

Московский Авиационный Институт
(Национальный Исследовательский Университет)
Институт № 8 информационных технологий и прикладной математики
Кафедра вычислительной математики и программирования

Лабораторная работа №4 по курсу
«Дискретный анализ»

Исследование программы

Студент: Пермяков Никита Александрович

Группа: М80 – 208Б-19

Вариант: 1

Преподаватель: *Кохтеве Антон Алексеевич*

Оценка: _____

Дата: _____

Подпись: _____

Москва, 2020

Содержание

1. Дневник отладки
2. Выводы
3. Исправления
4. Общие выводы

Дневник отладки

тесты выполнял из файла test.txt

Объектный файл приложения не подходит для динамических библиотек. Так как не знают адреса памяти в которые будет загружена использующая их программа. Несколько различных программ могут использовать одну библиотеку, и каждая из них располагается в различном адресном пространстве. Поэтому требуется, чтобы переходы в функциях библиотеки (операции goto на ассемблере) использовали не абсолютную адресацию, а относительную. То есть генерируемый компилятором код должен быть независимым от адресов, такая технология получила название PIC - Position Independent Code. В компиляторе gcc данная возможность включается ключом -fPIC.

gprof

g++

-pg

Сгенерировать дополнительный код для записи информации профилирования, подходящей для программы анализа gprof. Эта опция используется при компиляции исходных файлов, о которых поступают данные, а также при связывании.

-o

Указание названий объектных файлов, получаемых при компиляции

-Wall

Включает множество флагов предупреждений компилятора, в частности (-Waddress, -Wcomment, -Wformat, -Wbool-compare, -Wuninitialized, -Wunknown-pragmas, -Wunused-value, -Wunused-value...)

-L

Каталог с библиотеками указывается ключом

-l

Используемые библиотеки перечисляются через ключ,

```
$ g++ -Wall -pg -o profile main.cpp
```

```
$ ./profile
```

```
+ irojgr 43
```

```
+ i 43
```

```
+ b,obomvIIMC9V 5
```

```
+ t-4l4kg0 7
```

```
- irojgr
```

```
! S res.txt
```

```
i
```

```
irojgr
```

```
?
```

```
OK
```

```
OK
```

```
OK
```

```
OK
```

```
OK
```

```
OK
```

```
OK: 43
```

```
NoSuchWord
```

```
$ gprof profile > profile.output
```

dmalloc.h

`eval` – интерфейс POSIX. примет аргумент и создаст из него команду, которая будет выполнена оболочкой.

`command` – утилита указывающая выполнить команду

`dmalloc`

`-d <число>` опция, указывающая уровень подробности проверок

`-p` алиас для запуска

`-l <файл>` указание лога для записи

подключил в `cpp`

```
#ifndef DMALLOC
#include "dmalloc.h"
#endif
```

установка

скачать архив и распаковать

`./configure`

`make`

`make install`

либо

`$sudo apt install libdmalloc-dev`

`$eval `command dmalloc -b -l logfile -i 100 low``

`dmalloc` - может быть скомпонована с приложением статически или связана динамически при помощи `LD_PRELOAD`

проверить переменную окружения

`$echo $DMALLOC_OPTIONS`

Полный список лексем вместе с кратким объяснением и соответствующим каждой лексеме числовым значением можно получить с помощью

`dmalloc -DV`

компиляция статически - почему-то не удалась

```
g++ -Wall -DDMALLOC -DDMALLOC_FUNC_CHECK -I/usr/local/include -  
L/usr/local/lib main.cpp -ldmalloc
```

выполнение динамически

```
LD_PRELOAD=libdmalloc.so ./a.out
```

logfile

```
1606889229: 83: Dmalloc version '5.5.2' from 'http://dmalloc.com/'
```

```
1606889229: 83: flags = 0x4e48503, logfile 'logfile'
```

```
1606889229: 83: interval = 100, addr = 0, seen # = 0, limit = 0
```

```
1606889229: 83: threads enabled, lock-on = 0, lock-init = 2
```

```
1606889229: 83: starting time = 1606889218
```

```
1606889229: 83: process pid = 3025
```

libdmalloc выводит данные о

выделении памяти, количестве вызовов конкретных функций

```
1606889229: 83: Dumping Chunk Statistics:
```

```
1606889229: 83: basic-block 4096 bytes, alignment 8 bytes
```

```
1606889229: 83: heap address range: 0x7f7ced230000 to 0x7f7ced3f1000,  
1839104 bytes
```

```
1606889229: 83:   user blocks: 61 blocks, 224256 bytes (73%)
```

```
1606889229: 83:   admin blocks: 14 blocks, 57344 bytes (18%)
```

```
1606889229: 83:   total blocks: 75 blocks, 307200 bytes
```

```
1606889229: 83: heap checked 1
```

```

1606889229: 83: alloc calls: malloc 45, calloc 0, realloc 0, free 38
1606889229: 83: alloc calls: realloc 0, memalign 0, posix_memalign 0, valloc 0
1606889229: 83: alloc calls: new 0, delete 0
1606889229: 83:  current memory in use: 195584 bytes (7 pnts)
1606889229: 83:  total memory allocated: 204640 bytes (45 pnts)
1606889229: 83:  max in use at one time: 204429 bytes (20 pnts)
1606889229: 83: max allocated with 1 call: 72704 bytes
1606889229: 83: max unused memory space: 29891 bytes (12%)
1606889229: 83: top 10 allocations:
1606889229: 83: total-size  count in-use-size  count source
1606889229: 83:          0    0          0    0 Total of 0

```

в программе имеются ошибки, приводящие к утечкам памяти

```

1606889229: 83: Dumping Not-Freed Pointers Changed Since Start:
1606889229: 83: not freed: '0x7f7ced340008|s1' (32768 bytes) from 'unknown'
1606889229: 83: not freed: '0x7f7ced350008|s1' (32768 bytes) from 'unknown'
1606889229: 83: not freed: '0x7f7ced360008|s1' (32768 bytes) from 'unknown'
1606889229: 83: not freed: '0x7f7ced370008|s1' (8192 bytes) from 'unknown'
1606889229: 83: not freed: '0x7f7ced380008|s1' (8192 bytes) from 'unknown'
1606889229: 83: not freed: '0x7f7ced3a0008|s1' (8192 bytes) from 'unknown'
1606889229: 83: not freed: '0x7f7ced3b0008|s1' (72704 bytes) from 'unknown'
1606889229: 83: total-size  count source
1606889229: 83:          0    0 Total of 0
1606889229: 83: ending time = 1606889229, elapsed since start = 0:00:11

```

Выводы о найденных недочетах

gprof

Разобрал отчет, увидел наиболее часто вызываемые функции.

Топ 6

| time | seconds | seconds | calls | Ts/call | Ts/call | name |
|------|---------|---------|-------|---------|---------|---------------------------------------|
| 0.00 | 0.00 | 0.00 | 170 | 0.00 | 0.00 | TVector::At(int) const |
| 0.00 | 0.00 | 0.00 | 104 | 0.00 | 0.00 | TVector::At(int) |
| 0.00 | 0.00 | 0.00 | 104 | 0.00 | 0.00 | TVector::operator[](int) |
| 0.00 | 0.00 | 0.00 | 66 | 0.00 | 0.00 | TVector::operator[](int) const |
| 0.00 | 0.00 | 0.00 | 64 | 0.00 | 0.00 | char* std::__niter_base<char*>(char*) |
| 0.00 | 0.00 | 0.00 | 61 | 0.00 | 0.00 | TVector::Size() const |

Построил на бумаге граф вызовов

| index | % time | self | children | called | name |
|-------|--------|------|----------|--------|-------------------------------------|
| | 0.00 | 0.00 | 66/170 | | TVector::operator[](int) const [11] |
| | 0.00 | 0.00 | 104/170 | | TVector::At(int) [9] |
| [8] | 0.0 | 0.00 | 0.00 | 170 | TVector::At(int) const [8] |
| ----- | | | | | |
| | 0.00 | 0.00 | 104/104 | | TVector::operator[](int) [10] |
| [9] | 0.0 | 0.00 | 0.00 | 104 | TVector::At(int) [9] |
| | 0.00 | 0.00 | 104/170 | | TVector::At(int) const [8] |
| ----- | | | | | |
| | 0.00 | 0.00 | 1/104 | | TDetailAvl::SaveLoad() [55] |
| | 0.00 | 0.00 | 21/104 | | main [6] |
| | 0.00 | 0.00 | 82/104 | | TDetailAvl::Lower(TVector&) [37] |
| [10] | 0.0 | 0.00 | 0.00 | 104 | TVector::operator[](int) [10] |
| | 0.00 | 0.00 | 104/104 | | TVector::At(int) [9] |
| ----- | | | | | |

| | | | | |
|------|------|------|--------|---|
| | 0.00 | 0.00 | 66/66 | operator<(TVector const&, TVector const&) |
| [31] | | | | |
| [11] | 0.0 | 0.00 | 0.00 | 66 |
| | | | | TVector::operator[](int) const [11] |
| | 0.00 | 0.00 | 66/170 | TVector::At(int) const [8] |



Вывод:

Максимальная связанность у **At(int)** и **At(int) const**

Максимальная степень вершины – **[]int** (оптимизировать нечего)

решение:

Удалить функции **At(int)** и **At(int) const**, вызывая их на уровень выше в **[]int** и **[]int const**

dmalloc

Поменял типы, с подсчетом потребляемой памяти:

с -- unit64_t – на -- unit32_t

с -- int – на – uint64_t (в перегрузках [])

Свернул две функции в одну PrintTree с вызовом из public

Сократил число вызовов функции Size();

Сравнение исправленной программы с предыдущей версией

gprof

| time | seconds | seconds | calls | Ts/call | Ts/call | name |
|-------|---------|---------|-------|---------|---------|---|
| 0.00 | 0.00 | 0.00 | 106 | 0.00 | 0.00 | TVector::operator[](unsigned long) |
| 0.00 | 0.00 | 0.00 | 66 | 0.00 | 0.00 | TVector::operator[](unsigned long) |
| const | | | | | | |
| 0.00 | 0.00 | 0.00 | 64 | 0.00 | 0.00 | char* std::__niter_base<char*>(char*) |
| 0.00 | 0.00 | 0.00 | 43 | 0.00 | 0.00 | TVector::Size() const |
| 0.00 | 0.00 | 0.00 | 36 | 0.00 | 0.00 | TVector::~~TVector() |
| 0.00 | 0.00 | 0.00 | 32 | 0.00 | 0.00 | char* std::__copy_move<false, true, std::random_access_iterator_tag>::__copy_m<char>(char const*, char const*, char*) |

Изменения

Число вызовов

TVector::Size() const уменьшилось 61 на 42

TVector::~~TVector() уменьшилось с 38 на 36

Остальные вызовы на 3-6 раз

Граф вызовов изменился – стало меньше простых циклов. Максимальная степень вершины уменьшилась на 2.

dmalloc

1606917466: 83: Dmalloc version '5.5.2' from 'http://dmalloc.com/'
1606917466: 83: flags = 0x4e48503, logfile 'logfile'
1606917466: 83: interval = 100, addr = 0, seen # = 0, limit = 0
1606917466: 83: threads enabled, lock-on = 0, lock-init = 2
1606917466: 83: starting time = 1606917460
1606917466: 83: process pid = 1237
1606917466: 83: Dumping Chunk Statistics:
1606917466: 83: basic-block 4096 bytes, alignment 8 bytes
1606917466: 83: heap address range: 0x7fb977c90000 to 0x7fb9784c1000,
8589312 bytes
1606917466: 83: user blocks: 61 blocks, 224256 bytes (75%)
1606917466: 83: admin blocks: 12 blocks, 49152 bytes (16%)
1606917466: 83: total blocks: 73 blocks, 299008 bytes
1606917466: 83: heap checked 1
1606917466: 83: alloc calls: malloc 45, calloc 0, realloc 0, free 38
1606917466: 83: alloc calls: realloc 0, memalign 0, posix_memalign 0, valloc 0
1606917466: 83: alloc calls: new 0, delete 0
1606917466: 83: current memory in use: 195584 bytes (7 pnts)
1606917466: 83: total memory allocated: 204672 bytes (45 pnts)
1606917466: 83: max in use at one time: 204453 bytes (20 pnts)
1606917466: 83: max alloced with 1 call: 72704 bytes
1606917466: 83: max unused memory space: 30059 bytes (12%)
1606917466: 83: top 10 allocations:
1606917466: 83: total-size count in-use-size count source
1606917466: 83: 0 0 0 0 Total of 0
1606917466: 83: Dumping Not-Freed Pointers Changed Since Start:
1606917466: 83: not freed: '0x7fb977d80008|s1' (32768 bytes) from 'unknown'

1606917466: 83: not freed: '0x7fb977d90008|s1' (32768 bytes) from 'unknown'
1606917466: 83: not freed: '0x7fb977da0008|s1' (32768 bytes) from 'unknown'
1606917466: 83: not freed: '0x7fb977db0008|s1' (8192 bytes) from 'unknown'
1606917466: 83: not freed: '0x7fb977dc0008|s1' (8192 bytes) from 'unknown'
1606917466: 83: not freed: '0x7fb977dd0008|s1' (8192 bytes) from 'unknown'
1606917466: 83: not freed: '0x7fb977de0008|s1' (72704 bytes) from 'unknown'
1606917466: 83: total-size count source
1606917466: 83: 0 0 Total of 0
1606917466: 83: ending time = 1606917466, elapsed since start = 0:00:06

Изменения

elapsed since start = 0:00:11

elapsed since start = 0:00:06

total blocks: 75 blocks, 307200 bytes

total blocks: 73 blocks, 299008 bytes

total memory allocated: 204640 bytes (45 pnts)

total memory allocated: 204672 bytes (45 pnts)

Общие выводы

Оптимизация C/C++ кода обеспечивается через **выравнивание**

```
struct A {  
    char a;  
    long int b;  
    int c;  
};
```

24 байта

узнаем через sizeof(A)

```
struct B {  
    char a;  
    int c;  
    long int b;  
};
```

16 байт.

Списки инициализации

A(T id): size(1), parent(id){ }

не так:

```
B(T id) {  
    size = 1;  
    parent = id;  
}
```

При использовании списков инициализации происходил вызов конструктора с переданными значениями, в отличие от второго случая, когда вызывается конструктор по умолчанию для полей класса и в теле конструкторе класса производится присваивание.

Префиксный или постфиксный оператор

inline функции не добавляет зависимости (*.h файлов) при компиляции.

Вызов встроенных функций (– не использовал)

Все остальные техники были задействованы и показали свой результат комплексно.