Лабораторная работа №3 по курсу дискретного анализа: Исследование качества программ

Выполнил студент группы М8О-208Б-20 Фаттяхетдинов Сильвестр Динарович

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.

gprof

Основная информация

Утилита gprof позволяет измерить время работы всех функций, методов и операторов программы, количество их вызовов и долю от общего времени работы программы в процентах.

Команды для работы с утилитой

Шаг 1) скомпилировать программу с флагом *–pg*:

```
g++ main.cpp –pg –o prog
```

Шаг 2) запустить программу, передав ей на ввод файл *test.txt*, в котором содержится по 5000 команд на вставку, поиск и удаление, вывод сохранить в out.txt:

```
prog.exe <test.txt >out.txt
```

Кроме файла *out.txt*, в котором содержатся результаты выполнения команд из *test.txt*, также в каталоге появился файл *gmon.out*, в котором содержится вся информация, предоставленная утилитой gprof.

```
Шаг 3) получить всю информацию в текстовом виде
gprof prog.exe gmon.out > profile–data.txt
```

Таким образом, получен текстовый файл с подробной информацией о времени работы и вызовах всех функций и операторов, которыеиспользовались в программе.

Результат работы утилиты

Ниже приведена таблица, в которую перенесены данные из файла *profile data.txt*, полученного с помощью утилиты gprof.

```
Each sample counts as 0.01 seconds.
  %   cumulative   self              self     total
 time   seconds   seconds    calls  us/call  us/call  name
 38.21      7.55     7.55                             _mcount_private
 19.38     11.38     3.83                             __fentry__
  5.57     12.48     1.10  1000001     1.10     1.22  Patricia::Search(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >&)
  5.52     13.57     1.09 765531350     0.00     0.00  __gnu_cxx::__normal_iterator<char*, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > >::operator++()
  4.76     14.51     0.94  3000001     0.31     1.18  __gnu_cxx::__normal_iterator<char*, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > > std::transform<__
  4.55     15.41     0.90  1000000     0.90     1.62  Patricia::Add(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >&, unsigned long long const&)
  4.30     16.26     0.85 167581977     0.01     0.01  GetIndex(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >&, int)
  4.30     17.11     0.85  1000000     0.85     1.02  Patricia::Delete(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >&)
  3.19     17.74     0.63 385765676     0.00     0.00  bool __gnu_cxx::operator!=<char*, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > >(__gnu_cxx::__norma
  2.68     18.27     0.53 765531350     0.00     0.00  __gnu_cxx::__normal_iterator<char*, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > >::operator*() con
  1.82     18.63     0.36 771531352     0.00     0.00  __gnu_cxx::__normal_iterator<char*, std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> > >::base() const
  1.16     18.86     0.23                             std::basic_istream<char, std::char_traits<char> >& std::operator>><char, std::char_traits<char>, std::allocator<char> >(std::basic_istr
  0.71     19.00     0.14                             std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >::size() const
  0.56     19.11     0.11                             std::istreambuf_iterator<char, std::char_traits<char> > std::num_get<char, std::istreambuf_iterator<char, std::char_traits<char> > >::_
  0.46     19.20     0.09                             main
  0.40     19.28     0.08 34535379     0.00     0.00  unsigned long long const& std::max<unsigned long long>(unsigned long long const&, unsigned long long const&)
  0.35     19.35     0.07                             std::istream::sentry::sentry(std::istream&, bool)
  0.35     19.42     0.07                             std::basic_ostream<char, std::char_traits<char> >& std::__ostream_insert<char, std::char_traits<char> >(std::basic_ostream<char, std::c
  0.28     19.48     0.06   992261     0.06     0.49  FirstDifferentBit(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >&, std::__cxx11::basic_string<char, st
  0.20     19.52     0.04                             std::basic_filebuf<char, std::char_traits<char> >::xsputn(char const*, long long)
  0.15     19.55     0.03                             std::ostreambuf_iterator<char, std::char_traits<char> > std::num_put<char, std::ostreambuf_iterator<char, std::char_traits<char> > >::_
  0.15     19.57     0.03                             std::basic_streambuf<char, std::char_traits<char> >::xsputn(char const*, long long)
  0.10     19.59     0.02   992262     0.02     0.02  Node::Node()
  0.10     19.61     0.02                             std::ostream::operator<<(unsigned long long)
  0.10     19.64     0.02                             std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >::_M_append(char const*, unsigned long long)
  0.05     19.65     0.01                             std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >::data() const
  0.05     19.66     0.01                             std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >::compare(char const*) const
  0.05     19.66     0.01                             std::codecvt<char, char, int>::do_always_noconv() const
  0.05     19.68     0.01                             std::istream::operator>>(unsigned long long&)
  0.05     19.68     0.01                             std::basic_filebuf<char, std::char_traits<char> >::underflow()
  0.05     19.70     0.01                             std::locale::~locale()
  0.05     19.70     0.01                             std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >::begin()
  0.05     19.72     0.01                             std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >::operator=(char const*)
  0.05     19.73     0.01                             std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >::operator=(std::__cxx11::basic_string<char, std::char_t
  0.05     19.73     0.01                             std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >::operator[](unsigned long long)
  0.05     19.75     0.01                             std::ctype<char> const& std::use_facet<std::ctype<char> >(std::locale const&)
  0.05     19.75     0.01                             std::basic_ostream<char, std::char_traits<char> >& std::operator<< <std::char_traits<char> >(std::basic_ostream<char, std::char_traits<
  0.03     19.76     0.01                             GetAmountOfLeadingZeroes(std::__cxx11::basic_string<char, std::char_traits<char>, std::allocator<char> >&)
```

Все остальные функции по данным gprof работали 0% времени, поэтому на скриншоте они отсутствуют. _mcount_private, __fentry__ - в программе отсутствуют, запускаются при работе утилиты gprof, их можно опустить. Больше всего времени заняла функция поиска элемента, за ней идут инкремент и std::transform, также много времени задали Add, Delete и функция работы со строками GetIndex. Меньше времени заняли различные операторы работы со строками, а также со вводом-выводом.

valgrind

Valgrind является самым распространённым и одним из самых удобных инструментов для отслеживания утечек памяти и других подобных ошибок. Для проверки программы на ошибки, связанные с памятью, необходимо ввести следующую команду в терминал:

```
valgrind ./main <test.txt >out.txt
```

В результате выполнения этой команды получаем следующее сообщение:

==13594== Memcheck, a memory error detector

==13594== Copyright (C) 2002-2017, and GNU GPL'd, by Julian Seward et al.

==13594== Using Valgrind-3.15.0 and LibVEX; rerun with -h for copyright info

==13594== Command: ./main

==13594==

==13594==

==13594== HEAP SUMMARY:

==13594== in use at exit: 122,880 bytes in 6 blocks

==13594== total heap usage: 10,900 allocs, 10,894 frees, 1,457,413 bytes allocated

==13594==

==13594== LEAK SUMMARY:

==13594== definitely lost: 0 bytes in 0 blocks

==13594== indirectly lost: 0 bytes in 0 blocks

==13594== possibly lost: 0 bytes in 0 blocks

==13594== still reachable: 122,880 bytes in 6 blocks

==13594== suppressed: 0 bytes in 0 blocks

==13594== Rerun with --leak-check=full to see details of leaked memory

==13594==

==13594== For lists of detected and suppressed errors, rerun with: -s

==13594== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

Утечек обнаружено не было. Остались лишь still reachable из-за следующих строк:
std::ios::sync_with_stdio(false);

std::cin.tie(nullptr);

std::cout.tie(nullptr);

Но, так как они ускоряют программу и всё же не вызывают утечек, они были оставлены в программе.

Выводы

Выполнив данную лабораторную работу, я узнал о существовании такой утилиты как gprof и научился её использовать, также вспомнил valgrind и закрепил навыки работы с ним.