ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

КАФЕДРА ВС

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

«Оценка производительности подсистемы памяти»

по дисциплине «Архитектура вычислительных систем»

Выполнил: студент гр. ИП-811
Разумов Д.Б.
Проверил: ст. преп. Кафедры ВС
Ткачёва Т.А.

Новосибирск 2020

Содержание

Постановка задачи

Разработать программу (benchmark) для оценки производительности подсистемы памяти.

1. Написать программу(функцию) на языке C/C++/C# для оценки производительности подсистемы памяти.

На вход программы подать следующие аргументы.

1) Подсистема памяти. Предусмотреть возможность указать подсистему для проверки производительности: RAM (оперативная память), HDD/SSD и flash.

2) Размер блока данных в байтах, Кб или Мб. Если размерность не указана, то в байтах, если указана, то соответственно в Кбайтах или Мбайтах.

3) Число испытаний, т.е. число раз повторений измерений.

Пример вызова программы: `./memory_test –m RAM –b 1024|1Kb –l 10`

или

```
./memory_bandwidth ––memory-type RAM|HDD|SSD|flash
 ––block-size 1024|1Kb
 ––launch-count 10
```

В качестве блока данных использовать одномерный массив, в котором произведение числа элементов на их размерность равна требуемому размеру блока данных. Массив инициализировать случайными значениями. Для тестирования HDD/SSD и flash создать в программе файлы в соответствующих директориях.

Измерение времени реализовать с помощью функции clock_gettime() или аналогичной с точность до наносекунд. Измерять время исключительно на запись элемента в память или считывание из неё, без операций генерации или преобразования данных.

На выходе программы в одну строку CSV файла со следующей структурой:

[MemoryType;BlockSize;ElementType;BufferSize;LaunchNum;Timer;WriteTime;AverageWriteTime;WriteBandwidth;

AbsError(write);RelError(write);ReadTime;AverageReadTime;ReadBandwidthAbsError(read);RelError(read);], где

MemoryType – тип памяти (RAM|HDD|SSD|flash) или модель устройства, на котором проводятся испытания;

BlockSize – размер блока данных для записи и чтения на каждом испытании;

ElementType – тип элементов используемых для заполнения массива данных;

BufferSize – размер буфера, т.е. порции данных для выполнения одно операции записи или чтения;

LaunchNum – порядковый номер испытания;

Timer – название функции обращения к таймеру (для измерения времени);

WriteTime – время выполнения отдельного испытания с номером LaunchNum [секунды];

AverageWriteTime – среднее время записи из LaunchNum испытаний [секунды];

WriteBandwidth – пропускная способность памяти (BLOCK_SIZE/AverageWriteTime) * 106 [Mb/s]

AbsError(write) – абсолютная погрешность измерения времени записи или СКО [секунды];

RelError(write) – относительная погрешность измерения времени [%];

ReadTime – время выполнения отдельного испытания LaunchNum [секунды];

AverageReadTime – среднее время записи из LaunchNum испытаний [секунды];

ReadBandwidth – пропускная способность памяти (BLOCK_SIZE/AverageReadTime) * 106 [Мб/сек.]

AbsError(read) – абсолютная погрешность измерения времени чтения или СКО [секунды];

RelError(read) – относительная погрешность измерения времени [%].

2. Написать программу(функцию) на языке C/C++/C# или скрипт (benchmark) реализующий серию испытаний программы(функции) из п.1. Оценить пропускную способность оперативной памяти при работе с блоками данных равными объёму кэш-линии, кэш-памяти L1, L2 и L3 уровня и превышающего его. Для HDD|SSD и flash провести серию из 20 испытаний с блоками данных начиная с 4 Мб с шагом 4Мб. Результаты всех испытаний сохранить в один CSV файл со структурой, описанной в п.1.

* Для HDD|SSD и flash оценить влияние размера буфера (BufferSize) на пропускную способность памяти.

3. На основе CSV файла построить сводные таблицы и диаграммы отражающие:

1) Зависимость пропускной способности записи и чтения от размера блока данных (BlockSize) для разного типа памяти;

2) Зависимость погрешности измерения пропускной способности от размера блока данных для разного типа памяти;

3) Зависимость погрешности измерений от числа испытаний LaunchNum;

4) * Зависимость пропускной способности памяти от размера буфера для HDD|SSD и flash памяти;

4. ** Оценить пропускную способность файла подкачки (windows) или раздела SWAP (linux). Сравнить с пропускной способностью RAM, HDD/ SSD и flash.

Выполнение работы

Моя лабораторная работа состоит из файлов main.cpp, foo.cpp, foo.h и нескольких скриптов. При запуске программы main.exe можно задать аргументы, аналогичные тем, что приведены в примере в задании 1. Обработка аргументов происходит в функции Process_parameters(). Затем полученные параметры испытаний передаются в функцию Tests_handler(). В этой функции вызывается функция Test_RAM() (если параметр типа памяти был RAM) или Test_storage_device() (если параметр типа памяти был SSD или flash). В обоих этих функциях создается динамический массив типа uint8_t (фиксированный 1 байт) размерностью block-size. Далее массив заполняется рандомными числами и происходит запись и чтение массива в соответствующий тип памяти. Если пользователь при запуске указал аргумент `--maximum-buffer`, то запись и чтение будет происходит не через цикл по одному элементу, а через функции memcpy(), fwrite(), fread().

В функциях Test_RAM() и Test_storage_device() вычисляется время в соответствии с заданием. Вызываются эти функции launch_count раз. После этого в Tests_handler() вычисляются нужные значения для записи в файл output.csv в соответствии с заданием.

Для выполнения заданий 2 и 3 используются Bash- и gnuplot-скрипты. В Bash-скриптах сначала идет очищение файла output.csv, затем программа main.exe вызывается несколько раз с нужными по заданию аргументами. Все данные сохраняются вместе. Для 3-го задания gnuplot-скрипты создают изображения из данных output.csv.

Задание *

В моей программе есть только два вида буфера: размером 1 байт или block-size байт. Один байт считывается/записывается через цикл отдельно для каждого элемента массива. Block-size байт считывается/записывается через такие функции, как memcpy(), fread(), fwrite(). Протестировав память с этими буферами, можно легко заметить, что чем больше буфер, тем больше пропускная способность памяти.

Результат работы

```
dmitry@dmitry-pc:~/acs/lab3/source$ ./main.exe --block-size 3Kb -l 8 -m SSD --maximum-buffer
--- arguments of main(): ---
mem_type = SSD
mem_size = 3072 bytes
launch_cnt = 8
max_buffer = 1
dmitry@dmitry-pc:~/acs/lab3/source$ ./main.exe
--- arguments of main(): ---
mem_type = RAM
mem_size = 1024 bytes
launch_cnt = 10
max_buffer = 0
dmitry@dmitry-pc:~/acs/lab3/source$
```

Рисунок 1. Запуск main.exe с аргументами. Если какие-то аргументы отсутствуют, то используется аргументы по умолчанию (в коде).

MemoryType	BlockSize	ElementType	BufferSize	LaunchNum	Timer	WriteTime	AverageWriteTime	WriteBandwidth	AbsError(write)	RelError(write)	ReadTime	AverageReadTime	ReadBandwidth	AbsError(read)	RelError(read)
SSD	3072	uint8_t	3072	1	clock()	1.500000e-05	6.875000e-06	4.468364e+14	3.139964e-06	1.434091e-04	1.800000e-05	1.175000e-05	2.614468e+14	2.861381e-06	6.968085e-05%
SSD	3072	uint8_t	3072	2	clock()	7.000000e-06	6.875000e-06	4.468364e+14	3.139964e-06	1.434091e-04	1.100000e-05	1.175000e-05	2.614468e+14	2.861381e-06	6.968085e-05%
SSD	3072	uint8_t	3072	3	clock()	5.000000e-06	6.875000e-06	4.468364e+14	3.139964e-06	1.434091e-04	1.000000e-05	1.175000e-05	2.614468e+14	2.861381e-06	6.968085e-05%
SSD	3072	uint8_t	3072	4	clock()	6.000000e-06	6.875000e-06	4.468364e+14	3.139964e-06	1.434091e-04	1.500000e-05	1.175000e-05	2.614468e+14	2.861381e-06	6.968085e-05%
SSD	3072	uint8_t	3072	5	clock()	6.000000e-06	6.875000e-06	4.468364e+14	3.139964e-06	1.434091e-04	1.000000e-05	1.175000e-05	2.614468e+14	2.861381e-06	6.968085e-05%
SSD	3072	uint8_t	3072	6	clock()	5.000000e-06	6.875000e-06	4.468364e+14	3.139964e-06	1.434091e-04	1.000000e-05	1.175000e-05	2.614468e+14	2.861381e-06	6.968085e-05%
SSD	3072	uint8_t	3072	7	clock()	5.000000e-06	6.875000e-06	4.468364e+14	3.139964e-06	1.434091e-04	1.000000e-05	1.175000e-05	2.614468e+14	2.861381e-06	6.968085e-05%
SSD	3072	uint8_t	3072	8	clock()	6.000000e-06	6.875000e-06	4.468364e+14	3.139964e-06	1.434091e-04	1.000000e-05	1.175000e-05	2.614468e+14	2.861381e-06	6.968085e-05%
RAM	1024	uint8_t	1	1	clock()	1.700000e-05	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05	1.700000e-05	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05%
RAM	1024	uint8_t	1	2	clock()	9.000000e-06	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05	9.000000e-06	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05%
RAM	1024	uint8_t	1	3	clock()	1.000000e-05	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05	1.000000e-05	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05%
RAM	1024	uint8_t	1	4	clock()	1.000000e-05	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05	1.000000e-05	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05%
RAM	1024	uint8_t	1	5	clock()	1.000000e-05	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05	1.000000e-05	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05%
RAM	1024	uint8_t	1	6	clock()	9.000000e-06	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05	9.000000e-06	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05%
RAM	1024	uint8_t	1	7	clock()	9.000000e-06	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05	9.000000e-06	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05%
RAM	1024	uint8_t	1	8	clock()	1.000000e-05	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05	1.000000e-05	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05%
RAM	1024	uint8_t	1	9	clock()	1.000000e-05	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05	1.000000e-05	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05%
RAM	1024	uint8_t	1	10	clock()	1.000000e-05	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05	1.000000e-05	1.040000e-05	9.846154e+13	2.244994e-06	4.846154e-05%

Рисунок 2. Файл output.csv, где хранятся данные согласно заданию 1.

MemoryType	BlockSize	ElementType	BufferSize	LaunchNum	Timer	WriteTime	AverageWriteTime	WriteBandwidth	AbsError(write)	RelError(write)	ReadTime	AverageReadTime	ReadBandwidth	AbsError(read)	RelError(read)
RAM	64	uint8_t	1	1	clock()	1.000000e-06	1.000000e-06	6.400000e+13	0.000000e+00	0.000000e+00	1.000000e-06	1.000000e-06	6.400000e+13	0.000000e+00	0.000000e+00%
RAM	64	uint8_t	1	2	clock()	1.000000e-06	1.000000e-06	6.400000e+13	0.000000e+00	0.000000e+00	1.000000e-06	1.000000e-06	6.400000e+13	0.000000e+00	0.000000e+00%
RAM	64	uint8_t	1	3	clock()	1.000000e-06	1.000000e-06	6.400000e+13	0.000000e+00	0.000000e+00	1.000000e-06	1.000000e-06	6.400000e+13	0.000000e+00	0.000000e+00%
RAM	32768	uint8_t	1	1	clock()	6.700000e-05	6.566667e-05	4.990051e+14	7.408704e-06	8.358714e-05	6.700000e-05	6.566667e-05	4.990051e+14	7.408704e-06	8.358714e-05%
RAM	32768	uint8_t	1	2	clock()	5.600000e-05	6.566667e-05	4.990051e+14	7.408704e-06	8.358714e-05	5.600000e-05	6.566667e-05	4.990051e+14	7.408704e-06	8.358714e-05%
RAM	32768	uint8_t	1	3	clock()	7.400000e-05	6.566667e-05	4.990051e+14	7.408704e-06	8.358714e-05	7.400000e-05	6.566667e-05	4.990051e+14	7.408704e-06	8.358714e-05%
RAM	262144	uint8_t	1	1	clock()	5.290000e-04	5.633333e-04	4.653444e+14	6.097176e-05	6.599211e-04	5.290000e-04	5.633333e-04	4.653444e+14	6.097176e-05	6.599211e-04%
RAM	262144	uint8_t	1	2	clock()	5.120000e-04	5.633333e-04	4.653444e+14	6.097176e-05	6.599211e-04	5.120000e-04	5.633333e-04	4.653444e+14	6.097176e-05	6.599211e-04%
RAM	262144	uint8_t	1	3	clock()	6.490000e-04	5.633333e-04	4.653444e+14	6.097176e-05	6.599211e-04	6.490000e-04	5.633333e-04	4.653444e+14	6.097176e-05	6.599211e-04%
RAM	4194304	uint8_t	1	1	clock()	8.747000e-03	8.400000e-03	4.993219e+14	2.454071e-04	7.169603e-04	8.747000e-03	8.400000e-03	4.993219e+14	2.454071e-04	7.169603e-04%
RAM	4194304	uint8_t	1	2	clock()	8.232000e-03	8.400000e-03	4.993219e+14	2.454071e-04	7.169603e-04	8.232000e-03	8.400000e-03	4.993219e+14	2.454071e-04	7.169603e-04%
RAM	4194304	uint8_t	1	3	clock()	8.221000e-03	8.400000e-03	4.993219e+14	2.454071e-04	7.169603e-04	8.221000e-03	8.400000e-03	4.993219e+14	2.454071e-04	7.169603e-04%
RAM	4456448	uint8_t	1	1	clock()	8.998000e-03	8.810333e-03	5.058206e+14	1.332275e-04	2.014629e-04	8.998000e-03	8.810333e-03	5.058206e+14	1.332275e-04	2.014629e-04%
RAM	4456448	uint8_t	1	2	clock()	8.731000e-03	8.810333e-03	5.058206e+14	1.332275e-04	2.014629e-04	8.731000e-03	8.810333e-03	5.058206e+14	1.332275e-04	2.014629e-04%
RAM	4456448	uint8_t	1	3	clock()	8.702000e-03	8.810333e-03	5.058206e+14	1.332275e-04	2.014629e-04	8.702000e-03	8.810333e-03	5.058206e+14	1.332275e-04	2.014629e-04%

Рисунок 3. Выполнение задания 2. Пропускная способность (для RAM) растет по мере того, как увеличивается объем оперируемой памяти. Однако исключение составляет объем кэш-линии — здесь самая большая пропускная способность.

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	MemoryType	BlockSize	ElementType	BufferSize	LaunchNum	Timer	WriteTime	AverageWriteTime	WriteBandwidth	AbsError(write)	RelError(write)	ReadTime	AverageReadTime	ReadBandwidth	AbsError(read)	RelError(read)
2	SSD	4194304	uint8_t	4194304	1	clock()	2.434000e-03	2.434000e-03	1.723214e+15	0.000000e+00	0.000000e+00	9.930000e-04	9.930000e-04	4.223871e+15	0.000000e+00	0.000000e+00
3	SSD	8388608	uint8_t	8388608	1	clock()	4.485000e-03	4.485000e-03	1.870370e+15	0.000000e+00	0.000000e+00	2.469000e-03	2.469000e-03	3.397573e+15	0.000000e+00	0.000000e+00
4	SSD	12582912	uint8_t	12582912	1	clock()	6.233000e-03	6.233000e-03	2.018757e+15	0.000000e+00	0.000000e+00	3.342000e-03	3.342000e-03	3.765084e+15	0.000000e+00	0.000000e+00
5	SSD	16777216	uint8_t	16777216	1	clock()	8.243000e-03	8.243000e-03	2.035329e+15	0.000000e+00	0.000000e+00	4.728000e-03	4.728000e-03	3.548481e+15	0.000000e+00	0.000000e+00
6	SSD	20971520	uint8_t	20971520	1	clock()	1.039100e-02	1.039100e-02	2.018239e+15	0.000000e+00	0.000000e+00	5.694000e-03	5.694000e-03	3.683091e+15	0.000000e+00	0.000000e+00
7	SSD	25165824	uint8_t	25165824	1	clock()	1.248300e-02	1.248300e-02	2.016008e+15	0.000000e+00	0.000000e+00	1.181000e-02	1.181000e-02	2.130891e+15	0.000000e+00	0.000000e+00
8	SSD	29360128	uint8_t	29360128	1	clock()	1.427800e-02	1.427800e-02	2.056319e+15	0.000000e+00	0.000000e+00	8.704000e-03	8.704000e-03	3.373176e+15	0.000000e+00	0.000000e+00
9	SSD	33554432	uint8_t	33554432	1	clock()	1.656300e-02	1.656300e-02	2.025867e+15	0.000000e+00	0.000000e+00	1.195800e-02	1.195800e-02	2.806024e+15	0.000000e+00	0.000000e+00
10	SSD	37748736	uint8_t	37748736	1	clock()	1.906700e-02	1.906700e-02	1.979794e+15	0.000000e+00	0.000000e+00	1.604000e-02	1.604000e-02	2.353412e+15	0.000000e+00	0.000000e+00
11	SSD	41943040	uint8_t	41943040	1	clock()	2.049200e-02	2.049200e-02	2.046801e+15	0.000000e+00	0.000000e+00	1.893500e-02	1.893500e-02	2.215106e+15	0.000000e+00	0.000000e+00
12	SSD	46137344	uint8_t	46137344	1	clock()	3.285200e-02	3.285200e-02	1.404400e+15	0.000000e+00	0.000000e+00	2.345800e-02	2.345800e-02	1.966806e+15	0.000000e+00	0.000000e+00
13	SSD	50331648	uint8_t	50331648	1	clock()	3.562000e-02	3.562000e-02	1.413017e+15	0.000000e+00	0.000000e+00	2.507700e-02	2.507700e-02	2.007084e+15	0.000000e+00	0.000000e+00
14	SSD	54525952	uint8_t	54525952	1	clock()	3.865200e-02	3.865200e-02	1.410689e+15	0.000000e+00	0.000000e+00	2.427900e-02	2.427900e-02	2.245807e+15	0.000000e+00	0.000000e+00
15	SSD	58720256	uint8_t	58720256	1	clock()	4.167300e-02	4.167300e-02	1.409072e+15	0.000000e+00	0.000000e+00	2.353200e-02	2.353200e-02	2.495336e+15	0.000000e+00	0.000000e+00
16	SSD	62914560	uint8_t	62914560	1	clock()	3.289700e-02	3.289700e-02	1.912471e+15	0.000000e+00	0.000000e+00	2.620800e-02	2.620800e-02	2.400586e+15	0.000000e+00	0.000000e+00
17	SSD	67108864	uint8_t	67108864	1	clock()	3.265100e-02	3.265100e-02	2.055339e+15	0.000000e+00	0.000000e+00	2.977700e-02	2.977700e-02	2.253715e+15	0.000000e+00	0.000000e+00
18	SSD	71303168	uint8_t	71303168	1	clock()	3.451600e-02	3.451600e-02	2.065800e+15	0.000000e+00	0.000000e+00	2.898800e-02	2.898800e-02	2.459748e+15	0.000000e+00	0.000000e+00
19	SSD	75497472	uint8_t	75497472	1	clock()	5.363800e-02	5.363800e-02	1.407537e+15	0.000000e+00	0.000000e+00	2.757000e-02	2.757000e-02	2.738392e+15	0.000000e+00	0.000000e+00
20	SSD	79691776	uint8_t	79691776	1	clock()	5.637700e-02	5.637700e-02	1.413551e+15	0.000000e+00	0.000000e+00	3.383600e-02	3.383600e-02	2.355236e+15	0.000000e+00	0.000000e+00
21	SSD	83886080	uint8_t	83886080	1	clock()	4.066500e-02	4.066500e-02	2.062857e+15	0.000000e+00	0.000000e+00	3.162700e-02	3.162700e-02	2.652357e+15	0.000000e+00	0.000000e+00
22	flash	4194304	uint8_t	4194304	1	clock()	4.613000e-03	4.613000e-03	9.092356e+14	0.000000e+00	0.000000e+00	1.998000e-03	1.998000e-03	2.099251e+15	0.000000e+00	0.000000e+00
23	flash	8388608	uint8_t	8388608	1	clock()	1.867700e-02	1.867700e-02	4.491411e+14	0.000000e+00	0.000000e+00	2.468000e-03	2.468000e-03	3.398950e+15	0.000000e+00	0.000000e+00
24	flash	12582912	uint8_t	12582912	1	clock()	3.418800e-02	3.418800e-02	3.680505e+14	0.000000e+00	0.000000e+00	6.709000e-03	6.709000e-03	1.875527e+15	0.000000e+00	0.000000e+00
25	flash	16777216	uint8_t	16777216	1	clock()	1.973900e-02	1.973900e-02	8.499527e+14	0.000000e+00	0.000000e+00	6.410000e-03	6.410000e-03	2.617350e+15	0.000000e+00	0.000000e+00
26	flash	20971520	uint8_t	20971520	1	clock()	5.148800e-02	5.148800e-02	4.073089e+14	0.000000e+00	0.000000e+00	7.971000e-03	7.971000e-03	2.630977e+15	0.000000e+00	0.000000e+00
27	flash	25165824	uint8_t	25165824	1	clock()	5.356800e-02	5.356800e-02	4.697921e+14	0.000000e+00	0.000000e+00	8.795000e-03	8.795000e-03	2.861379e+15	0.000000e+00	0.000000e+00
28	flash	29360128	uint8_t	29360128	1	clock()	3.361100e-02	3.361100e-02	8.735274e+14	0.000000e+00	0.000000e+00	9.894000e-03	9.894000e-03	2.967468e+15	0.000000e+00	0.000000e+00
29	flash	33554432	uint8_t	33554432	1	clock()	3.749700e-02	3.749700e-02	8.948564e+14	0.000000e+00	0.000000e+00	1.077000e-02	1.077000e-02	3.115546e+15	0.000000e+00	0.000000e+00
30	flash	37748736	uint8_t	37748736	1	clock()	4.128300e-02	4.128300e-02	9.143894e+14	0.000000e+00	0.000000e+00	1.961000e-02	1.961000e-02	1.924974e+15	0.000000e+00	0.000000e+00
31	flash	41943040	uint8_t	41943040	1	clock()	4.389700e-02	4.389700e-02	9.554876e+14	0.000000e+00	0.000000e+00	1.720800e-02	1.720800e-02	2.437415e+15	0.000000e+00	0.000000e+00
32	flash	46137344	uint8_t	46137344	1	clock()	6.068800e-02	6.068800e-02	7.602383e+14	0.000000e+00	0.000000e+00	2.384100e-02	2.384100e-02	1.935210e+15	0.000000e+00	0.000000e+00
33	flash	50331648	uint8_t	50331648	1	clock()	5.488000e-02	5.488000e-02	9.171219e+14	0.000000e+00	0.000000e+00	1.522700e-02	1.522700e-02	3.305421e+15	0.000000e+00	0.000000e+00
34	flash	54525952	uint8_t	54525952	1	clock()	7.185900e-02	7.185900e-02	7.587909e+14	0.000000e+00	0.000000e+00	1.525200e-02	1.525200e-02	3.575003e+15	0.000000e+00	0.000000e+00
35	flash	58720256	uint8_t	58720256	1	clock()	7.320700e-02	7.320700e-02	8.021126e+14	0.000000e+00	0.000000e+00	1.672800e-02	1.672800e-02	3.510297e+15	0.000000e+00	0.000000e+00
36	flash	62914560	uint8_t	62914560	1	clock()	7.501000e-02	7.501000e-02	8.387490e+14	0.000000e+00	0.000000e+00	1.796300e-02	1.796300e-02	3.502453e+15	0.000000e+00	0.000000e+00
37	flash	67108864	uint8_t	67108864	1	clock()	7.050700e-02	7.050700e-02	9.518043e+14	0.000000e+00	0.000000e+00	1.875300e-02	1.875300e-02	3.578567e+15	0.000000e+00	0.000000e+00
38	flash	71303168	uint8_t	71303168	1	clock()	7.777400e-02	7.777400e-02	9.167995e+14	0.000000e+00	0.000000e+00	1.854500e-02	1.854500e-02	3.844873e+15	0.000000e+00	0.000000e+00
39	flash	75497472	uint8_t	75497472	1	clock()	7.905800e-02	7.905800e-02	9.549631e+14	0.000000e+00	0.000000e+00	2.047000e-02	2.047000e-02	3.688201e+15	0.000000e+00	0.000000e+00
40	flash	79691776	uint8_t	79691776	1	clock()	9.899600e-02	9.899600e-02	8.050000e+14	0.000000e+00	0.000000e+00	2.183400e-02	2.183400e-02	3.649894e+15	0.000000e+00	0.000000e+00
41	flash	83886080	uint8_t	83886080	1	clock()	8.775400e-02	8.775400e-02	9.559231e+14	0.000000e+00	0.000000e+00	2.256500e-02	2.256500e-02	3.717531e+15	0.000000e+00	0.000000e+00

Рисунок 4. Выполнение задания 2. Работа с памятью на SSD быстрее, чем на USB3.0.

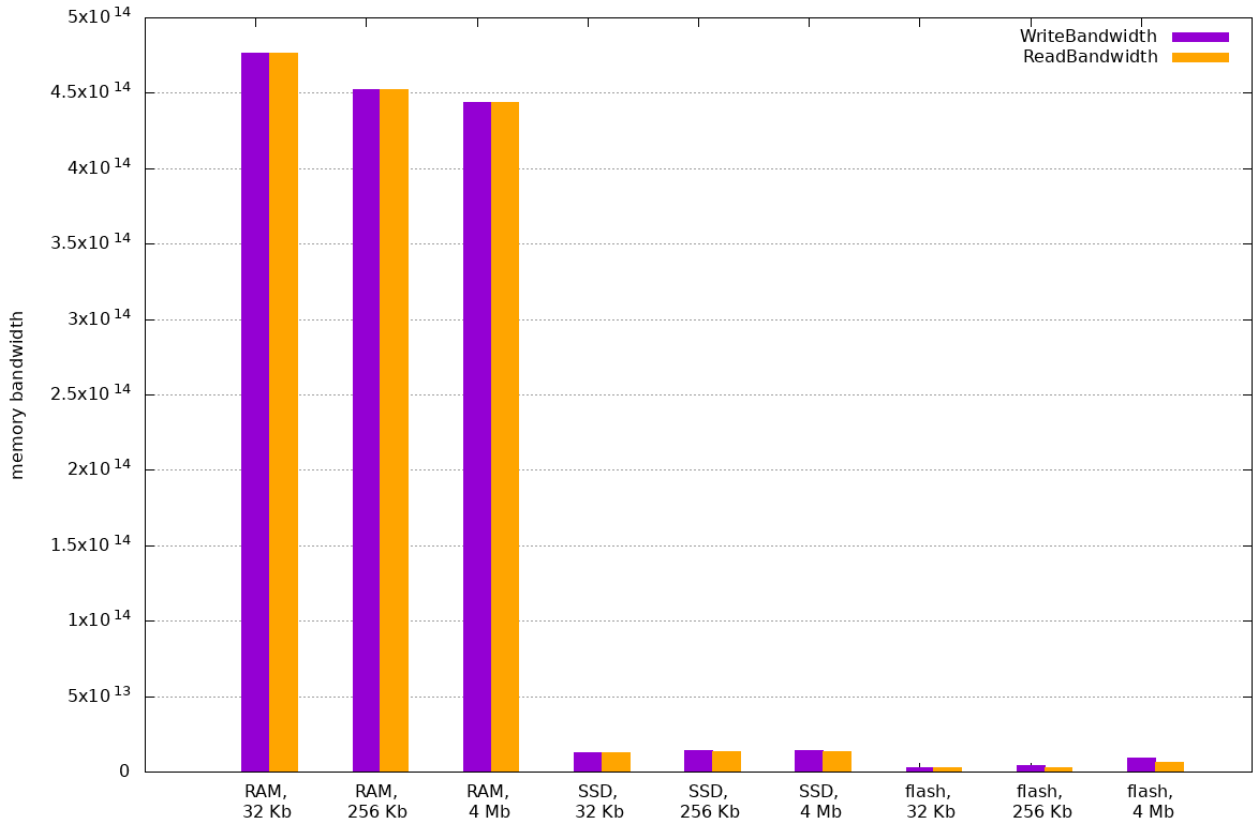

Рисунок 5. Выполнение задания 3.1.

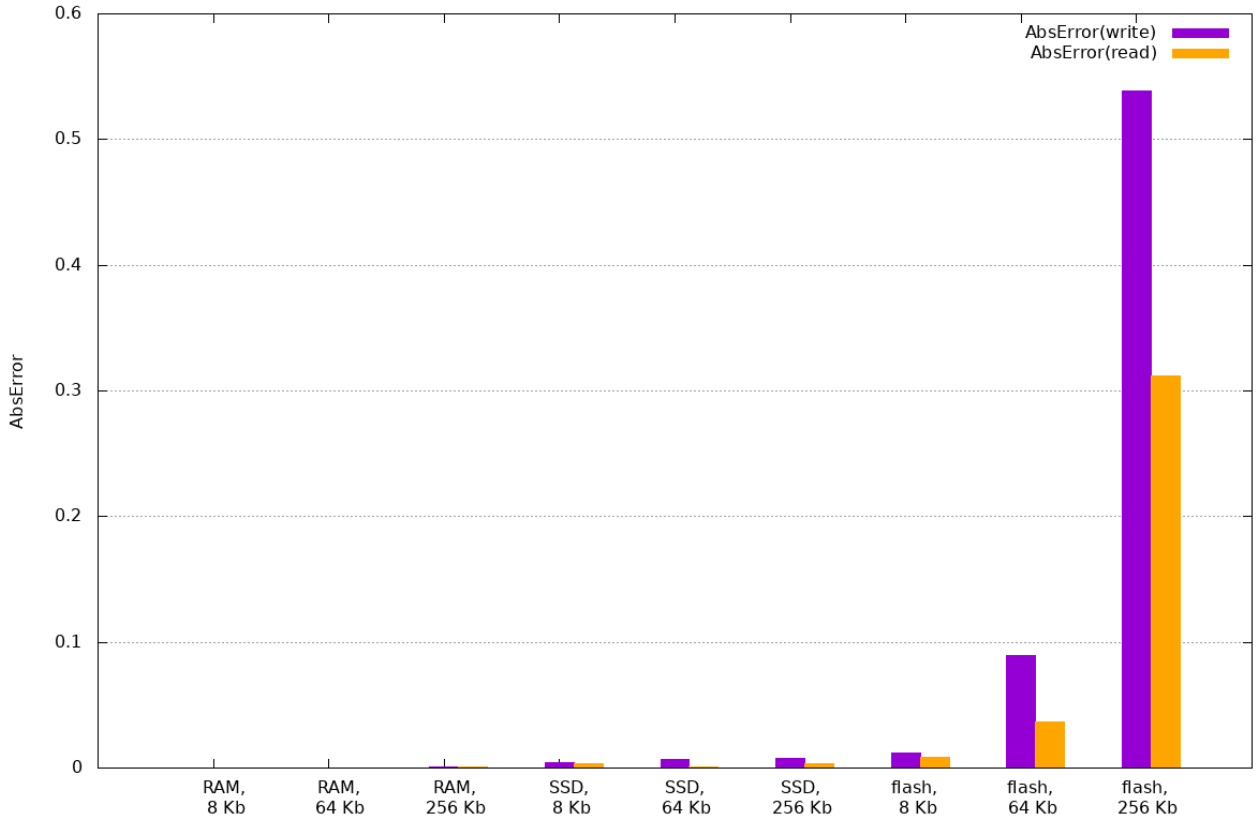

Рисунок 6. Выполнение задания 3.2.

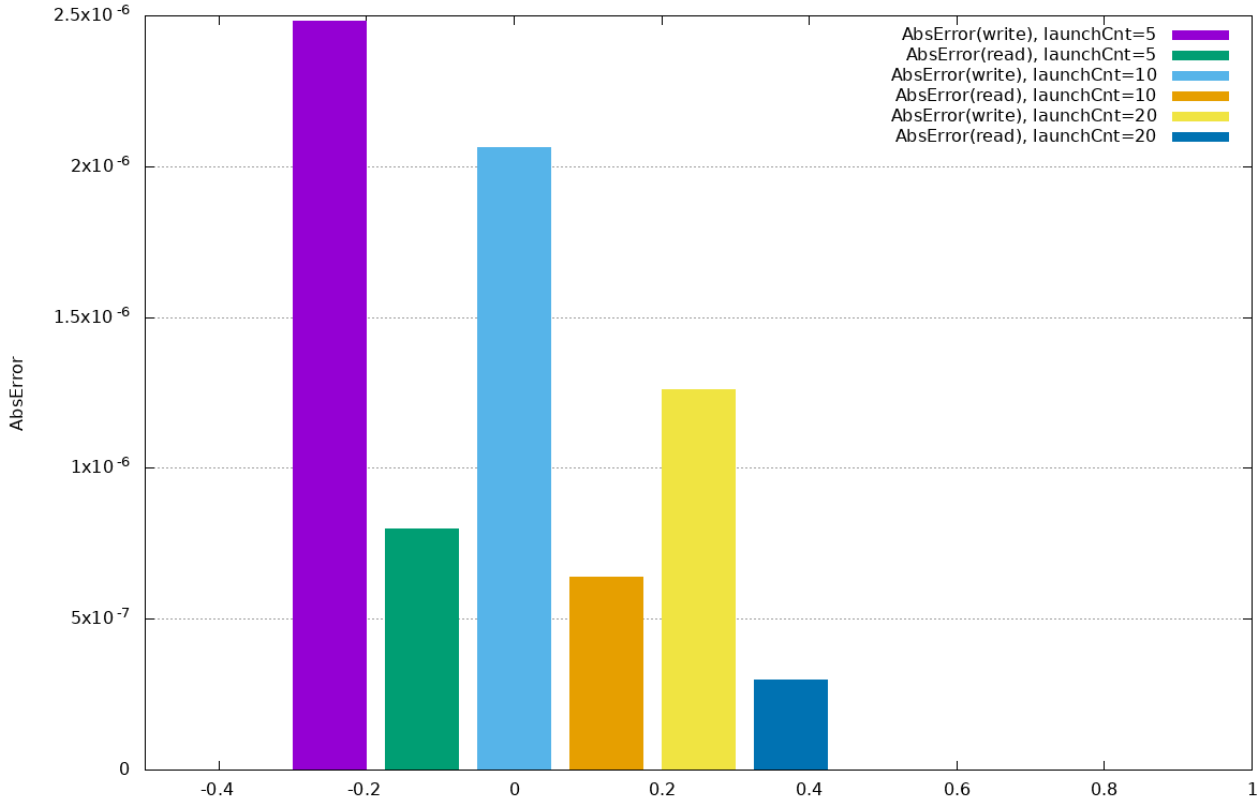

Рисунок 7. Выполнение задания 3.3 (тип памяти — SSD).

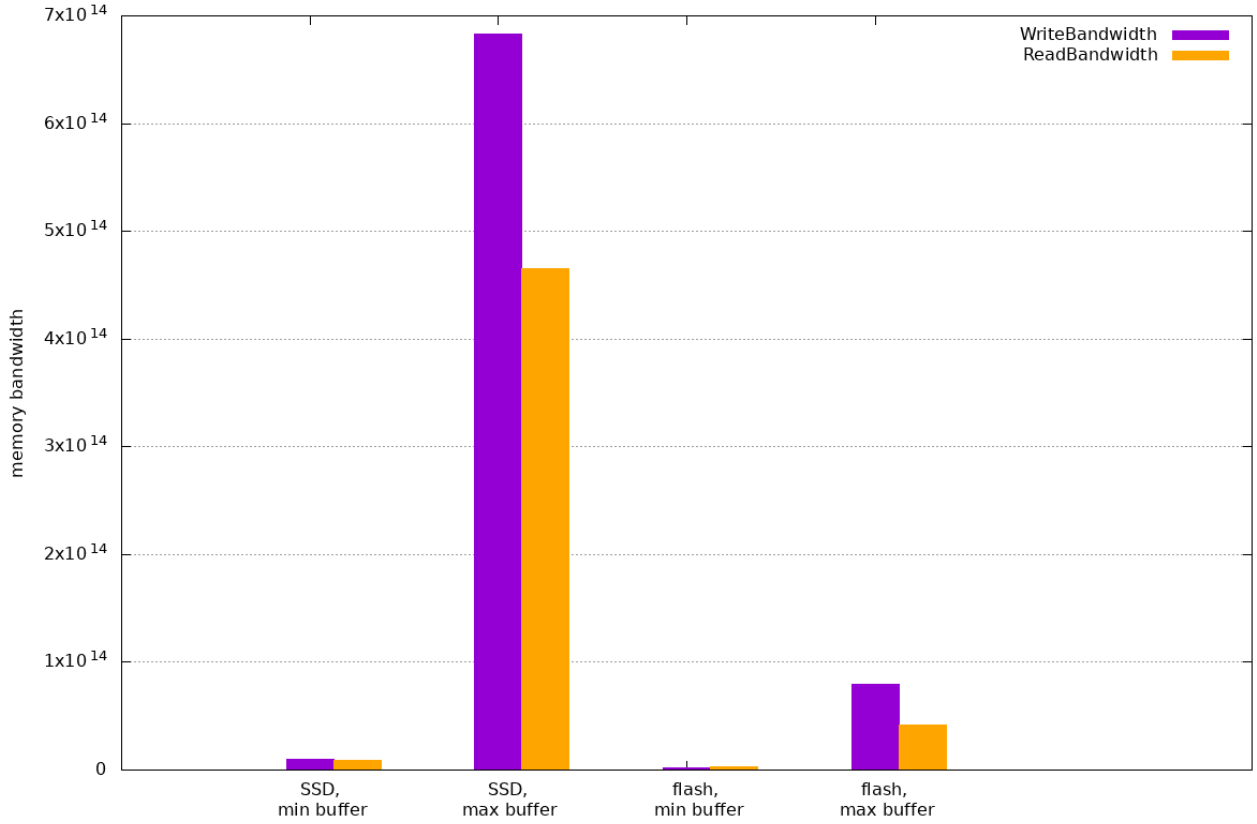

Рисунок 8. Выполнение задания *.

Приложение

Листинг source/main.cpp

```
#include "foo.h"

int main(int argc, char *argv[]) {
    ///4 параметра тестирования:
    char* mem_type = (char*) malloc(5); ///если пользователь ввел
    strcpy(mem_type, "RAM\0"); ///не все параметры, то будут
    long long mem_size = 1024; ///использоваться параметры по умолчанию
    long long launch_cnt = 10;
    bool max_buffer = false;
    ///считывание параметров заданных через аргументы ф-ии main:
    if (Process_parameters(argc, argv, mem_type, mem_size,
        launch_cnt, max_buffer) == 1)
    {
        return 1;
    }
    ///проверка работы process_parameters():
    printf("--- arguments of main(): --- \n");
    printf("mem_type = %s \n", mem_type);
    printf("mem_size = %lld bytes \n", mem_size);
    printf("launch_cnt = %lld \n", launch_cnt);
    printf("max_buffer = %d \n", max_buffer);

    srand(time(0));
    Tests_handler(mem_type, launch_cnt, mem_size, max_buffer);

    free(mem_type);
    return 0;
}
```

Листинг source/foo.h

```
#ifndef FOO
#define FOO

#include <math.h>
#include <stdio.h>
#include <stdlib.h>
#include <time.h>
#include <string.h>
#include <stdint.h> //подключение фиксированных типов данных

int Process_parameters(int argc, char *argv[], char* mem_type,
                       long long &mem_size, long long &launch_cnt,
                       bool &max_buffer);
int Tests_handler(char* mem_type, long long launch_cnt,
               long long mem_size, bool max_buffer);

#endif
```

Листинг source/foo.cpp

```
#include "foo.h"

int Process_parameters(int argc, char *argv[], char* mem_type,
                       long long &mem_size, long long &launch_cnt,
                       bool &max_buffer)
```

```
{
    int i;
    for (i = 1; i < argc; i++)
    {
        ///если тип памяти:
        if (strcmp("-m", argv[i]) == 0 ||
            strcmp("--memory-type", argv[i]) == 0)
        {
            i++;
            if (strcmp("RAM", argv[i]) == 0 ||
                strcmp("SSD", argv[i]) == 0 ||
                strcmp("flash", argv[i]) == 0 ||
                strcmp("SWAP", argv[i]) == 0)
            {
                strcpy(mem_type, argv[i]);
            }
            else {
                  printf("Error in arguments of main(): incorrect value for --
memory-type \n");
                return 1;
            }
        }
        ///если размер блока данных:
        else if (strcmp("-b", argv[i]) == 0 ||
                 strcmp("--block-size", argv[i]) == 0)
        {
            i++;
            size_t len = strlen(argv[i]);
            int coefficient = 1;
            if (argv[i][len-1] == 'b') { ///проверка на Кб или Мб
                if (argv[i][len-2] == 'K')
                    coefficient = 1024;
                else if (argv[i][len-2] == 'M')
                    coefficient = 1024 * 1024;
                else {
                     printf("Error in arguments of main(): incorrect description
of unit of measure byte \n");
                    return 1;
                }
                strncpy(argv[i], argv[i], len - 2); ///убрать единицы измерения
            }
            mem_size = atoll(argv[i]) * coefficient; ///приведение строки в long
long int
        }
        ///если число испытаний:
        else if (strcmp("-l", argv[i]) == 0 ||
                 strcmp("--launch-count", argv[i]) == 0)
        {
            i++;
            launch_cnt = atoll(argv[i]); ///приведение строки в long long int
            if (launch_cnt == 0) {
                  printf("Error in arguments of main(): incorrect value for --
launch-count \n");
                return 1;
            }
        }
        ///если задана опция на максимально возможный буфер:
        else if (strcmp("--maximum-buffer", argv[i]) == 0)
        {
```

```
            max_buffer = true;
        }
        else {
             printf("Error in arguments of main(): incorrect value (ind = %d) \
n", i);
            return 1;
        }
    }

    return 0;
}

int Test_RAM(long long mem_size, bool max_buffer,
             double &time_write, double &time_read)
 {
    uint8_t* arr1 = (uint8_t*) malloc(mem_size); ///тип данных - фиксированный 1
байт [0..255]
    long long i;
    for (i = 0; i < mem_size; i++) {
        arr1[i] = rand() % 256;
    }
    ///подсчет времени на запись массива:
    uint8_t* arr2 = (uint8_t*) malloc(mem_size);
    clock_t start, stop;
    if (max_buffer) {
        start = clock();
          memcpy(arr2, arr1, mem_size); ///копировать содержимое одной области
памяти в другую
        stop = clock();
    }
    else {
        start = clock();
        for (i = 0; i < mem_size; i++) {
            arr2[i] = arr1[i];
        }
        stop = clock();
    }
    time_write = ((double)(stop - start)) / CLOCKS_PER_SEC;
    time_read = time_write; ///чтение и запись - идентичные операции
    ///проверка:
//    for(i = 0; i < mem_size; i++) {
//        printf("i=%lld arr1[i]=%hhu arr2[i]=%hhu \n", i, arr1[i], arr2[i]);
//    }
    free(arr1);
    free(arr2);
    return 0;
}

int Test_storage_device(char* mem_type, long long mem_size, bool max_buffer,
                        double &time_write, double &time_read)
{
     uint8_t* arr = (uint8_t*) malloc(mem_size); ///тип данных - фиксированный 1
байт [0..255]
    long long i;
    for (i = 0; i < mem_size; i++) {
        arr[i] = rand() % 256;
    }
    ///открытие файла на накопителе для записи:
```

```
    FILE *fp;
    if (strcmp("SSD", mem_type) == 0) {///SSD
        if ((fp = fopen("test_SSD.txt", "w")) == NULL) {
            printf("Error in Test_storage_device(): can't open file test_SSD.txt
\n");
            return 1;
        }
    }
    else if (strcmp("flash", mem_type) == 0) { ///USB
          if ((fp = fopen("/media/dmitry/SMARTBUYUSB/test_flash.txt", "w")) ==
NULL) {
                    printf("Error in Test_storage_device(): can't open file
test_flash.txt in USB \n");
            return 1;
        }
    }
    else { ///swap
        if ((fp = fopen("/swapfile", "wb")) == NULL) {
             printf("Error in Test_storage_device(): can't open file swapfile \
n");
            return 1;
        }
    }
    ///запись массива:
    clock_t start, stop;
    if (max_buffer) {
        start = clock();
        fwrite(arr, mem_size, 1, fp);
        stop = clock();
    }
    else {
        start = clock();
        for (i = 0; i < mem_size; i++) {
            fprintf(fp, "%hhu ", arr[i]);
        }
        stop = clock();
    }
    time_write = ((double)(stop - start)) / CLOCKS_PER_SEC;
    ///открытие файла на накопителе для чтения:
    fclose(fp);
    if (strcmp("SSD", mem_type) == 0) {///SSD
        if ((fp = fopen("test_SSD.txt", "r")) == NULL) {
            printf("Error in Test_storage_device(): can't open file test_SSD.txt
\n");
            return 1;
        }
    }
    else if (strcmp("flash", mem_type) == 0) { ///USB
          if ((fp = fopen("/media/dmitry/SMARTBUYUSB/test_flash.txt", "r")) ==
NULL) {
                    printf("Error in Test_storage_device(): can't open file
test_flash.txt in USB \n");
            return 1;
        }
    }
    else { ///swap
        if ((fp = fopen("/swapfile", "r")) == NULL) {
             printf("Error in Test_storage_device(): can't open file swapfile \
n");
```

```
            return 1;
        }
    }
    ///чтение массива:
    if (max_buffer) {
        start = clock();
        fread(arr, mem_size, 1, fp);
        stop = clock();
    }
    else {
        start = clock();
        for (i = 0; i < mem_size; i++) {
            fscanf(fp, "%hhu", &arr[i]);
        }
        stop = clock();
    }
    time_read = ((double)(stop - start)) / CLOCKS_PER_SEC;

    free(arr);
    fclose(fp);
    return 0;
}


int Write_to_csv(char* mem_type, long long launch_cnt,
                 long long mem_size, bool max_buffer,
                 double time_write[], double avg_time_write,
                 double abs_error_write, double rel_error_write,
                 double time_read[], double avg_time_read,
                 double abs_error_read, double rel_error_read)
{
    ///открытие output.csv:
    FILE *fout;
    if ((fout = fopen("../data/output.csv", "a")) == NULL) {
        printf("Error in Write_to_csv(): can't open output.csv \n");
        return 1;
    }
                                                                    //fprintf(fout,
"MemoryType;BlockSize;ElementType;BufferSize;LaunchNum;Timer;WriteTime;AverageWr
iteTime;WriteBandwidth;AbsError(write);RelError(write);ReadTime;AverageReadTime;
ReadBandwidth;AbsError(read);RelError(read);\n");
    for (long long i = 0; i < launch_cnt; i++) {
        fprintf(fout, "%s;", mem_type); ///MemoryType
        fprintf(fout, "%lld;", mem_size); ///BlockSize
        fprintf(fout, "uint8_t;"); ///ElementType
        if (max_buffer) ///BufferSize
            fprintf(fout, "%lld;", mem_size);
        else
            fprintf(fout, "1;");
        fprintf(fout, "%lld;", i+1); ///LaunchNum
        fprintf(fout, "clock();"); ///Timer
        fprintf(fout, "%e;", time_write[i]); ///WriteTime
        fprintf(fout, "%e;", avg_time_write); ///AverageWriteTime
             fprintf(fout,  "%e;",  (double)mem_size  /  avg_time_write  *  1e6);
///WriteBandwidth
        fprintf(fout, "%e;", abs_error_write); ///AbsError(write)
        fprintf(fout, "%e%%;", rel_error_write); ///RelError(write)
        fprintf(fout, "%e;", time_read[i]); ///ReadTime
```

```
        fprintf(fout, "%e;", avg_time_read); ///AverageReadTime
              fprintf(fout, "%e;", (double)mem_size / avg_time_read * 1e6);
///ReadBandwidth
        fprintf(fout, "%e;", abs_error_read); ///AbsError(read)
        fprintf(fout, "%e%%;", rel_error_read); ///RelError(read)
        fprintf(fout, "\n");
    }

    return 0;
}

int Tests_handler(char* mem_type, long long launch_cnt,
               long long mem_size, bool max_buffer)
{
    ///начальные переменные для измерений:
    double summand1_write = 0, summand2_write = 0;
    double summand1_read = 0, summand2_read = 0;
    double time_write[launch_cnt]; ///время выполнения записи массива
    double time_read[launch_cnt]; ///время выполнения чтения массива
    ///измерения (тестирование):
    for (long long i = 0; i < launch_cnt; i++) {
        if (strncmp("RAM", mem_type, 3) == 0) { ///если RAM
             if (Test_RAM(mem_size, max_buffer, time_write[i], time_read[i]) ==
1)
                return 1;
        }
        else { ///если SSD или flash или SWAP
            if (Test_storage_device(mem_type, mem_size, max_buffer,
                time_write[i], time_read[i]) == 1)
                return 1;
        }
        summand1_write += time_write[i] * time_write[i]; ///первое слагаемое для
дисперсии
        summand2_write += time_write[i]; ///2-ое слагаемое
          summand1_read += time_read[i] * time_read[i]; ///первое слагаемое для
дисперсии
        summand2_read += time_read[i]; ///2-ое слагаемое
    }
    ///заключительные переменные для измерений записи:
    summand1_write /= launch_cnt;
    summand2_write /= launch_cnt;
     double avg_time_write = summand2_write; ///сумма времени всех тестов / n ==
среднее время
    summand2_write *= summand2_write;
      double dispersion_write = summand1_write - summand2_write; ///дисперсия
(точность измерения времени)
     double abs_error_write = sqrt(dispersion_write); ///среднее квадратическое
отклонение (погрешность)
        double rel_error_write = dispersion_write / avg_time_write * 100;
///относительная погрешность в %
    ///заключительные переменные для измерений чтения:
    summand1_read /= launch_cnt;
    summand2_read /= launch_cnt;
     double avg_time_read = summand2_read; ///сумма времени всех тестов / n ==
среднее время
    summand2_read *= summand2_read;
       double dispersion_read = summand1_read - summand2_read; ///дисперсия
(точность измерения времени)
```

```
      double abs_error_read = sqrt(dispersion_read); ///среднее квадратическое
отклонение (погрешность)
         double rel_error_read = dispersion_read / avg_time_read * 100;
///относительная погрешность в %

    Write_to_csv(mem_type, launch_cnt, mem_size, max_buffer,
                 time_write, avg_time_write,
                 abs_error_write, rel_error_write,
                 time_read, avg_time_read,
                 abs_error_read, rel_error_read);
    return 0;
}
```

Листинг scripts/s21.sh

```
#!/bin/bash

#очистить output.csv и записать туда заданную строку
echo
"MemoryType;BlockSize;ElementType;BufferSize;LaunchNum;Timer;WriteTime;AverageWr
iteTime;WriteBandwidth;\
AbsError(write);RelError(write);ReadTime;AverageReadTime;ReadBandwidth;AbsError(
read);RelError(read);" > ../data/output.csv

# $ getconf -a | grep CACHE
#dCache - cache lvl1 for data
#iCache - cache lvl1 for instructions
cache_line=$(getconf LEVEL1_DCACHE_LINESIZE)
cache_lvl1=$(getconf LEVEL1_DCACHE_SIZE)
cache_lvl2=$(getconf LEVEL2_CACHE_SIZE)
cache_lvl3=$(getconf LEVEL3_CACHE_SIZE)
cache_more=$(($cache_lvl3 + $cache_lvl2))

cd ../source
make
./../source/main.exe --launch-count 3 --memory-type RAM --block-size $cache_line
./../source/main.exe --launch-count 3 --memory-type RAM --block-size $cache_lvl1
./../source/main.exe --launch-count 3 --memory-type RAM --block-size $cache_lvl2
./../source/main.exe --launch-count 3 --memory-type RAM --block-size $cache_lvl3
./../source/main.exe --launch-count 3 --memory-type RAM --block-size $cache_more
```

Листинг scripts/s22.sh

```
#!/bin/bash

#очистить output.csv и записать туда заданную строку
echo
"MemoryType;BlockSize;ElementType;BufferSize;LaunchNum;Timer;WriteTime;AverageWr
iteTime;WriteBandwidth;\
AbsError(write);RelError(write);ReadTime;AverageReadTime;ReadBandwidth;AbsError(
read);RelError(read);" > ../data/output.csv

mem_size=4

cd ../source
make

for (( i=1; i <= 20; i++ ))
do
```

```
./../source/main.exe -l 1 -m SSD -b $mem_size"Mb" --maximum-buffer
mem_size=$(( $mem_size + 4))
done

mem_size=4
for (( i=1; i <= 20; i++ ))
do
./../source/main.exe -l 1 -m flash -b $mem_size"Mb" --maximum-buffer
mem_size=$(( $mem_size + 4))
done
```

Листинг scripts/s31-32.sh

```
#!/bin/bash

#очистить output.csv и записать туда заданную строку
echo
"MemoryType;BlockSize;ElementType;BufferSize;LaunchNum;Timer;WriteTime;AverageWr
iteTime;WriteBandwidth;\
AbsError(write);RelError(write);ReadTime;AverageReadTime;ReadBandwidth;AbsError(
read);RelError(read);" > ../data/output.csv

cd ../source
make
./../source/main.exe --launch-count 10 --memory-type RAM --block-size 8Kb
./../source/main.exe --launch-count 10 --memory-type RAM --block-size 64Kb
./../source/main.exe --launch-count 10 --memory-type RAM --block-size 256Kb
./../source/main.exe --launch-count 10 --memory-type SSD --block-size 8Kb
./../source/main.exe --launch-count 10 --memory-type SSD --block-size 64Kb
./../source/main.exe --launch-count 10 --memory-type SSD --block-size 256Kb
./../source/main.exe --launch-count 10 --memory-type flash --block-size 8Kb
./../source/main.exe --launch-count 10 --memory-type flash --block-size 64Kb
./../source/main.exe --launch-count 10 --memory-type flash --block-size 256Kb
```

Листинг scripts/d31.gpi

```
#! /usr/bin/gnuplot
#! /usr/bin/gnuplot -persist

#изображение, где будет диаграмма
set terminal png font "Verdana,12" size 1200, 800
set output "../data/diagram31.png"

#символ-раделитель в outout.csv
set datafile separator ';'

# ось игрек
set ylabel "memory bandwidth"

#установки сетки по одной оси
set grid ytics

#область значений для осей
set xrange [0:10]

#отступ снизу
set bmargin 4
```

```
#подписи по оси абсцисс
set xtics ("RAM,\n 32 Kb" 1, "RAM,\n 256 Kb" 2, "RAM,\n 4 Mb" 3, \
          "SSD,\n 32 Kb" 4, "SSD,\n 256 Kb" 5, "SSD,\n 4 Mb" 6, \
          "flash,\n 32 Kb" 7, "flash,\n 256 Kb" 8, "flash,\n 4 Mb" 9)

#передвинуть надписи по оси x на 2
set xtic offset character 1, 0

#установить стиль гистограмм
set style data histograms

#установить ширину столбцов 1 от максимальной
set boxwidth 1 absolute
set style fill solid 1

#считать информацию, every 5 - считывать значение только из каждой 5 строки
plot "../data/output.csv" every 10 using 9 title "WriteBandwidth", \
        "../data/output.csv" every 10 using 14 title "ReadBandwidth" lt rgb
'orange'
```

Листинг scripts/d32.gpi

```
#! /usr/bin/gnuplot
#! /usr/bin/gnuplot -persist

#изображение, где будет диаграмма
set terminal png font "Verdana,12" size 1200, 800
set output "../data/diagram32.png"

#символ-раделитель в outout.csv
set datafile separator ';'

# ось игрек
set ylabel "AbsError"

#установки сетки по одной оси
set grid ytics

#область значений для осей
set xrange [0:10]

#отступ снизу
set bmargin 4

#подписи по оси абсцисс
set xtics ("RAM,\n 8 Kb" 1, "RAM,\n 64 Kb" 2, "RAM,\n 256 Kb" 3, \
          "SSD,\n 8 Kb" 4, "SSD,\n 64 Kb" 5, "SSD,\n 256 Kb" 6, \
          "flash,\n 8 Kb" 7, "flash,\n 64 Kb" 8, "flash,\n 256 Kb" 9)

#передвинуть надписи по оси x на 2
set xtic offset character 1, 0

#установить стиль гистограмм
set style data histograms

#установить ширину столбцов 1 от максимальной
set boxwidth 1 absolute
```

```
set style fill solid 1

#считать информацию, every 5 - считывать значение только из каждой 5 строки
plot "../data/output.csv" every 10 using 11 title "AbsError(write)", \
        "../data/output.csv" every 10 using 16 title "AbsError(read)" lt rgb
'orange'
```

Листинг scripts/s33.sh

```
#!/bin/bash

#очистить output.csv и записать туда заданную строку
echo
"MemoryType;BlockSize;ElementType;BufferSize;LaunchNum;Timer;WriteTime;AverageWr
iteTime;WriteBandwidth;\
AbsError(write);RelError(write);ReadTime;AverageReadTime;ReadBandwidth;AbsError(
read);RelError(read);" > ../data/output.csv

cd ../source
make
./../source/main.exe --launch-count 5 --memory-type SSD --block-size 12Kb --
maximum-buffer
./../source/main.exe --launch-count 10 --memory-type SSD --block-size 12Kb --
maximum-buffer
./../source/main.exe --launch-count 20 --memory-type SSD --block-size 12Kb --
maximum-buffer
```

Листинг scripts/d33.gpi

```
#! /usr/bin/gnuplot
#! /usr/bin/gnuplot -persist

#изображение, где будет диаграмма
set terminal png font "Verdana,12" size 1200, 800
set output "../data/diagram33.png"

#символ-раделитель в outout.csv
set datafile separator ';'

# ось игрек
set ylabel "AbsError"

#установки сетки по одной оси
set grid ytics

#область значений для осей
set xrange [-0.5:1]

#отступ снизу
set bmargin 4

#передвинуть надписи по оси x на 2
#set xtic offset character 1, 0

#устаносить стиль гистограмм
set style data histograms

#установить ширину столбцов 1 от максимальной
```

```
set boxwidth 0.8 absolute
set style fill solid 1

plot "<(sed -n '5,5p' ../data/output.csv)" using 10 title "AbsError(write), launchCnt=5", \
       "<(sed -n '5,5p' ../data/output.csv)" using 15 title "AbsError(read), launchCnt=5", \
      "<(sed -n '15,15p' ../data/output.csv)" using 10 title "AbsError(write), launchCnt=10", \
       "<(sed -n '15,15p' ../data/output.csv)" using 15 title "AbsError(read), launchCnt=10", \
      "<(sed -n '35,35p' ../data/output.csv)" using 10 title "AbsError(write), launchCnt=20", \
       "<(sed -n '35,35p' ../data/output.csv)" using 15 title "AbsError(read), launchCnt=20"
```

Листинг scripts/start1.sh

```
#!/bin/bash

#очистить output.csv и записать туда заданную строку
echo
"MemoryType;BlockSize;ElementType;BufferSize;LaunchNum;Timer;WriteTime;AverageWriteTime;WriteBandwidth;\
AbsError(write);RelError(write);ReadTime;AverageReadTime;ReadBandwidth;AbsError(read);RelError(read);" > ../data/output.csv

cd ../source
make
./../source/main.exe --launch-count 10 --memory-type SSD --block-size 2Kb
./../source/main.exe --launch-count 10 --memory-type SSD --block-size 2Kb --maximum-buffer
./../source/main.exe --launch-count 10 --memory-type flash --block-size 2Kb
./../source/main.exe --launch-count 10 --memory-type flash --block-size 2Kb –maximum-buffer
```

Листинг scripts/start1.sh

```
#! /usr/bin/gnuplot
#! /usr/bin/gnuplot -persist

#изображение, где будет диаграмма
set terminal png font "Verdana,12" size 1200, 800
set output "../data/star1.png"

#символ-раделитель в outout.csv
set datafile separator ';'

# ось игрек
set ylabel "memory bandwidth"

#установки сетки по одной оси
set grid ytics

#область значений для осей
set xrange [0:6]

#отступ снизу
set bmargin 4
```

```
#подписи по оси абсцисс
set xtics ("SSD,\n min buffer" 1, "SSD,\n max buffer" 2, \
           "flash,\n min buffer" 3, "flash,\n max buffer" 4)

#передвинуть надписи по оси x на 2
set xtic offset character 1, 0

#устаносить стиль гистограмм
set style data histograms

#установить ширину столбцов 1 от максимальной
set boxwidth 1 absolute
set style fill solid 1

#считать информацию, every 5 - считывать значение только из каждой 5 строки
plot "../data/output.csv" every 10 using 9 title "WriteBandwidth", \
        "../data/output.csv"  every  10  using  14  title  "ReadBandwidth"  lt  rgb
'orange'
```