Московский Авиационный Институт

(Национальный Исследовательский Университет)

Факультет информационных технологий и прикладной математики

Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу

«Дискретный анализ»

Профилирование

Студент: Эсмедляев Федор Романович
Группа: М8О–212Б–22
Преподаватель: Н.Д.Глушин
Оценка:____
Дата:_____
Подпись:_____

Москва, 2024.

Условие

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).

Метод решения

Leask — инструментальное программное обеспечение, предназначенное для отладки использования памяти, обнаружения утечек памяти, а также профилирования. Мною была использована данная утилита для поиска возможных утечек памяти на достаточно обширном тесте. Результат ее работы можно увидеть ниже:

1)Результат работы на маленьком тесте
main(9805) MallocStackLogging: could not tag MSL-related memory as no_footprint, so those pages will be included in process footprint - (null)
main(9805) MallocStackLogging: recording malloc and VM allocation stacks using lite mode
+ a 1
OK
+ b 2
OK
+ c 3
OK
+ d 4
OK
- b
OK
Process 9805 is not debuggable. Due to security restrictions, leaks can only

show or save contents of readonly memory of restricted processes.

Process: main [9805]
Path: /Users/USER/*/main
Load Address: 0x102860000
Identifier: main
Version: 0
Code Type: ARM64
Platform: macOS
Parent Process: leaks [9804]

Date/Time: 2024-04-22 21:28:06.884 +0300
Launch Time: 2024-04-22 21:27:53.025 +0300
OS Version: macOS 13.2.1 (22D68)
Report Version: 7
Analysis Tool: /usr/bin/leaks

Physical footprint: 2945K
Physical footprint (peak): 2945K
Idle exit: untracked
---

leaks Report Version: 4.0, multi-line stacks
Process 9805: 225 nodes malloced for 21 KB
Process 9805: 0 leaks for 0 total leaked bytes.

Как мы видим 0 утечек

2) На тесте размером 1e5

Process 9982 is not debuggable. Due to security restrictions, leaks can only show or save contents of readonly memory of restricted processes.

Process: main [9982]
Path: /Users/USER/*/main
Load Address: 0x1003bc000
Identifier: main
Version: 0
Code Type: ARM64
Platform: macOS
Parent Process: leaks [9981]

Date/Time: 2024-04-22 21:29:00.488 +0300
Launch Time: 2024-04-22 21:28:59.910 +0300
OS Version: macOS 13.2.1 (22D68)
Report Version: 7
Analysis Tool: /usr/bin/leaks

Physical footprint: 3265K
Physical footprint (peak): 3265K
Idle exit: untracked
---

leaks Report Version: 4.0, multi-line stacks
Process 9982: 225 nodes malloced for 21 KB
Process 9982: 0 leaks for 0 total leaked bytes.

Так же 0 утечек

Gprof — утилита для профилирования программ, с помощью которой можно замерить время выполнения как всей программы, так и определённых функций и методов (ниже приведен простой профиль):

Из предоставленных данных можно сделать следующие выводы:
1)Функция main занимает 100% времени выполнения программы.
2)Самая вызываемая функция из main - это lower, вызываемая 139,721 раз, и она занимает 100% времени выполнения main.
3)Большая часть времени в lower тратится на вызов функции __gnu_cxx::__normal_iterator<char*, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > >::base(), которая вызывается 737,984 раза и занимает 50% времени выполнения программы.
4)Другие значительные функции, вызываемые из main, включают std::pair<std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >, unsigned long long>::operator=, std::make_pair, std::operator|, и т. д.
5)__gnu_cxx::__normal_iterator<char*, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > >::base() const и unsigned long long&& std::forward<unsigned long long>(std::remove_reference<unsigned long long>::type&) занимают по 50% времени выполнения программы. Это указывает на то, что операции связанные с итерированием по строкам (__normal_iterator) и передачей параметров (в этом случае беззнакового длинного целого числа) занимают значительную часть времени выполнения программы.
6)Остальные функции занимают намного меньшую долю времени, но могут все ещё вносить вклад в общую производительность программы. Например, функции связанные с операциями над строками (std::operator==, std::operator<, std::operator!=) и операциями над парами (std::pair) занимают незначительное количество времени, но могут быть оптимизированы для улучшения производительности.
7)Функции, связанные с деревом RB_tree (RB_tree::rotateRight, RB_tree::rotateLeft, RB_tree::fix_delete и т.д.), также занимают небольшую долю времени, что указывает на то, что эффективность работы этой структуры данных может быть улучшена.

Вот небольшая вставка вывода gprof

```
Flat profile:

Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  ns/call  ns/call  name
 50.00      0.01     0.01   737984     6.78     6.78
__gnu_cxx::__normal_iterator<char*, std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > >::base() const
 50.00      0.01     0.01   100184    49.91    49.91  unsigned long long&&
std::forward<unsigned long long>(std::remove_reference<unsigned long
long>::type&)
  0.00      0.01     0.00   368992     0.00    13.55  bool
__gnu_cxx::operator!=<char*, std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >
>(__gnu_cxx::__normal_iterator<char*, std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > > const&,
__gnu_cxx::__normal_iterator<char*, std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > > const&)
  0.00      0.01     0.00   229271     0.00     0.00
__gnu_cxx::__normal_iterator<char*, std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > >::operator++()
  0.00      0.01     0.00   229271     0.00     0.00
__gnu_cxx::__normal_iterator<char*, std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > >::operator*() const
  0.00      0.01     0.00   139721     0.00    35.79
lower(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >&)
  0.00      0.01     0.00   120052     0.00     0.00  bool
std::operator==<char, std::char_traits<char>, std::allocator<char>
>(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > const&, char const*)
  0.00      0.01     0.00   120008     0.00     0.00
std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >& std::forward<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char>
>&>(std::remove_reference<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >&>::type&)
  0.00      0.01     0.00    99477     0.00     0.00
std::pair<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >, unsigned long long>::~pair()
  0.00      0.01     0.00    72014     0.00     0.00  bool std::operator<
<char, std::char_traits<char>, std::allocator<char>
>(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > const&, std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > const&)
  0.00      0.01     0.00    60004     0.00     0.00
RB_tree::seek(std::pair<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >, unsigned long long> const&,
Node*)
  0.00      0.01     0.00    60004     0.00     0.00
RB_tree::seeker(std::pair<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >, unsigned long long> const&)
  0.00      0.01     0.00    40180     0.00    49.91
std::pair<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >, unsigned long
long>::pair<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >&, unsigned long long,
```

```
true>(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >&, unsigned long long&&)
  0.00      0.01     0.00    40180     0.00    99.82
std::pair<std::__strip_reference_wrapper<std::decay<std::__cxx11::basic_stri
ng<char, std::char_traits<char>, std::allocator<char> >&>::type>::__type,
std::__strip_reference_wrapper<std::decay<unsigned long
long>::type>::__type> std::make_pair<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >&, unsigned long
long>(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >&, unsigned long long&&)
  0.00      0.01     0.00    40165     0.00     0.00
std::operator|(std::_Ios_Openmode, std::_Ios_Openmode)
  0.00      0.01     0.00    39648     0.00     0.00  unsigned long long&
std::forward<unsigned long long&>(std::remove_reference<unsigned long
long&>::type&)
  0.00      0.01     0.00    39477     0.00     0.00
std::pair<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >, unsigned long
long>::pair<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >, unsigned long long, true>()
  0.00      0.01     0.00    22096     0.00     0.00  bool
std::operator><char, std::char_traits<char>, std::allocator<char>
>(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> > const&, std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > const&)
  0.00      0.01     0.00    20338     0.00     0.00
RB_tree::clearHelper(Node*)
  0.00      0.01     0.00    20338     0.00     0.00
RB_tree::Load(std::istream&, Node*)
  0.00      0.01     0.00    20338     0.00     0.00  RB_tree::clear()
  0.00      0.01     0.00    19827     0.00     0.00
RB_tree::Save(std::ostream&, Node*)
  0.00      0.01     0.00    19824     0.00     0.00
std::pair<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >, unsigned long
long>::pair<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >&, unsigned long long&,
true>(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >&, unsigned long long&)
  0.00      0.01     0.00    19824     0.00    49.91
std::pair<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >, unsigned long
long>::operator=(std::pair<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >, unsigned long long>&&)
  0.00      0.01     0.00    19824     0.00     0.00
std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >&& std::forward<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >
>(std::remove_reference<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> > >::type&)
  0.00      0.01     0.00    19824     0.00     0.00
std::pair<std::__strip_reference_wrapper<std::decay<std::__cxx11::basic_stri
ng<char, std::char_traits<char>, std::allocator<char> >&>::type>::__type,
std::__strip_reference_wrapper<std::decay<unsigned long
long&>::type>::__type> std::make_pair<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >&, unsigned long
long&>(std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >&, unsigned long long&)
```

```
  0.00      0.01     0.00    19672     0.00     0.00
std::pair<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >, unsigned long
long>::operator=(std::pair<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >, unsigned long long> const&)
  0.00      0.01     0.00    19652     0.00     0.00
Node::Node(std::pair<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >, unsigned long long> const&)
  0.00      0.01     0.00    19652     0.00     0.00  RB_tree::fix(Node*)
  0.00      0.01     0.00    19652     0.00     0.00
RB_tree::insert(std::pair<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >, unsigned long long> const&)
  0.00      0.01     0.00    19648     0.00     0.00  Node::~Node()
  0.00      0.01     0.00     9580     0.00     0.00  bool
std::operator><std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >, unsigned long
long>(std::pair<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >, unsigned long long> const&,
std::pair<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >, unsigned long long> const&)
  0.00      0.01     0.00     9580     0.00     0.00  bool std::operator<
<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >, unsigned long
long>(std::pair<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >, unsigned long long> const&,
std::pair<std::__cxx11::basic_string<char, std::char_traits<char>,
std::allocator<char> >, unsigned long long> const&)
  0.00      0.01     0.00     1494     0.00     0.00
RB_tree::rotateRight(Node*)
  0.00      0.01     0.00     1483     0.00     0.00
RB_tree::rotateLeft(Node*)
  0.00      0.01     0.00      170     0.00     0.00
RB_tree::fix_delete(Node*)
  0.00      0.01     0.00      170     0.00     0.00
RB_tree::del(std::pair<std::__cxx11::basic_string<char,
std::char_traits<char>, std::allocator<char> >, unsigned long long> const&)
  0.00      0.01     0.00      170     0.00     0.00  RB_tree::erase(Node*)
  0.00      0.01     0.00        3     0.00     0.00
std::chrono::duration<long, std::ratio<1l, 1000000000l> >::count() const
  0.00      0.01     0.00        2     0.00     0.00
std::chrono::time_point<std::chrono::_V2::system_clock,
std::chrono::duration<long, std::ratio<1l, 1000000000l> >
>::time_since_epoch() const
  0.00      0.01     0.00        1     0.00     0.00
__static_initialization_and_destruction_0(int, int)
  0.00      0.01     0.00        1     0.00     0.00
std::chrono::duration<long, std::ratio<1l, 1000l> >::count() const
  0.00      0.01     0.00        1     0.00     0.00
std::enable_if<std::chrono::__is_duration<std::chrono::duration<long,
std::ratio<1l, 1000l> > >::value, std::chrono::duration<long, std::ratio<1l,
1000l> > >::type std::chrono::duration_cast<std::chrono::duration<long,
std::ratio<1l, 1000l> >, long, std::ratio<1l, 1000000000l>
>(std::chrono::duration<long, std::ratio<1l, 1000000000l> > const&)
  0.00      0.01     0.00        1     0.00     0.00
std::chrono::duration<long, std::ratio<1l, 1000l> >
std::chrono::__duration_cast_impl<std::chrono::duration<long, std::ratio<1l,
1000l> >, std::ratio<1l, 1000000l>, long, true, false>::__cast<long,
```

```
std::ratio<1l, 1000000000l> >(std::chrono::duration<long, std::ratio<1l,
1000000000l> > const&)
  0.00      0.01     0.00        1     0.00     0.00
std::chrono::duration<long, std::ratio<1l, 1000000000l> >::duration<long,
void>(long const&)
  0.00      0.01     0.00        1     0.00     0.00
std::chrono::duration<long, std::ratio<1l, 1000l> >::duration<long,
void>(long const&)
  0.00      0.01     0.00        1     0.00     0.00
std::common_type<std::chrono::duration<long, std::ratio<1l, 1000000000l> >,
std::chrono::duration<long, std::ratio<1l, 1000000000l> > >::type
std::chrono::operator-<std::chrono::_V2::system_clock,
std::chrono::duration<long, std::ratio<1l, 1000000000l> >,
std::chrono::duration<long, std::ratio<1l, 1000000000l> >
>(std::chrono::time_point<std::chrono::_V2::system_clock,
std::chrono::duration<long, std::ratio<1l, 1000000000l> > > const&,
std::chrono::time_point<std::chrono::_V2::system_clock,
std::chrono::duration<long, std::ratio<1l, 1000000000l> > > const&)
  0.00      0.01     0.00        1     0.00     0.00
std::common_type<std::chrono::duration<long, std::ratio<1l, 1000000000l> >,
std::chrono::duration<long, std::ratio<1l, 1000000000l> > >::type
std::chrono::operator-<long, std::ratio<1l, 1000000000l>, long,
std::ratio<1l, 1000000000l> >(std::chrono::duration<long, std::ratio<1l,
1000000000l> > const&, std::chrono::duration<long, std::ratio<1l,
1000000000l> > const&)

 %         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.
```

Gcov – утилита проверки покрытия кода

1) File 'main.cpp':

Lines executed: 90.21% of 286

Данный файл содержит исходный код вашей программы (вероятно, это главный файл main.cpp).

Из общего количества строк в файле (286 строк) исполнено 90.21%. Это означает, что 90.21% строк в файле были выполнены во время работы программы.

2) File '/usr/include/c++/11/...:

Далее идут отчёты о покрытии для файлов, включенных в вашу программу из стандартной библиотеки C++ (например, iostream, chrono, basic_string, и т.д.).

Эти файлы содержат шаблонные классы и функции, которые могут быть использованы вашей программой.

Каждый отчёт включает:

3) Lines executed: процент выполненных строк кода из общего количества строк в файле.

После этого файлы с покрытием менее чем на 100% отмечаются как созданные (например, 'basic_string.h.gcov'). Это происходит потому, что не все строки кода в этих файлах были выполнены.

Lines executed: 79.13% of 393:

Это общий отчёт о покрытии для всех файлов, включенных в анализ. Из 393 строк, которые были выполнены в процессе выполнения программы, 79.13% строки были исполнены.

Итак, общая картина покрывает выполнение кода ваших исходных файлов, а также включенных файлов из стандартной библиотеки C++. Общий процент выполнения кода ваших файлов составляет 90.21%, что означает, что большинство кода было исполнено в процессе работы программы.

Вот сам вывод **gcov**

```
fedorubuntu@fedorubuntu-ZenBook-UX325JA-UX325JA:~/Документы/oop/lab1$ gcov main.cpp

 File 'main.cpp'

Lines executed:90.21% of 286

Creating 'main.cpp.gcov'

File '/usr/include/c++/11/iostream'

No executable lines

Removing 'iostream.gcov'

File '/usr/include/c++/11/bits/stl_iterator_base_funcs.h'
```

```
 Lines executed:0.00% of 5

Creating 'stl_iterator_base_funcs.h.gcov'

File '/usr/include/c++/11/bits/stl_iterator_base_types.h'

Lines executed:0.00% of 2

Creating 'stl_iterator_base_types.h.gcov'

File '/usr/include/c++/11/ext/type_traits.h'

Lines executed:0.00% of 2

Creating 'type_traits.h.gcov'

File '/usr/include/c++/11/bits/basic_string.tcc'

Lines executed:0.00% of 13

Creating 'basic_string.tcc.gcov'

File '/usr/include/c++/11/bits/alloc_traits.h'

Lines executed:0.00% of 2

Creating 'alloc_traits.h.gcov'

File '/usr/include/c++/11/bits/basic_string.h'

Lines executed:35.29% of 17

Creating 'basic_string.h.gcov'

File '/usr/include/c++/11/ext/new_allocator.h'

Lines executed:0.00% of 3

Creating 'new_allocator.h.gcov'

File '/usr/include/c++/11/ext/alloc_traits.h'

Lines executed:0.00% of 2

Creating 'alloc_traits.h.gcov'

File '/usr/include/c++/11/chrono'

Lines executed:100.00% of 15

Creating 'chrono.gcov'

File '/usr/include/c++/11/bits/move.h'

Lines executed:50.00% of 4

Creating 'move.h.gcov'

File '/usr/include/c++/11/bits/stl_pair.h'

Lines executed:100.00% of 19

Creating 'stl_pair.h.gcov'
```

```
File '/usr/include/c++/11/bits/stl_iterator.h'

Lines executed:100.00% of 9

Creating 'stl_iterator.h.gcov'

File '/usr/include/c++/11/bits/char_traits.h'

Lines executed:0.00% of 12

Creating 'char_traits.h.gcov'

File '/usr/include/c++/11/bits/ios_base.h'

Lines executed:100.00% of 2

Creating 'ios_base.h.gcov'

Lines executed:79.13% of 393

fedorubuntu@fedorubuntu-ZenBook-UX325JA-UX325JA:~/Документы/oop/lab1$
```

Вывод

При выполнении данной лабораторной работе мною были изучены и опробованы различные инструменты профилирования кода, направленные на поиск утечек памяти (Leask), анализирующие время выполнения всякой функции для поиска слабых мест в программе с точки зрения скорости выполнения(gprof), а также оценивать покрытие кода тестами и выводить информацию об этом в удобном формате(gcov). Данные утилиты очень полезны при отладке и написании своей программы, могут открыть глаза на узкие места кода в тех случаях, когда его объем почти не поддается “бумажному” анализу.