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

Выполнил студент группы М08-312Б МАИ *Лобанов Олег*.

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).

Описание

gprof

Утилита gprof позволяет измерить время работы всех функций, методов и операторов программы, количество их вызовов и долю от общего времени работы программы в проценах. Для использования необходимо скомпилировать программу с кючом $-pg$:

g++ laba2.cpp $-pg$ $-o$ laba2

Затем запустим программу, передав ей на ввод тест *test*, в котором содержатся по 1000 команд на вставку, поиск и удаление:

./laba2 <test >result.txt

После выполнения данной команды появляется также файл *gmon.out*, в котором содержится информация, предоставленная для утилиты gprof. Теперь выполним следующую команду и получим следующие данные в текстовом файле:

$gprof$ laba2 gmon.out > $gprof.txt$

```
Flat profile:

Each sample counts as 0.01 seconds.
%   cumulative   self              self     total
time   seconds   seconds    calls  ns/call  ns/call  name
40.02    0.10     0.10                             main
20.00    0.17     0.05     1000     4.33     4.33  Insert
20.01    0.22     0.05     1000     4.73     4.73  DeleteNode
20.00    0.25     0.05     1000     4.54     4.54  FindNode
0.00     0.25     0.00      508     0.00     0.00  Rotate_L
0.00     0.25     0.00      485     0.00     0.00  Rotate_R
0.00     0.25     0.00        1     0.00     0.00  Clean
```

Valgrind

Valgrind является самым распространенным инструментом для отслаживания утечек памяти и других ошибок, связанных с памятью. Для проверки программы laba2 на проблемы с памятью нужно выполнить следующую команду:

valgrind $--leak-check=full$ $--show-leak-kinds=all$ ./laba2 <test >valgrind.txt

где

- *./laba2* - запуск программы
- *test* - текстовый файл с набором тестов
- *valgrind.txt* - текстовый файл, в который записывается вывод **valgrind**
- $--leak-check=full$ - ключ для детального анализа утечек памяти
- $--show-leak-kinds=all$ - ключ для отображения всех выдов утечек памяти (**Definite, Possible, Indirect**)

Дневник отладки

После выполнения команды получаем следующий вывод:

==176542== Memcheck, a memory error detector

==176542== Copyright (C) 2002-2022, and GNU GPL'd, by Julian Seward et al.

==176542== Using Valgrind-3.20.0 and LibVEX; rerun with -h for copyright info

==176542== Command: ./laba2

==176542== Parent PID: 142788

==176542==

==176542== HEAP SUMMARY:

==176542== total heap usage: 2,012 allocs, 2,004 frees, 572,658 bytes allocated

==176542==

==176542== LEAK SUMMARY:

==176542== definitely lost: 0 bytes in 0 blocks

==176542== indirectly lost: 0 bytes in 0 blocks

==176542== possibly lost: 0 bytes in 0 blocks

==176542== still reachable: 0 bytes in 0 blocks

==176542== suppressed: 0 bytes in 0 blocks

==176542==

==176542== For lists of detected and suppressed errors, rerun with: -s

==176542== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)

С помощью **valgrind** обнаружили неосвобожденную память, но это не является критической проблемой, так как она не является "потерянной". После чего исправил код и неосвобожденная память освободилась.

Выводы

Выполнив третью лабораторную работу я познакомился с такой утилитой как **valgrind**, а также обнаружил не освобожденную память. Я научился грамотнее отлаживать свои программы при помощи современной утилиты. Также научился использовать утилиту **gprof** для измерения времени работы программы.