

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: *Кухтичев 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

-1

Используемые библиотеки перечисляются через ключ,

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

```
$ ./profile
```

```
$ 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 сpp

```
#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

```
1607180440: 19881: Dmalloc version '5.5.2' from 'http://dmalloc.com/'
```

```
1607180440: 19881: flags = 0x4e48503, logfile 'logfile'
```

```
1607180440: 19881: interval = 100, addr = 0, seen # = 0, limit = 0
```

```
1607180440: 19881: threads enabled, lock-on = 0, lock-init = 2
```

```
1607180440: 19881: starting time = 1607180428
```

```
1607180440: 19881: process pid = 22
```

libdmalloc выводит данные о

выделении памяти, количестве вызовов конкретных функций

1607180440: 19881: Dumping Chunk Statistics:

1607180440: 19881: basic-block 4096 bytes, alignment 8 bytes

1607180440: 19881: heap address range: 0x7f8a600e0000 to 0x7f8a60571000, 4788224 bytes

1607180440: 19881: user blocks: 83 blocks, 314368 bytes (65%)

1607180440: 19881: admin blocks: 35 blocks, 143360 bytes (29%)

1607180440: 19881: total blocks: 118 blocks, 483328 bytes

1607180440: 19881: heap checked 199

1607180440: 19881: alloc calls: malloc 9944, calloc 0, realloc 0, free 9937

1607180440: 19881: alloc calls: realloc 0, memalign 0, posix\_memalign 0, valloc 0

1607180440: 19881: alloc calls: new 0, delete 0

1607180440: 19881: current memory in use: 195584 bytes (7 pnts)

1607180440: 19881: total memory allocated: 306790 bytes (9944 pnts)

1607180440: 19881: max in use at one time: 236180 bytes (1319 pnts)

1607180440: 19881: max allocated with 1 call: 72704 bytes

1607180440: 19881: max unused memory space: 89468 bytes (27%)

1607180440: 19881: top 10 allocations:

1607180440: 19881: total-size count in-use-size count source

1607180440: 19881:	63348	5848	63348	5848	Other pointers
1607180440: 19881:	63348	5848	63348	5848	Total of 0

в программе имеются ошибки, приводящие к утечкам памяти

1607180440: 19881: Dumping Not-Freed Pointers Changed Since Start:

1607180440: 19881: not freed: '0x7f8a604c0008|s1' (32768 bytes) from 'unknown'

1607180440: 19881: not freed: '0x7f8a604d0008|s1' (32768 bytes) from 'unknown'

1607180440: 19881: not freed: '0x7f8a604e0008|s1' (32768 bytes) from 'unknown'

1607180440: 19881: not freed: '0x7f8a604f0008|s1' (8192 bytes) from 'unknown'

```

1607180440: 19881: not freed: '0x7f8a60500008|s1' (8192 bytes) from 'unknown'
1607180440: 19881: not freed: '0x7f8a60520008|s1' (8192 bytes) from 'unknown'
1607180440: 19881: not freed: '0x7f8a60530008|s1' (72704 bytes) from
'unknown'
1607180440: 19881: total-size count source
1607180440: 19881:      0    0 Total of 0
1607180440: 19881: ending time = 1607180440, elapsed since start = 0:00:12

```

## 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	56300	0.00	0.00	TVector::At(int) const
0.00	0.00	0.00	46532	0.00	0.00	TVector::operator[](int) const
0.00	0.00	0.00	26422	0.00	0.00	TAvl<TVector, long>::Height(TAvl<TVector, long>::TAvlNode const*)
0.00	0.00	0.00	24030	0.00	0.00	char* std::__niter_base<char*>(char*)
0.00	0.00	0.00	20079	0.00	0.00	TVector::Size() const
0.00	0.00	0.00	14794	0.00	0.00	char* std::__miter_base<char*>(char*)
		0.00	0.00	9768/56300		TVector::At(int) [15]
		0.00	0.00	46532/56300		TVector::operator[](int) const [9]
[8]	0.0	0.00	0.00	56300		TVector::At(int) const [8]
-----						
		0.00	0.00	46532/46532		operator<(TVector const&, TVector const&) [18]

[9] 0.0 0.00 0.00 46532 TVector::operator[](int) const [9]  
0.00 0.00 46532/56300 TVector::At(int) const [8]

-----  
0.00 0.00 12654/26422 TAvl<TVector,  
long>::Balance(TAvl<TVector, long>::TAvlNode const\*) [27]

0.00 0.00 13768/26422 TAvl<TVector,  
long>::Reheight(TAvl<TVector, long>::TAvlNode\*) [25]

[10] 0.0 0.00 0.00 26422 TAvl<TVector,  
long>::Height(TAvl<TVector, long>::TAvlNode const\*) [10]

-----  
0.00 0.00 1839/24030 char\* std::\_\_copy\_move\_a2<false, char  
const\*, char\*>(char const\*, char const\*, char\*) [35]

0.00 0.00 22191/24030 char\* std::\_\_copy\_move\_a2<false, char\*,  
char\*>(char\*, char\*, char\*) [23]

[11] 0.0 0.00 0.00 24030 char\* std::\_\_niter\_base<char\*>(char\*) [11]

-----  
0.00 0.00 50/20079 main [6]

0.00 0.00 799/20079 TDetailAvl::Lower(TVector&) [40]

0.00 0.00 19230/20079 operator<(TVector const&, TVector  
const&) [18]

[12] 0.0 0.00 0.00 20079 TVector::Size() const [12]



Построил на бумаге граф вызовов для меньшего количества

	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]
--	--	------	------	---------	--	----------------------

-----

[31]		0.00	0.00	66/66		operator<(TVector const&, TVector const&)
------	--	------	------	-------	--	---

[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:

c -- **unit64\_t** – на -- **unit32\_t**

c -- **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	46532	0.00	0.00	TVector::operator[](unsigned long) const
0.00	0.00	0.00	26422	0.00	0.00	TAvl<TVector, long>::Height(TAvl<TVector, long>::TAvlNode const*)
0.00	0.00	0.00	24030	0.00	0.00	std::__niter_base<char*>(char*)
0.00	0.00	0.00	20055	0.00	0.00	TVector::Size() const
0.00	0.00	0.00	14794	0.00	0.00	std::__miter_base<char*>(char*)
0.00	0.00	0.00	9797	0.00	0.00	TVector::~~TVector()

### Изменения

#### Число вызовов

TVector::Size() const уменьшилось на 5%

TVector::~~TVector() уменьшилось 0.2%

Остальные вызовы на 0.15% раз

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

### dmalloc

1607180701: 20161: Dmalloc version '5.5.2' from 'http://dmalloc.com/'

1607180701: 20161: flags = 0x4e48503, logfile 'logfile2'

1607180701: 20161: interval = 100, addr = 0, seen # = 0, limit = 0

```

1607180701: 20161: threads enabled, lock-on = 0, lock-init = 2
1607180701: 20161: starting time = 1607180701
1607180701: 20161: process pid = 34
1607180701: 20161: Dumping Chunk Statistics:
1607180701: 20161: basic-block 4096 bytes, alignment 8 bytes
1607180701: 20161: heap address range: 0x7fe95b030000 to 0x7fe95b421000,
4132864 bytes
1607180701: 20161:   user blocks: 73 blocks, 273408 bytes (61%)
1607180701: 20161:   admin blocks: 35 blocks, 143360 bytes (32%)
1607180701: 20161:   total blocks: 108 blocks, 442368 bytes
1607180701: 20161: heap checked 202
1607180701: 20161: alloc calls: malloc 10084, calloc 0, realloc 0, free 10077
1607180701: 20161: alloc calls: realloc 0, memalign 0, posix_memalign 0, valloc
0
1607180701: 20161: alloc calls: new 0, delete 0
1607180701: 20161:   current memory in use: 195584 bytes (7 pnts)
1607180701: 20161:   total memory allocated: 301862 bytes (10084 pnts)
1607180701: 20161:   max in use at one time: 231004 bytes (1319 pnts)
1607180701: 20161:   max alloced with 1 call: 72704 bytes
1607180701: 20161:   max unused memory space: 52980 bytes (18%)
1607180701: 20161: top 10 allocations:
1607180701: 20161: total-size  count in-use-size  count  source
1607180701: 20161:      61566  5988      61566  5988  Other pointers
1607180701: 20161:      61566  5988      61566  5988  Total of 0
1607180701: 20161: Dumping Not-Freed Pointers Changed Since Start:
1607180701: 20161: not freed: '0x7fe95b370008|s1' (32768 bytes) from
'unknown'
1607180701: 20161: not freed: '0x7fe95b380008|s1' (32768 bytes) from
'unknown'

```

1607180701: 20161: not freed: '0x7fe95b390008|s1' (32768 bytes) from 'unknown'  
1607180701: 20161: not freed: '0x7fe95b3a0008|s1' (8192 bytes) from 'unknown'  
1607180701: 20161: not freed: '0x7fe95b3b0008|s1' (8192 bytes) from 'unknown'  
1607180701: 20161: not freed: '0x7fe95b3d0008|s1' (8192 bytes) from 'unknown'  
1607180701: 20161: not freed: '0x7fe95b3e0008|s1' (72704 bytes) from 'unknown'  
1607180701: 20161: total-size count source  
1607180701: 20161:        0    0 Total of 0  
1607180701: 20161: ending time = 1607180701, elapsed since start = 0:00:00

## 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 происходит 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.