBTree::TNode*,std::string
key)
6.89  1697919116     0.00     0.00  std::string::_M_data() const
4.90  679259395     0.00     0.00  std::string::size() const
4.08     21461     0.02     0.02  void std::__detail::__to_chars_10_impl<unsigned
long>(char*,unsigned int,unsigned long)
3.63   5000000     0.00     0.00  ToLower(std::string)
3.26  164102639     0.00     0.00  std::string::compare(std::string const&)
const
3.08  356251961     0.00     0.00  std::string::_M_set_length(unsigned long)
2.54  402402493     0.00     0.00  std::string::_M_is_local() const
2.45                               _init
2.36  609359178     0.00     0.00  std::char_traits<char>::assign(char&,char
const&)
2.27  252482141     0.00     0.00  std::string::push_back(char)
2.04  75590382     0.00     0.00  std::string::_M_create(unsigned long&,unsigned
long)
1.72  360523096     0.00     0.00  std::string::_M_length(unsigned long)
1.36  514964282     0.00     0.00  __gnu_cxx::__normal_iterator<char*,std::string
>::base() const
1.36  257482141     0.00     0.00  bool __gnu_cxx::operator==<char*,std::string
>()
1.31  174101515     0.00     0.00  unsigned long const& std::min<unsigned long>(unsig
```

```
long const&,unsigned long const&)
1.27  402402493     0.00     0.00  std::pointer_traits<char const*>::pointer_to(char
const&)
1.22  174483356     0.00     0.00  std::char_traits<char>::compare(char const*,char
const*,unsigned long)
1.18  402402493     0.00     0.00  std::string::_M_local_data() const
1.18  252482141     0.00     0.00  __gnu_cxx::__normal_iterator<char*,std::string
>::operator++()
1.13  402402493     0.00     0.00  char const* std::__addressof<char const>(char
const&)
1.13  84132652     0.00     0.00  std::string::_M_data(char*)
1.04  287311487     0.00     0.00  std::string::capacity() const
1.04  99779403     0.00     0.00  std::allocator_traits<std::allocator<char>>::max_siz
1.00  75590382     0.00     0.00  std::string::_M_destroy(unsigned long)
1.00     21461     0.01     0.01  std::string::basic_string(std::string&&)
0.95  64242366     0.00     0.00  std::string::basic_string(std::string const&)
0.91  402402493     0.00     0.00  char const* std::addressof<char const>(char
const&)
0.91  99779403     0.00     0.00  std::string::max_size() const
0.82  246289919     0.00     0.00  std::is_constant_evaluated()
0.82                               std::__size_to_integer(unsigned long)
0.77  252482141     0.00     0.00  std::string::operator+=(char)
0.77  73351635     0.00     0.00  std::string::_Alloc_hider::_Alloc_hider(char*,std::a
0.77  64242366     0.00     0.00  std::string::_S_copy_chars(char*,char*,char*)
0.73  75590382     0.00     0.00  std::allocator_traits<std::allocator<char>>::dealloc
long)
0.68  164866321     0.00     0.00  std::string::data() const
0.68  164102639     0.00     0.00  decltype ((__char_traits_cmp_cat<std::char_traits<c
std::operator<=><char,std::char_traits<char>,std::allocator<char>>(std::string
const&,std::string const&)
0.68  100819871     0.00     0.00  std::string::_M_dispose()
0.59  75590382     0.00     0.00  __gnu_cxx::new_allocator<char>::deallocate(char*,uns
long)
0.59  64242366     0.00     0.00  void std::string::_M_construct<char*>(char*,char*,st
0.59                               std::char_traits<char>::lt(char const&,char
const&)
0.54  100087694     0.00     0.00  std::char_traits<char>::copy(char*,char
const*,unsigned long)
0.54   1646861     0.00     0.00  TBTree::InsertNonfull(TBTree::TNode*,TElem)
0.54     21461     0.00     0.00  std::string::_S_assign(char*,unsigned long,char)
0.50  75590382     0.00     0.00  __gnu_cxx::new_allocator<char>::allocate(unsigned
```

```
long,void const*)
0.45  252482141     0.00     0.00  __gnu_cxx::__normal_iterator<char*,std::string
>::operator*() const
0.45  80018300     0.00     0.00  std::string::_Alloc_hider::~_Alloc_hider()
0.45    719309     0.00     0.00  TBTree::Deallocate(TBTree::TNode*)
0.41  161202225     0.00     0.00  std::string::_M_get_allocator()
0.41  64242366     0.00     0.00  __gnu_cxx::__alloc_traits<std::allocator<char>,char
0.41  10000000     0.00     0.00  __gnu_cxx::__normal_iterator<char*,std::string
>::__normal_iterator(char* const&)
0.36  80039761     0.00     0.00  std::string::_M_local_data()
0.36  80039761     0.00     0.00  char* std::addressof<char>(char&)
0.36  80018300     0.00     0.00  std::string::~basic_string()
0.36  75590382     0.00     0.00  std::allocator_traits<std::allocator<char>>::alloca
long)
0.36  64242366     0.00     0.00  bool __gnu_cxx::__is_null_pointer<char>(char*)
0.36  64242366     0.00     0.00  std::allocator_traits<std::allocator<char>>::select_
0.36   9769981     0.00     0.00  std::string::_M_mutate(unsigned long,unsigned
long,char const*,unsigned long)
0.36   6645204     0.00     0.00  std::string::basic_string<std::allocator<char>>(char
const*,std::allocator<char>const&)
0.32  164102639     0.00     0.00  auto std::__detail::__char_traits_cmp_cat<std::char
0.32  124916982     0.00     0.00  std::operator>(std::strong_ordering,std::__cmp_cat
0.32  114395066     0.00     0.00  std::string::length() const
0.32  100712770     0.00     0.00  std::string::_S_copy(char*,char const*,unsigned
long)
0.32  84111191     0.00     0.00  std::string::_M_capacity(unsigned long)
0.32   5775933     0.00     0.00  std::string::_M_check_length(unsigned long,unsigned
long,char const*) const
0.27   6666665     0.00     0.00  std::string::_Alloc_hider::_Alloc_hider(char*,std::a
0.27   6645204     0.00     0.00  void std::string::_M_construct<char const*>(char
const*,char const*,std::forward_iterator_tag)
0.27                             main
0.23  80039761     0.00     0.00  std::pointer_traits<char*>::pointer_to(char&)
0.23  78373096     0.00     0.00  std::remove_reference<std::allocator<char>&>::type&&
std::move<std::allocator<char>&>(std::allocator<char>&)
0.23  39185657     0.00     0.00  std::operator<(std::strong_ordering,std::__cmp_cat:
0.23  12225261     0.00     0.00  __gnu_cxx::__alloc_traits<std::allocator<char>,char
0.23   9998876     0.00     0.00  std::string::compare(char const*) const
0.23   5754472     0.00     0.00  std::string::operator=(char const*)
0.23   5000000     0.00     0.00  std::string::begin()
0.23     21461     0.00     0.00  std::remove_reference<std::string&>::type&&
```

```
std::move<std::string&>(std::string&)
0.23     21461     0.00     0.01  std::string std::operator+<char,std::char_traits<cha
const*,std::string&&)
0.18  64242366     0.00     0.00  void std::string::_M_construct_aux<char*>(char*,char
0.18  14806959     0.00     0.00  TElem::TElem(TElem const&)
0.18   5000000     0.00     0.00  std::string::operator=(std::string&&)
0.18     21461     0.00     0.00  std::string::basic_string<std::allocator<char>>(unsi
long,char,std::allocator<char>const&)
0.14  64242366     0.00     0.00  std::iterator_traits<char*>::difference_type
std::distance<char*>(char*,char*)
0.14  12225261     0.00     0.00  std::string::_M_assign(std::string const&)
0.14   7422261     0.00     0.00  TElem::TElem()
0.14   5754472     0.00     0.00  std::string::_M_disjunct(char const*) const
0.14     21461     0.00     0.00  std::string::_M_limit(unsigned long,unsigned
long) const
0.14     21461     0.00     0.00  std::string::insert(unsigned long,char const*)
0.14                             void std::__fill_a<char*,char>(char*,char*,char
const&)
0.09  164102639     0.00     0.00  std::__cmp_cat::__unspec::__unspec(std::__cmp_cat:
0.09  164021769     0.00     0.00  std::string::_M_get_allocator() const
0.09  80039761     0.00     0.00  char* std::__addressof<char>(char&)
0.09  64242366     0.00     0.00  void std::string::_M_construct<char*>(char*,char*)
0.09  64242366     0.00     0.00  std::iterator_traits<char*>::difference_type
std::__distance<char*>(char*,char*,std::random_access_iterator_tag)
0.09  22229220     0.00     0.00  TElem::~TElem()
0.09  12225261     0.00     0.00  TElem::operator=(TElem const&)
0.09  12225261     0.00     0.00  std::string::assign(std::string const&)
0.09  12225261     0.00     0.00  std::string::operator=(std::string const&)
0.09   8749706     0.00     0.00  __gnu_cxx::__alloc_traits<std::allocator<char>,char>
0.09   5775933     0.00     0.00  std::string::_M_replace(unsigned long,unsigned
long,char const*,unsigned long)
0.09   5754472     0.00     0.00  std::less<char const*>::operator()(char const*,char
const*) const
0.09   4645542     0.00     0.00  std::string::_S_compare(unsigned long,unsigned
long)
0.09   1646861     0.00     0.00  TBTree::Insert(TElem)
0.09    750326     0.00     0.00  std::string* std::__addressof<std::string
>(std::string&)
0.09    719309     0.00     0.00  TBTree::AllocateTNode()
0.09     21461     0.00     0.00  std::string::_M_construct(unsigned long,char)
0.09         1    10.00    55.93  TBTree::DeleteTree(TBTree::TNode*)
```

```
0.09                                std::string::c_str() const
0.09                                std::char_traits<char>::move(char*,char const*,unsig
long)
0.05  75590382     0.00     0.00  __gnu_cxx::new_allocator<char>::_M_max_size()
const
0.05   6645204     0.00     0.00  bool __gnu_cxx::__is_null_pointer<char const>(char
const*)
0.05   6645204     0.00     0.00  std::string::_S_copy_chars(char*,char const*,char
const*)
0.05   5000000     0.00     0.00  std::string::clear()
0.05                                char* std::__copy_move<false,false,std::random_acces
const*,char*>(char const*,char const*,char*)
0.05                                std::string::_S_move(char*,char const*,unsigned
long)
0.05                                char* std::__copy_move_a2<false,char const*,char*>(c
const*,char const*,char*)
0.00  64242366     0.00     0.00  std::iterator_traits<char*>::iterator_category
std::__iterator_category<char*>(char* const&)
0.00  31784780     0.00     0.00  __gnu_cxx::__enable_if<std::__is_char<char>::__value
std::operator==<char>(std::string const&,std::string const&)
0.00  22420013     0.00     0.00  std::char_traits<char>::length(char const*)
0.00   9998876     0.00     0.00  bool std::operator==<char,std::char_traits<char>,std
const&,char const*)
0.00   9087808     0.00     0.00  std::string::basic_string()
0.00   8749706     0.00     0.00  __gnu_cxx::__alloc_traits<std::allocator<char>,char>
0.00   6645204     0.00     0.00  std::iterator_traits<char const*>::difference_type
std::__distance<char const*>(char const*,char const*,std::random_access_iterator_tag)
0.00   6645204     0.00     0.00  std::iterator_traits<char const*>::iterator_category
std::__iterator_category<char const*>(char const* const&)
0.00   6645204     0.00     0.00  std::iterator_traits<char const*>::difference_type
std::distance<char const*>(char const*,char const*)
0.00   5754472     0.00     0.00  std::string::assign(char const*)
0.00   5000000     0.00     0.00  std::string::end()
0.00   5000000     0.00     0.00  void std::__alloc_on_move<std::allocator<char>>(std
0.00   1667789     0.00     0.00  TBTree::Search(std::string)
0.00   1666665     0.00     0.00  TBTree::SWV(std::string)
0.00   1665546     0.00     0.00  TBTree::Delete(std::string)
0.00    549808     0.00     0.00  TBTree::SplitChild(TBTree::TNode*,int,TBTree::TNode*
0.00    206643     0.00     0.00  TBTree::MergeTNodes(TBTree::TNode*,TBTree::TNode*,TB
0.00     21461     0.00     0.00  std::string::_M_check(unsigned long,char
const*) const
```

```
0.00     21461     0.00     0.00  std::char_traits<char>::assign(char*,unsigned
long,char)
0.00     21461     0.00     0.00  std::string::replace(unsigned long,unsigned
long,char const*,unsigned long)
0.00     21461     0.00     0.00  std::string::operator[](unsigned long)
0.00     21461     0.00     0.03  std::__cxx11::to_string(unsigned long)
0.00     21461     0.00     0.00  unsigned int std::__detail::__to_chars_len<unsigned
long>(unsigned long,int)
0.00      1013     0.00     0.00  TBTree::TNode::FindSuccessor(TBTree::TNode*)
0.00       211     0.00     0.00  TBTree::TNode::FindPredecessor(TBTree::TNode*)
0.00         1     0.00     0.00  __static_initialization_and_destruction_0(int,int)
0.00         1     0.00     0.00  TBTree::TBTree()
0.00         1     0.00    55.93  TBTree::~TBTree()
```

Исходя из вывода утилиты, можем сделать вывод, что программа тратит большую часть времени на операции поиска и удаления. Я считаю, что это связано с тем, что в процессе поиска элементов, программа делает много сравнений и копирований строк. Чтобы оптимизировать работу программы нужно меня способ работы с парой ключ-значение и, возможно, использовать более эффективный контейнер, например, вектор.

3 Дневник отладки

1. 09.05.2024 Ознакомился с утилитой Valgrind и использовал ее для поиска утечек памяти и ошибок, связанных с памятью. Найденную ошибку устранил.

2. 09.05.2024 Ознакомился с утилитой Gprof и провел профилирование программы с ее помощью.

4 Выводы

Выполнив третью лабораторную работу по курсу «Дискретный анализ», я ознакомился с утилитами для профилирования программ и применил их на практике.

С помощью Valgrind я смог найти ошибку при работе с памятью, после исправления которой тестирующая система приняла код.

С помошью утилиты Gprof я узал, какая функция тратит больше времени во время работы программы. А точнее на поиск и удаление.

Обе утилиты достаточно просты в использовании и помогают обнаружить функции, которые можно оптимизировать, тем самым улучшить программу, чтобы она работала быстрее и использовала меньше памяти.

Список литературы

[1] *Valgrind — Википедия.*
URL: `https://ru.wikipedia.org/wiki/Valgrind` (дата обращения: 09.05.2024).

[2] *Как использовать команду Gprof Linux.*
URL: `https://ru.linux-console.net/?p=14238` (дата обращения: 09.05.2024).