Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИУ — Информатика и системы управления
КАФЕДРА ИУ7 — Программное обеспечение ЭВМ и информационные технологии

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Студент

Хамзина Регина Ренатовна

Группа

ИУ7-83Б

Тип практики

преддипломная практика

Название предприятия

НУК ИУ МГТУ им. Н. Э. Баумана

Студент	_____	**Хамзина Р. Р.**
Руководитель практики от предприятия	_____	**Оленев А. А.**
Руководитель практики	_____	**Кострицкий А. С.**

Оценка_____

2023 г.

«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

УТВЕРЖДАЮ
Заведующий кафедрой ИУ7
_____ Рудаков И. В.
«13» мая 2023 г.

З А Д А Н И Е
на прохождение производственной практики (преддипломная практика)

Студент **4** курса группы **ИУ7-83Б**

Хамзина Регина Ренатовна

в период с 15.05.2023 г. по 28.05.2023 г.

Предприятие: **НУК ИУ МГТУ им. Н. Э. Баумана**

Руководитель практики от предприятия (наставник):

Оленев А. А.

Руководитель практики от кафедры:

Кострицкий А. С.

Задание:

1. Изучить программные средства проектирования и разработки информационных систем.

2. Собрать материалы в области разработки информационных систем.

3. Получить практические навыки в области разработки информационных систем.

Дата выдачи задания «13» мая 2023 г.

Руководитель практики от кафедры _____ **Кострицкий А. С.**

Руководитель практики от предприятия _____ **Оленев А. А.**

Студент _____ **Хамзина Р. Р.**

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Данные процессов операционной системы хранятся в оперативной памяти. При ее недостатке производительность системы может снижаться. Одним из способов увеличения объема доступной оперативной памяти является сжатие данных [1]. На качество сжатия помимо алгоритма сжатия влияет структура данных. Оценить качество сжатия, не тратя ресурсы на его выполнение, позволяет значение информационной энтропии исходных данных [2].

Во время выполнения выпускной квалификационной работы была разработана оптимизация метода сжатия страниц памяти с использованием подсчета информационной энтропии. Целью данной работы является разработка программного обеспечения, демонстрирующего практическую осуществимость спроектированной оптимизации метода. Для достижения поставленной цели необходимо выполнить следующие задачи:

— дать формальное описание постановки задачи оптимизации метода сжатия страниц памяти;

— выделить требования к программному обеспечению,

— выбрать средства программной реализации;

— описать структуру программного обеспечения;

— описать конфигурацию программного обеспечения.

1 Основная часть

1.1 Постановка задачи

Разработанная оптимизация метода сжатия страниц памяти заключается в принятии решения о его выполнении на основании вычисленного значения информационной энтропии исходных данных и состоит из следующих этапов:

1. Вычисление значения информационной энтропии с помощью метода подсчета.
2. Сравнение вычисленного значения информационной энтропии с пороговым значением.
3. Сжатие данных страницы в случае, если полученное значение информационной энтропии меньше порогового значения.

На рисунках 1.1-1.2 показаны IDEF0-диаграммы нулевого и первого уровней, формализующие поставленную задачу оптимизации метода сжатия страниц памяти с использованием подсчета информационной энтропии.

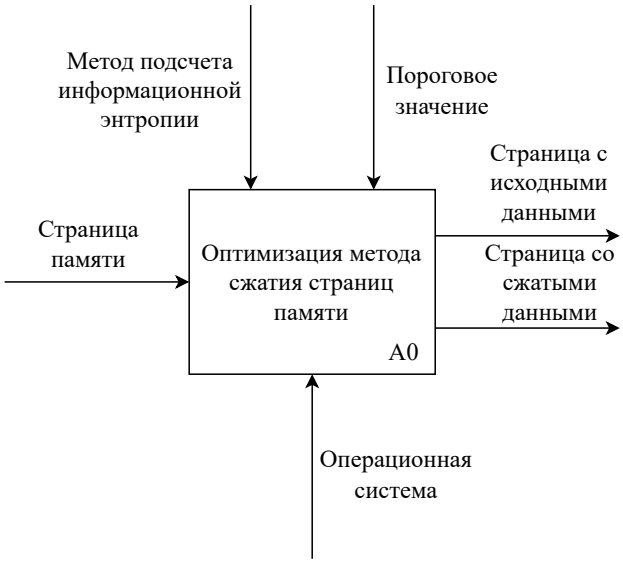

Рисунок 1.1 – IDEF0-диаграмма нулевого уровня

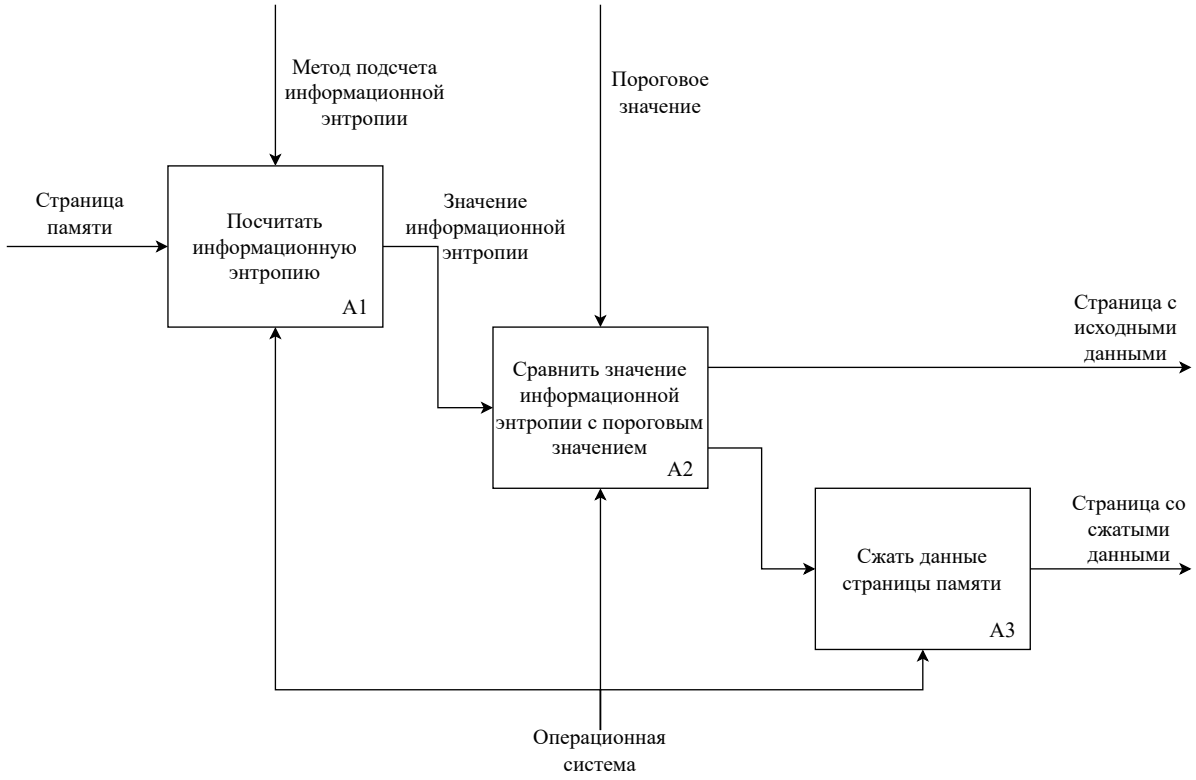

Рисунок 1.2 – IDEF0-диаграмма первого уровня

Входными данными является страница памяти, которая представляет собой вектор $a = (a_1\ a_2\ \ldots\ a_N)$ размером N, равным размеру страницы памяти. При этом, $0 \leq a_i \leq 255$.

На принятие решения о выполнении сжатия влияет сравнение подсчитанного значения информационной энтропии с пороговым значением. Сжатие страниц памяти выполняется ядром операционной системы.

В зависимости от принятого решения выходными данными могут быть:

— страница памяти со сжатыми исходными данными, которая представляет собой вектор $b = (b_1\ b_2\ \ldots\ b_N)$ размером N, $0 \leq b_i \leq 255$;

— страница памяти с исходными данными, которая представляет собой вектор a.

1.2 Требования к программному обеспечению

Согласно формальному описанию постановки задачи программное обеспечение должно:

— вычислять информационную энтропию страницы оперативной памяти, которая является вектором $a = (a_1\ a_2\ \ldots\ a_N)$ размером N, равным размеру страницы памяти, $0 \leq a_i \leq 255$;

— если вычисленное значение меньше порогового, сжимать данные входной страницы память, то есть, получать вектор $b = (b_1\ b_2\ \ldots\ b_N)$ размером N, $0 \leq b_i \leq 255$;

— если вычисленное значение больше или равно пороговому, данные входной страницы памяти не должны изменяться;

— выполняться как часть ядра операционной системы.

1.3 Выбор операционной системы

Программное обеспечение должно выполняться как часть ядра операционной системы, удовлетворяющей следующим критериям:

— поддержка сжатия страниц оперативной памяти;

— открытый исходный код.

Для сравнения, приведенного в таблице 1.1, были выбраны системы, занимающие наибольшие доли рынка операционных систем [3].

Таблица 1.1 – Сравнение операционных систем

Операционная система	**Поддержка сжатия**	**Открытый исходный код**	**Доля рынка (%)**
Windows	+	-	63.13
OS X	+	-	17.78
Linux	+	+	2.83
FreeBSD	+	+	0.01

В результате сравнения была выбрана операционная системы Linux, так как она удовлетворяет выделенным критериям и занимает долю рынка большую, чем операционная система FreeBSD. Возможность сжатия страниц оперативной памяти в ядре Linux предоставляют загружаемые модули zram и zswap.

1.4 Выбор модуля сжатия

При загрузке модуля zram в оперативной памяти создается блочный специальный файл, который моделирует диск. После загрузки модуля страницы, попадаемые на вход моделируемому диску, сжимаются и хранятся сжатыми. Таким образом, страницы находятся в оперативной памяти в сжатом виде.

Модуль zram обладает следующими преимуществами [4]:

— сокращение использования памяти за счет сжатия страниц;

— высокая скорость операций чтения и записи, так как блочный специальный файл создается в оперативной памяти;

— возможность выбора алгоритма сжатия, предоставляемого Crypto API [5];

— возможность реализации хранения временных файлов в сжатом виде;

— возможность реализации кэширования страниц памяти;

— возможность использования в качестве устройства подкачки;

— возможность записи несжимаемых страниц в резервное хранилище;

— возможность повторного сжатия с другим алгоритмом сжатия;

— многопоточное сжатие.

Модуль zswap работает следующим образом:

— перехватывает страницу, находящуюся в процессе выгрузки на устройство подкачки;

— осуществляет попытку записать данную страницу в сжатом виде в динамически создаваемый пул оперативной памяти;

— если размер пула сжатых страниц достигает предела, страницы вытесняются на устройство подкачки с использованием алгоритма LRU.

Модуль zswap обладает следующими преимуществами [6]:

— сокращение использования памяти за счет сжатия страниц;

— высокая скорость операций чтения и записи, так как сжатый пул находится в оперативной памяти;

— возможность выбора алгоритма сжатия, предоставляемого Crypto API [5].

В результате обзора загружаемых модулей сжатия ядра Linux был выбран модуль zram в связи с его преимуществами и тем, что для работы модуля zswap нужно устройство подкачки, на которое при полном заполнении пула вытесняются страницы памяти.

1.5 Выбор инструментов реализации

В качестве языка программирования был выбран язык C [7], так как большая часть исходного кода ядра операционной системы Linux и всех ее модулей написана на данном языке программирования.

Для сборки программного обеспечения выбрана утилита GNU make [8], так как с помощью данной утилиты осуществляется сборка загружаемых модулей ядра.

1.6 Структура программного обеспечения

Структура загружаемого модуля ядра zram включает в себя следующие части:

— модуль блочного устройства, который выполняет функции создания, настройки и удаления дисков, обработки операций записи и чтения страниц и получения статистики;

— модуль сжатия, который выполняет функции сжатия и восстановления данных, а также настройку этих операций.

Функция сжатия данных, предоставляемая модулем сжатия, вызывается в модуле блочного устройства во время обработки записи страницы на диск, как представлено на рисунке 1.3.

Программное обеспечение разрабатывалось как модификация загружаемого модуля ядра zram операционной системы Linux, поэтому имеет структуру, показанную на рисунке 1.4.

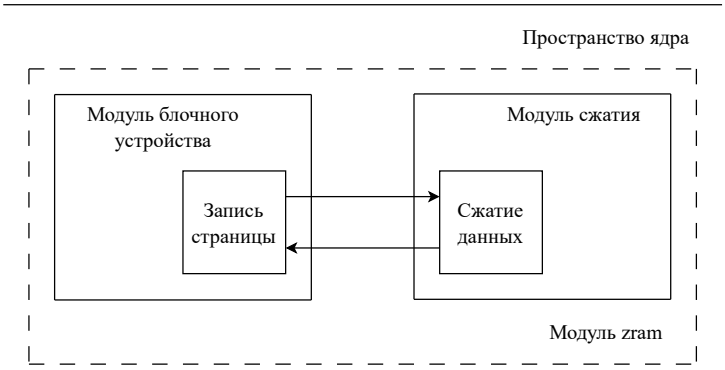

Рисунок 1.3 – Структура модуля zram

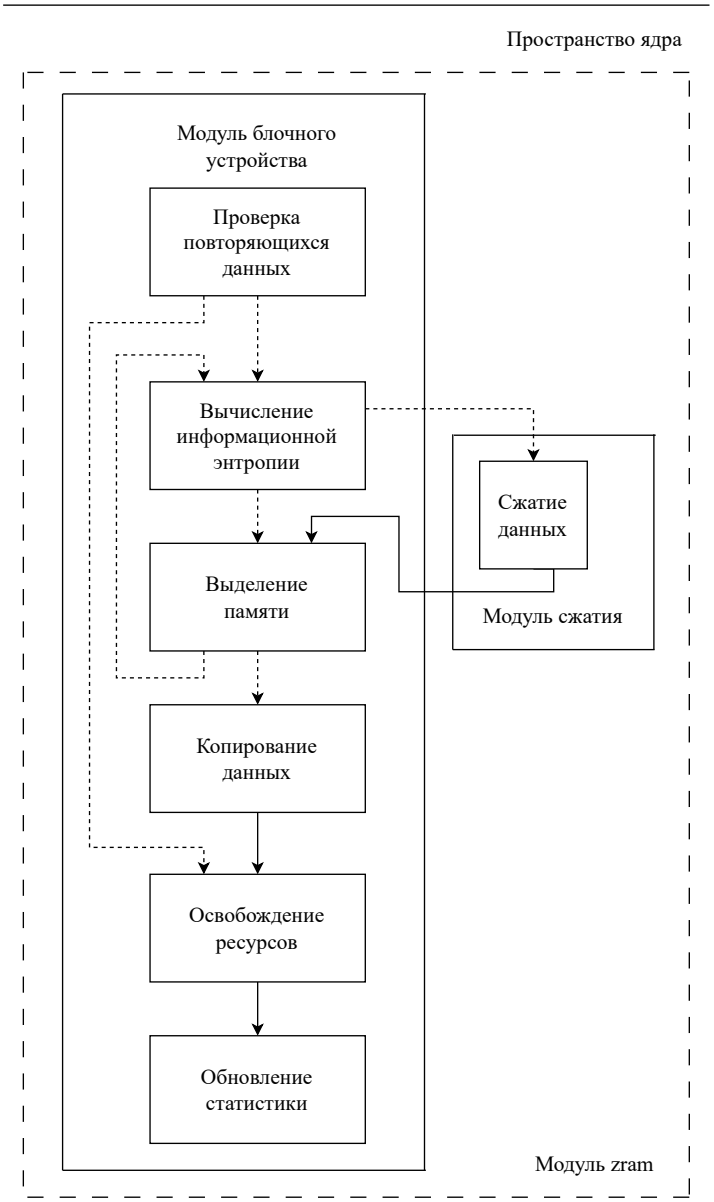

Рисунок 1.4 – Структура программного обеспечения

1.7 Детали реализации

В листинге 1.1 показана реализация метода скользящего окна для подсчета информационной энтропии. Размер страницы в байтах в ядре Linux определяется константой PAGE_SIZE [9].

Листинг 1.1 – Реализация метода скользящего окна для подсчета информационной энтропии

```
#define BYTES_NUM 256

static inline u32 ilog2_w(u64 n)
{
    return ilog2(n * n * n * n);
}

static inline s32 get_sw_entropy(const u8 *src)
{
    u16 bytes_frequency[BYTES_NUM] = { 0 };
    u32 i;

    for (i = 0; i < PAGE_SIZE; ++i) {
        bytes_frequency[src[i]]++;
    }

    u32 page_size = ilog2_w(PAGE_SIZE);
    s32 entropy = 0;

    for (i = 0; i < BYTES_NUM; ++i) {
        s32 probability = bytes_frequency[i];

        if (probability > 0) {
            entropy += probability * (page_size -
               ilog2_w((u64)probability));
        }
    }

    return entropy;
}
```

В листинге 1.2 приведена часть реализации записи страницы на диск в модуле блочного устройства.

Листинг 1.2 – Часть реализация записи страницы на диск

```
static int __zram_bvec_write(struct zram *zram, struct bio_vec *bvec,
                 u32 index, struct bio *bio)
{
    ...
```

```
    unsigned int comp_len = 0;
    void *src, *dst, *mem;
    struct zcomp_strm *zstrm;
    struct page *page = bvec->bv_page;
    ...

compress_again:
    zstrm = zcomp_stream_get(zram->comp);
    src = kmap_atomic(page);

    if (get_sw_entropy((const u8 *)src) < ENTROPY_THRESHOLD)
        ret = zcomp_compress(zstrm, src, &comp_len);
    else
        comp_len = PAGE_SIZE;
    ...
}
```

1.8 Конфигурация программного обеспечения

Для сборки, запуска и настройки программного обеспечения был написан make-файл, код которого представлен в листинге 1.3.

Листинг 1.3 – Конфигурационный файл

```
KERNEL_VERSION = $(shell uname -r)
MODULE_DIR = $(shell pwd)

OBJS = zcomp.o zram_drv.o
TARGET = zram

obj-m := $(TARGET).o
$(TARGET)-objs := $(OBJS)

all:
    $(MAKE) -C /lib/modules/$(KERNEL_VERSION)/build M=$(MODULE_DIR) modules

$(TARGET).o: $(OBJS)
    $(LD) -r -o $@ $(OBJS)

clean:
    @rm -f *.o .*.cmd .*.flags *.mod.c *.order
    @rm -f .*.*.cmd *~ *.*~
    @rm -fR .tmp*
    @rm -rf .tmp_versions

distclean:
    @rm -f *.ko *.symvers *.mod

```

```
load:
    sudo insmod $(TARGET).ko

info:
    sudo lsmod | grep $(TARGET)

log:
    sudo dmesg | grep $(TARGET)

unload:
    sudo rmmod $(TARGET).ko


add-disk:
    cat /sys/class/zram-control/hot_add

get-comp-algorithm:
    cat /sys/block/zram$(id)/comp_algorithm

set-comp-algorithm:
    echo $(comp-algoritm) > /sys/block/zram$(id)/comp_algorithm

set-disksize:
    echo $(disksize) > /sys/block/zram$(id)/disksize

memory-stat:
    cat  /sys/block/zram$(id)/mm_stat

reset-disk:
    echo 1 > /sys/block/zram$(id)/reset

remove-disk:
    echo $(id) > /sys/class/zram-control/hot_remove
```

Конфигурационный файл предоставляет следующие возможности для работы с программным обеспечением:

— получить исполняемый файл программного обеспечения — загружаемый модуль, с помощью команды make;

— загрузить полученный модуль с помощью команды make load;

— проверить, что модуль загружен, с помощью команды make info;

— получить сообщения модуля в системном журнале с помощью команды make info;

— выгрузить модуль с помощью команды make unload.

Средствами конфигурационного файла можно выполнять следующие действия с диском zram:

1. Добавить диск с помощью команды make add-disk, в результате выполнения которой на экран будет выведен идентификатор добавленного устройства или код ошибки.

2. Получить список доступных алгоритмов сжатия для диска с идентификатором i с помощью команды make get-comp-algorithm id=i.

3. Установить алгоритм сжатия a для диска с идентификатором i с помощью команды make set-comp-algorithm comp-algoritm=a id=i.

4. Установить размер s диска с идентификатором i с помощью команды make set-disksize disksize=s id=i. Для указания единиц измерения используются следующие постфиксы: K — размер в килобайтах, M — размер в мегабайт, G — размер в гигабайтах. При отсутствии постфикса устанавливается размер в байтах.

5. Получить статистику памяти диска с идентификатором i с помощью команды make memory-stat id=i.

6. Освободить память, выделенную для диска с идентификатором i, и сбросить его размер диска до нуля с помощью команды make reset-disk id=i.

7. Удалить диск с идентификатором i с помощью команды make remove-disk id=i.

1.9 Пример работы программного обеспечения

На рисунке 1.5 показана часть статистики обработки программным обеспечением страниц файлов с расширениями .h и .c.

ID	ENTROPY	ENTROPY_TIME	COMPRESSION_TIME	HANDLING_TIME	SIZE	COMPRESSED_SIZE	COMPRESSION_RATIO	SPACE_SAVINGS
1	85396	5010	261255	294568	4096	1622	2.525277435265105	0.60400390625
2	82762	9848	293666	333962	4096	1277	3.207517619420517	0.688232421875
3	83355	5440	345685	380491	4096	1669	2.4541641701617736	0.592529296875
4	87674	5070	244743	278718	4096	1580	2.5924050632911393	0.6142578125
5	83569	4840	242890	272406	4096	1586	2.5825977301387137	0.61279296875
6	87133	4959	235797	265593	4096	1675	2.4453731343283582	0.591064453125
7	90488	4970	174731	204227	4096	1347	3.0408314773570897	0.671142578125
8	88184	4329	181805	210579	4096	1337	3.06357516828721	0.673583984375
9	88687	4409	190731	219666	4096	1615	2.5362229102167184	0.605712890625

Рисунок 1.5 – Часть статистики обработки страниц файлов с расширениями .h и .c

Данные статистики оформлены в виде таблицы со следующими столбцами:

— ID — идентификатор страницы, соответствующий номеру страницы в порядке их обработки;

— ENTROPY — значение информационной энтропии исходных данных страницы;

— ENTROPY_TIME — время вычисления информационной энтропии в наносекундах;

— COMPRESSION_TIME — время сжатия данных страницы в наносекундах;

— HANDLING_TIME — время обработки страницы в наносекундах, равное времени выполнения функции записи страницы на диск;

— SIZE — размер исходных данных в байтах;

— COMRESSED_SIZE — размер сжатых данных в байтах;

— COMPRESSION_RATIO — отношение объема исходных данных к объему сжатых данных;

— SPACE_SAVINGS — уменьшение размера файла по сравнению с размером несжатого файла.

ЗАКЛЮЧЕНИЕ

В результате выполнения преддипломной практики было разработано программное обеспечение, демонстрирующее практическую осуществимость спроектированной в ходе выполнения выпускной квалификационной работы оптимизации метода сжатия страниц памяти с использованием подсчета информационной энтропии.

Были решены следующие задачи:

— формально описана постановка задачи оптимизации метода сжатия страниц памяти;

— выделены требования к программному обеспечению,

— выбраны средства программной реализации;

— описана структура программного обеспечения;

— разработано программное обеспечение, реализующее оптимизацию метода сжатия страниц памяти с использованием подсчета информационной энтропии.

— описана конфигурация программного обеспечения.

Таким образом, поставленная цель была достигнута.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Silberschatz A.*, *Galvin P. B.*, *Gagne G.* Operating System Concepts. — 10th. — Hoboken, NJ: Wiley, 2018. — 1278 с.
2. *Cheng X.*, *Li Z.* How does Shannon's source coding theorem fare in prediction of image comrpession ratio with current algorithms? // International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. — 2020. — 1313—1319 с.
3. Desktop Operating System Market Share Worldwide [Электронный ресурс]. — Режим доступа URL: `https://gs.statcounter.com/os-market-share/desktop/worldwide` (Дата обращения: 07.05.2023).
4. zram: Compressed RAM-based block devices [Электронный ресурс]. — Режим доступа URL: `https://docs.kernel.org/admin-guide/blockdev/zram.html?highlight=zram` (Дата обращения: 07.05.2023).
5. Kernel Crypto API Interface Specification [Электронный ресурс]. — Режим доступа URL: `https://docs.kernel.org/crypto/intro.html` (Дата обращения: 07.05.2023).
6. zswap [Электронный ресурс]. — Режим доступа URL: `https://docs.kernel.org/admin-guide/mm/zswap.html?highlight=zswap` (Дата обращения: 07.05.2023).
7. The GNU C Reference Manual [Электронный ресурс]. — Режим доступа URL: `https://www.gnu.org/software/gnu-c-manual/gnu-c-manual.html` (Дата обращения: 07.05.2023).
8. GNU make [Электронный ресурс]. — Режим доступа URL: `https://www.gnu.org/software/make/manual/make.html#Simple-Makefile` (Дата обращения: 07.05.2023).
9. *Fox R.* Linux with Operating System Concepts. — 2th. — Boca Raton: CRC Press, 2022. — 100 с.