Московский авиационный институт (национальный исследовательский университет)

Факультет прикладной математики и физики

Кафедра вычислительной математики и программирования

Лабораторная работа №3 по курсу «Дискретный анализ»

Студентка: А. Довженко
Преподаватель: Д. Е. Ильвохин
Группа: 08-207

Дата:
Оценка:
Подпись:

Москва, 2017

Лабораторная работа №3

Для реализации словаря из предыдущей лабораторной работы необходимо провести исследование скорости выполнения и потребления оперативной памяти. В случае выявления ошибок или явных недочетов требуется их исправить.

Используемые утилиты: gprof, valgrind, dmalloc, perf.

1 Описание программы

Разработка программы разбита на несколько этапов, каждый из которых символизирует уменьшение количества ошибок и рост качества реализации Патриции в программе. Сама по себе программа представляет собой реализацию Патриции, ее интерфейсную "обертку"для внешнего кода, реализующую АТД словаря, и участок кода, ответственный за взаимодействие с пользователем.

2 Дневник отладки

В первой версии программы не было ничего, кроме реализации дерева с его стандартными функциями поиска, вставки и удаления. В этой версии использовалась библиотека dmalloc, и после компиляции программы, которая ее использовала, была получена череда ошибок:

```
1214542872: 9: starting time = 1214542872
1214542872: 9: process pid = 25835
1214542872: 9: error details: finding address in heap
1214542872: 9: pointer '0x7F2FBB'from 'TPatricia.c:93'prev access 'unknown'
1214542872: 9: ERROR: free: tried to free previously freed pointer (err 61)
...
1214542872: 25: ERROR: free: tried to free previously freed pointer (err 61)
1214542872: 26: ending time = 1214542872,elapsed since start = 0:00:00
```

Получаем несколько double free error при удалении из дерева. Ошибка возникла из-за противоречий в функциях удаления узлов и удаления дерева.

Исправили удаление, тестируем программу valgrind'ом с ключом –leak-check=full, который включает функцию обнаружения утечек памяти.

```
==2301== HEAP SUMMARY:
==2301==     in use at exit: 1,191,797 bytes in 100,197 blocks
==2301==   total heap usage: 201,872 allocs,101,675 frees,5,841,568 bytes allocated
==2301==
==2301== LEAK SUMMARY:
==2301==    definitely lost: 1,110,011 bytes in 100,001 blocks
==2301==    indirectly lost: 0 bytes in 0 blocks
==2301==      possibly lost: 2,064 bytes in 1 blocks
==2301==    still reachable: 4,096 bytes in 1 blocks
```

```
==2301==          suppressed: 0 bytes in 0 blocks
==2301==
```

Как видим, повторного освобождения памяти уже не происходит, но обнаружена серьезная утечка. Причиной такой утечки оказалось отсутствие освобождения переменной, в которой хранился ключ, место для которого выделялось динамически.

Существовал ряд ошибок, связанных с использованием библиотечных функций <string.h>.

```
==24589== Invalid read of size 1
==24589==    at 0x4C31F93: strcmp (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux
==24589==    by 0x4015D9: KeysIsEqual (in /home/karma/mai_study/DA/lab2/da2)
==24589==    by 0x40102B: NodeSearch (in /home/karma/mai_study/DA/lab2/da2)
==24589==    by 0x401615: DictSearch (in /home/karma/mai_study/DA/lab2/da2)
==24589==    by 0x40226D: main (in /home/karma/mai_study/DA/lab2/da2)
==24589==  Address 0x5204b00 is 0 bytes inside a block of size 4 free'd
==24589==    at 0x4C2EDEB: free (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux.s
==24589==    by 0x40157A: TreeDelete (in /home/karma/mai_study/DA/lab2/da2)
==24589==    by 0x401736: DictRemove (in /home/karma/mai_study/DA/lab2/da2)
==24589==    by 0x4020F2: main (in /home/karma/mai_study/DA/lab2/da2)
==24589==  Block was alloc'd at
==24589==    at 0x4C2DB8F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux
==24589==    by 0x401671: DictAdd (in /home/karma/mai_study/DA/lab2/da2)
==24589==    by 0x40207C: main (in /home/karma/mai_study/DA/lab2/da2)

==25832== Invalid write of size 1
==25832==    at 0x4C3106F: strcpy (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux
==25832==    by 0x401301: TreeDelete (in /home/karma/mai_study/DA/da2/da2)
==25832==    by 0x40170F: DictRemove (in /home/karma/mai_study/DA/da2/da2)
==25832==    by 0x4020BC: main (in /home/karma/mai_study/DA/da2/da2)
==25832==  Address 0x5204542 is 0 bytes after a block of size 2 alloc'd
==25832==    at 0x4C2DB8F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux
==25832==    by 0x40164A: DictAdd (in /home/karma/mai_study/DA/da2/da2)
==25832==    by 0x402046: main (in /home/karma/mai_study/DA/da2/da2)
```

Обе ошибки допущены по невнимательности: в первом случае ошибка возникала, когда аргумент функции strcmp принимал значение NULL, во втором случае – когда копируемая строка была по размеру больше той, куда она копировалась. Были добавлены функции-обертки KeysIsEqual и KeysCopy.

После исправления этих ошибок еще раз запускаем программу с valgrind, используя флаг -v. Получаем ошибки в функции GetBit.

```
==31800== HEAP SUMMARY:
==31800==     in use at exit: 12,356 bytes in 1 blocks
==31800==   total heap usage: 1,922 allocs,1,921 frees,68,148 bytes allocated
==31800==
==31800== Searching for pointers to 1 not-freed blocks
==31800== Checked 78,792 bytes
==31800==
==31800== LEAK SUMMARY:
==31800==    definitely lost: 0 bytes in 0 blocks
==31800==    indirectly lost: 0 bytes in 0 blocks
==31800==      possibly lost: 0 bytes in 0 blocks
==31800==    still reachable: 12,356 bytes in 1 blocks
==31800==         suppressed: 0 bytes in 0 blocks
==31800== Rerun with --leak-check=full to see details of leaked memory
==31800==
==31800== ERROR SUMMARY: 7 errors from 1 contexts (suppressed: 0 from 0)
==31800==
==31800== 7 errors in context 1 of 1:
==31800== Invalid read of size 1
==31800==    at 0x400E57: GetBit (in /home/karma/mai_study/DA/da2/da2)
==31800==    by 0x40125C: TreeInsert (in /home/karma/mai_study/DA/da2/da2)
==31800==    by 0x4019D9: DictAdd (in /home/karma/mai_study/DA/da2/da2)
==31800==    by 0x402443: main (in /home/karma/mai_study/DA/da2/da2)
==31800==  Address 0x520e452 is 0 bytes after a block of size 2 alloc'd
==31800==    at 0x4C2DB8F: malloc (in /usr/lib/valgrind/vgpreload_memcheck-amd64-linux
==31800==    by 0x401985: DictAdd (in /home/karma/mai_study/DA/da2/da2)
==31800==    by 0x402443: main (in /home/karma/mai_study/DA/da2/da2)
==31800==
==31800== ERROR SUMMARY: 7 errors from 1 contexts (suppressed: 0 from 0)
```

Ошибки возникли по причине того, что индекс, передаваемый в функцию как аргумент, допускал в некоторых случаях выход за пределы массива. Была добавлена проверка такого случая, после чего valgrind показывал следующее:

```
==31838== HEAP SUMMARY:
==31838==     in use at exit: 0 bytes in 0 blocks
==31838==   total heap usage: 1,922 allocs,1,922 frees,68,244 bytes allocated
==31838==
==31838== All heap blocks were freed --no leaks are possible
==31838==
==31838== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
==31838== ERROR SUMMARY: 0 errors from 0 contexts (suppressed: 0 from 0)
```

Посмотрим плоский профиль производительности программы, который предлагает gprof:

```
Flat profile:

Each sample counts as 0.01 seconds.
%   cumulative   self              self     total
time   seconds   seconds    calls  ms/call  ms/call  name
27.54      0.11     0.11   500000     0.00     0.00  NodeSearch
22.53      0.20     0.09   302944     0.00     0.00  TreeInsert
20.03      0.28     0.08       12     6.68     6.68  DictWriteRec
8.76       0.32     0.04 14733585     0.00     0.00  GetBit
7.51       0.35     0.03       12     2.50     6.45  DictReadRec
5.01       0.37     0.02    92793     0.00     0.00  TreeDelete
5.01       0.39     0.02       13     1.54     1.54  NodeDestroy
2.50       0.40     0.01  1101280     0.00     0.00  NodeCreate
1.25       0.40     0.01   798336     0.00     0.00  NodeFill
0.00       0.40     0.00    40432     0.00     0.00  KeysCopy
0.00       0.40     0.00       12     0.00     6.45  DictRead
0.00       0.40     0.00       12     0.00     0.00  DictWrite
0.00       0.40     0.00       12     0.00     0.00  TreeCreate
0.00       0.40     0.00       12     0.00     1.67  TreeDestroy
```

После изменения де(сериализации), сложность которой должна была уменьшиться в $\lg(n)$ раз:

```
Flat profile:

Each sample counts as 0.01 seconds.
%   cumulative   self              self     total
time   seconds   seconds    calls  ms/call  ms/call  name
52.86      0.19     0.19   500000     0.00     0.00  NodeSearch
16.69      0.25     0.06       12     5.01     5.84  DictWriteLinear
8.35       0.28     0.03 21801234     0.00     0.00  GetBit
8.35       0.31     0.03       13     2.31     2.31  NodeDestroy
5.56       0.33     0.02   798336     0.00     0.00  NodeFill
2.78       0.34     0.01   302944     0.00     0.00  TreeInsert
2.78       0.35     0.01       12     0.83     0.83  GetArray
2.78       0.36     0.01       12     0.83     2.50  SetArray
0.00       0.36     0.00  1101280     0.00     0.00  NodeCreate
0.00       0.36     0.00    92793     0.00     0.00  TreeDelete
```

```
0.00      0.36     0.00    40432     0.00     0.00  KeysCopy
0.00      0.36     0.00       12     0.00     2.50  DictRead
0.00      0.36     0.00       12     0.00     2.50  DictReadLinear
0.00      0.36     0.00       12     0.00     0.00  DictWrite
0.00      0.36     0.00       12     0.00     0.00  TreeCreate
0.00      0.36     0.00       12     0.00     2.50  TreeDestroy
```

Оба профиля составлены при поддержке тестового файла, содержащего все возможные операции со словарем. Получили общий выигрыш во времени, хотя отдельные функции стали использоваться чаще. Функция поиска используется так часто, потому что при любой вставке и удалении из дерева, мы сначала проверяем есть ли запрашиваемый ключ в дереве. Ничего неожиданного здесь нет: подавляющая часть времени ушла на поиск, вставку, удаление и запись в файл.

С помощью утилиты perf посмотрим производительность программы.

```
17,09%  da2      libc-2.23.so       [.] _IO_vfscanf
14,62%  da2      da2                [.] GetBit
10,96%  da2      da2                [.] NodeSearch
5,70%  da2      libc-2.23.so       [.] vfprintf
5,19%  da2      da2                [.] TreeInsert
4,14%  da2      libc-2.23.so       [.] _int_malloc
3,42%  da2      libc-2.23.so       [.] __GI_____strtoull_l_internal
3,35%  da2      libc-2.23.so       [.] __GI_____strtoll_l_internal
2,99%  da2      libc-2.23.so       [.] strlen
2,70%  da2      libc-2.23.so       [.] __strcmp_sse2_unaligned
2,68%  da2      da2                [.] NodeDestroy
2,37%  da2      libc-2.23.so       [.] malloc_consolidate
2,13%  da2      da2                [.] DictWriteLinear
1,94%  da2      libc-2.23.so       [.] _IO_file_xsputn@@GLIBC_2.2.5
1,53%  da2      libc-2.23.so       [.] free
1,39%  da2      libc-2.23.so       [.] _itoa_word
1,32%  da2      da2                [.] GetArray
1,29%  da2      libc-2.23.so       [.] _IO_sputbackc
1,17%  da2      libc-2.23.so       [.] __GI___mempcpy
1,05%  da2      [kernel.kallsyms]  [k] finish_task_switch
1,05%  da2      libc-2.23.so       [.] malloc
1,03%  da2      libc-2.23.so       [.] __isoc99_fscanf
1,01%  da2      da2                [.] TreeDelete
0,79%  da2      libc-2.23.so       [.] _int_free
0,53%  da2      da2                [.] WordToLower
0,50%  da2      da2                [.] NodeFill
```

```
0,50%  da2      libc-2.23.so       [.] __strchrnul
0,48%  da2      [kernel.kallsyms]  [k] __softirqentry_text_start
0,45%  da2      da2                [.] NodeCreate
0,43%  da2      libc-2.23.so       [.] tolower
0,41%  da2      [kernel.kallsyms]  [k] __lock_text_start
0,41%  da2      da2                [.] SetArray
0,36%  da2      libc-2.23.so       [.] __strcpy_sse2_unaligned
0,31%  da2      da2                [.] ReadLine
0,24%  da2      da2                [.] KeysIsEqual
0,24%  da2      da2                [.] strlen@plt
0,24%  da2      libc-2.23.so       [.] _IO_getc
0,22%  da2      [kernel.kallsyms]  [k] copy_user_generic_string
0,14%  da2      [kernel.kallsyms]  [k] __do_page_fault
0,14%  da2      [kernel.kallsyms]  [k] get_page_from_freelist
0,12%  da2      [kernel.kallsyms]  [k] kmem_cache_alloc
0,12%  da2      da2                [.] DictReadLinear
0,12%  da2      da2                [.] main
0,10%  da2      [kernel.kallsyms]  [k] find_get_entry
0,10%  da2      libc-2.23.so       [.] __strtoll_internal
0,07%  da2      [kernel.kallsyms]  [k] __add_to_page_cache_locked
0,07%  da2      [kernel.kallsyms]  [k] clear_page
0,07%  da2      [kernel.kallsyms]  [k] ext4_do_update_inode
0,07%  da2      [kernel.kallsyms]  [k] fsnotify
0,07%  da2      [kernel.kallsyms]  [k] kmalloc_slab
```

Видим, что первые четыре функции имеют наибольшую нагрузку. В принципе, предположения о таком результате можно было сделать уже после использования gprof. Функции, которые чаще других используются и занимают большую часть времени исполнения программы, потребляют больше ресурсов.

3 Выводы

valgrind, dmalloc, gprof – простые и мощные инструменты, позволяющие существенно ускорить профилирование и отладку программ.
Очевидный толк был в первых двух, однако gprof ничего нового мне не поведал, поскольку я имею достаточное представление об устройстве программы: она вполне себе типовая и маленькая. gprof оказался бы полезнее, если бы разрабатывалась большая программу со сложными взаимосвязями. Кстати, профилирование можно осуществлять и с помощью valgrind'a. perf, по сравнению с другими тремя средствами диагностики, показался мне намного сложнее. Трудно оценить все его возможности в рамках данной работы. Существует и множество других инструментов анализа производительности, их выбор для своей работы, скорее, дело привычки.

Исследование качества программы и ее оптимизация требуют особого внимания, особенно для высоконагруженных сложных систем, где любая на первый взгяд незначительная ошибка или медленный модуль могут существенно повлиять на конечный продукт. Каждый программист должен писать эффективный код, насколько это возможно.