



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 «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии»

РАСЧЕТНО-ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

К ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЕ

НА ТЕМУ:

«Оптимизация метода сжатия страниц памяти с использованием подсчета информационной энтропии»

Студент ИУ7-83Б
(Группа)

(Подпись, дата)

Р. Р. Хамзина
(И. О. Фамилия)

Руководитель ВКР

(Подпись, дата)

А. А. Оленев
(И. О. Фамилия)

2023 г.

РЕФЕРАТ

Расчетно-пояснительная записка 44 с., 17 рис., 3 табл., 33 источн., 1 прил.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	5
1 Аналитический раздел	6
1.1 Управление памятью в операционной системе	6
1.1.1 Виртуальная память	6
1.1.2 Подкачка страниц	7
1.2 Сжатие данных	8
1.2.1 Сжатие памяти	9
1.2.2 Показатели качества сжатия	11
1.3 Информационная энтропия	12
1.3.1 Свойства информационной энтропии	13
1.3.2 Связь со сжатием данных	14
1.4 Методы подсчета информационной энтропии	15
1.4.1 Метод скользящего окна	15
1.4.2 Биномиальный метод	16
1.4.3 Сравнение существующих методов подсчета	18
1.5 Постановка задачи	19
1.6 Сравнение операционных систем	20
1.7 Сжатие памяти в ядре Linux	21
1.7.1 Модуль zram	21
1.7.2 Модуль zswap	23
2 Конструкторский раздел	25
2.1 Функциональная модель оптимизации метода сжатия страниц	25
2.2 Требования к разрабатываемому программному обеспечению	26
2.3 Структура разрабатываемого программного обеспечения	26
2.4 Выбор типов и структур данных	28
2.5 Описание оптимизации метода сжатия страниц	28
2.6 Проектирование тестирования программного обеспечения	31
3 Технологический раздел	34
3.1 Выбор средств программной реализации	34
3.2 Детали реализации	34

3.3	Конфигурация программного обеспечения	35
3.4	Тестирование разработанного программного обеспечения	37
4	Исследовательский раздел	40
	ЗАКЛЮЧЕНИЕ	41
	СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	44

ВВЕДЕНИЕ

1 Аналитический раздел

1.1 Управление памятью в операционной системе

Для хранения данных запущенных в операционной системе процессов, находящихся в состоянии выполнения, ожидания или блокировки, используется оперативная память. Устройство, предназначенное для записи, считывания и хранения инструкций и данных процессов, называется оперативным запоминающим устройством (ОЗУ) [1]. В многозадачных системах память должна быть распределена для размещения нескольких процессов. Для управления памятью в системе используется абстракция адресного пространства. У каждого процесса имеется собственное адресное пространство — набор адресов, который может быть использован процессом для обращения к памяти [2]. Если объем оперативной памяти, необходимый для размещения данных всех процессов, превышает объем ОЗУ, то возникает перегрузка памяти. Одним из способов решения проблемы является виртуальная память.

1.1.1 Виртуальная память

Механизм виртуальной памяти предполагает разделение логической памяти и физической памяти. У каждого процесса имеется собственное виртуальное адресное пространство. При использовании страничной организации памяти, виртуальное адресное пространство делится на блоки заданного размера, называемые страницами. Физическая память делится на блоки заданного размера, называемые страничным блоком или кадром. Страницы виртуального адресного пространства и страничные блоки имеют одинаковый размер, который определяется аппаратным обеспечением. Значение размера представляет собой степень двойки и изменяется от четырех килобайт до одного гигабайта [3].

При использовании виртуальной памяти процессы оперируют виртуальными адресами. Виртуальный адрес — это адрес, присвоенный местоположению в виртуальной памяти, который позволяет обращаться к данному местоположению так, как если бы это была часть физической памяти [2]. Отображение виртуального адреса на физический адрес памяти выполняется диспетчером памяти с использованием таблиц страниц, как показано на рисунке 1.1.

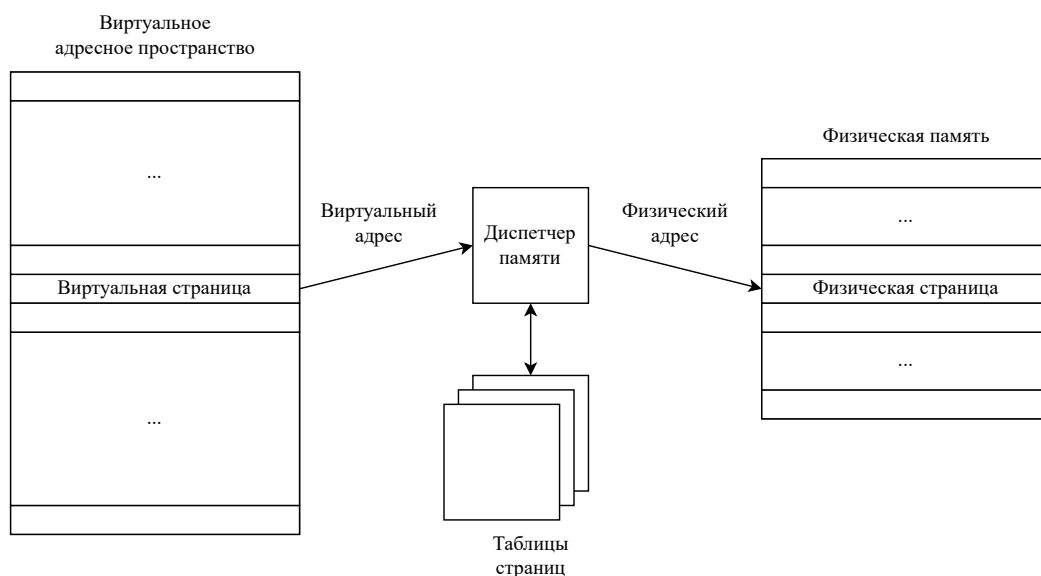


Рисунок 1.1 – Схема работы виртуальной памяти

Одним из основных преимуществ использования виртуальной памяти является то, что объем памяти, размещающий данные процессов не ограничен объемом доступной физической памяти. В системах с виртуальной памятью используется подкачка страниц по запросу.

1.1.2 Подкачка страниц

Данные процесса или их часть можно выгружать из оперативной памяти в резервное хранилище, а затем возвращать в память для продолжения выполнения. Резервное хранилище представляет собой вторичное хранилище, разделенное на блоки определенного размера, которые имеют тот же размер, что и страничные кадры. Таким образом, неиспользуемая страница памяти может быть перемещена в резервное хранилище, и загружена в оперативную память при необходимости. Этот принцип работы получил название подкачка страниц, при этом вторичное хранилище называется устройством подкачки, а его раздел, используемый для хранения данных — пространством подкачки [3].

Присутствие страницы в оперативной памяти отслеживается с помощью бита присутствия-отсутствия [2]. Если процесс попытается получить доступ к отсутствующей в памяти странице, возникает системное прерывание (page fault). Операционная система находит свободный страничный кадр или выбирает редко используемую страницу и сбрасывает ее содержимое во вторичное

хранилище. Затем она перемещает нужную физическую страницу в свободный страничный кадр. Схема работы подкачки страниц при возникновении page fault приведена на рисунке 1.2.

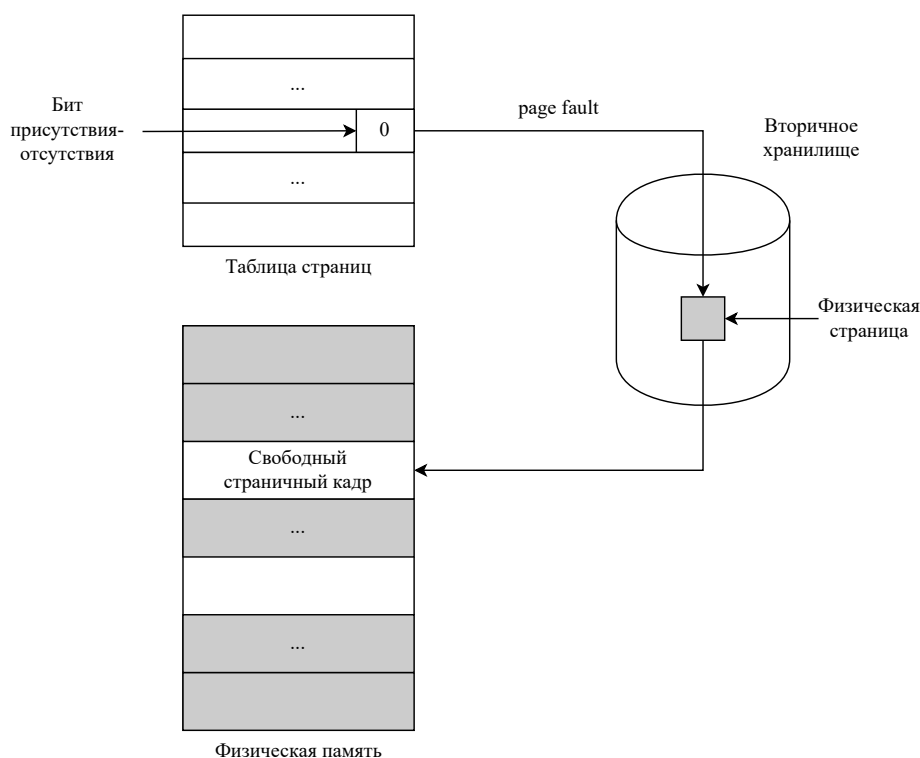


Рисунок 1.2 – Схема работы подкачки страниц при возникновении page fault

Подкачка страниц позволяет загружать только требуемые для выполнения данные, но для переноса страниц между вторичным хранилищем и оперативной памятью требуется время. Частое перемещение снижает производительность работы подсистемы памяти. Для решения этой проблемы используется сжатие памяти [4].

1.2 Сжатие данных

Неслучайные данные имеют некоторую структуру. Наличие у данных структуры, которую можно использовать для уменьшения их размера путем достижения такого представления данных, в котором никакая структура не выделяется, называют избыточностью. Сжатие данных — это процесс преобразования исходных данных в их компактную форму путем распознавания и использования избыточности данных [5]. Процесс сжатия состоит из двух этапов:

1. Этап моделирования, который включает в себя распознавание избыточности для построения модели. Модель представляет собой набор данных и правил, используемых для обработки входных символов.
2. Этап кодирования данных с использованием модели.

Описанные этапы сжатия данных представлены на рисунке 1.3.

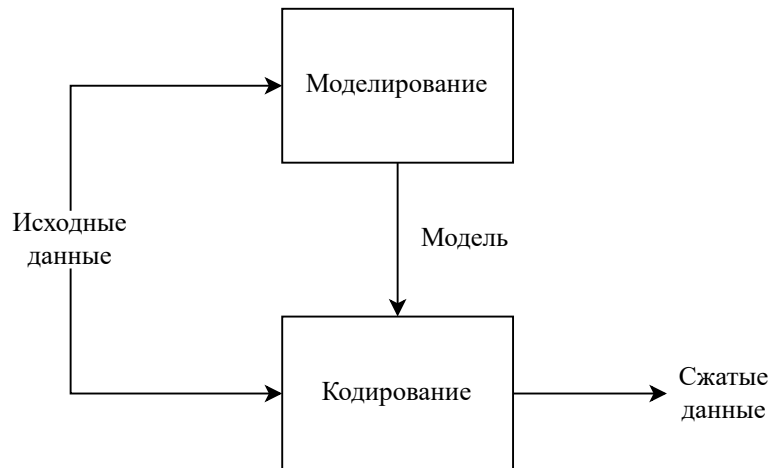


Рисунок 1.3 – Этапы сжатия данных

В результате сжатия исходных данных X получается их представление $X_{\text{сж}}$. При восстановлении сжатые данные $X_{\text{сж}}$ преобразуются в представление Y . На основании требований к восстановлению выделяют [6]:

- сжатие данных без потерь, при котором $Y = X$ (обратимое сжатие);
- сжатие данных с потерями, при котором $Y \neq X$ (необратимое сжатие).

Схемы обратимого и необратимого сжатия данных показаны на рисунке 1.4.

1.2.1 Сжатие памяти

Сжатие памяти заключается в том, что несколько страничных кадров сжимаются, и их сжатые версии сохраняются в меньшем числе страничных кадров. Это позволяет системе сократить использование памяти, не прибегая к подкачке страниц. Если процессу требуются сжатые страницы, они восстанавливаются [3].

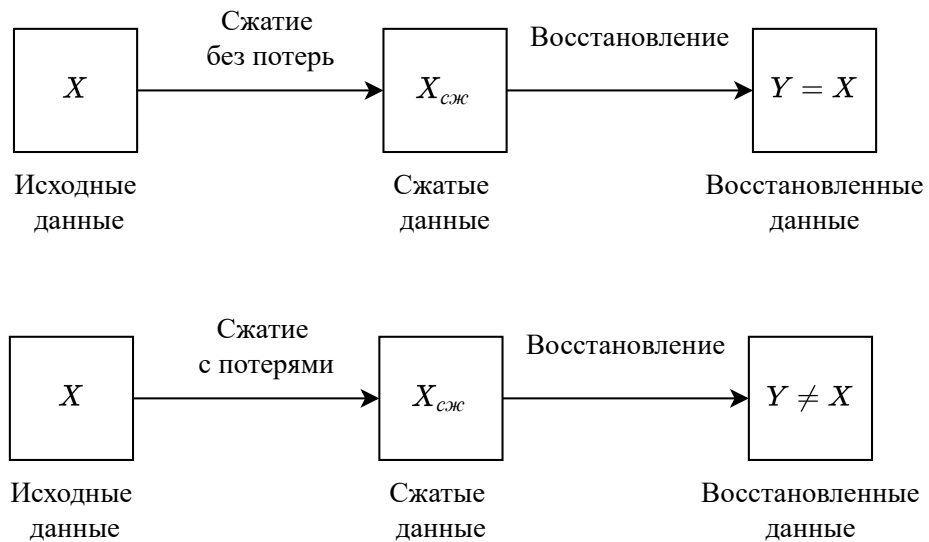


Рисунок 1.4 – Схемы обратимого и необратимого сжатия данных

На рисунке 1.5 приведена схема сжатия памяти: неиспользуемые страницы 2-5 хранятся в сжатом виде, в результате увеличивается объем свободной памяти.



Рисунок 1.5 – Схема сжатия памяти

Выделяют следующие требования к реализации сжатия памяти [7]:

- при запросе на чтение данные должны восстанавливаться без потерь;
- операции сжатия и восстановления должны иметь низкую временную

задержку и высокое качество сжатия.

1.2.2 Показатели качества сжатия

Для измерения качества сжатия используют следующие характеристики: коэффициент сжатия, сила сжатия (compression gain) и экономия пространства (space savings).

Коэффициент сжатия определяет средний объем памяти, необходимый для хранения сжатых данных, и вычисляется следующим образом:

$$K_{\text{сж}} = \frac{L_{\text{исх}}}{L_{\text{сж}}}, \quad (1.1)$$

где $L_{\text{исх}}$ — объем исходных данных, $L_{\text{сж}}$ — объем сжатых данных. Высокий коэффициент сжатия говорит о высоком качестве сжатия. У сжатия низкого качества могут быть следующие причины [8]:

- исходные данные являются сжатыми;
- исходные данные являются зашифрованными;
- исходные данные не содержат повторяющихся шаблонов.

Сила сжатия определяется так:

$$CG = 100 \cdot \log_e \frac{L_{\text{исх}}}{L_{\text{сж}}} = 100 \cdot \log_e K_{\text{сж}}. \quad (1.2)$$

Экономия пространства показывает уменьшение размера файла по сравнению с размером несжатого файла. Ее можно получить по следующей формуле:

$$SS = 1 - \frac{L_{\text{сж}}}{L_{\text{исх}}}. \quad (1.3)$$

Например, если размер исходных данных равен десяти мегабайтам, а размер сжатых данных равен трем мегабайтам, то экономия пространства указывает на то, что 70% пространства сохраняется благодаря сжатию.

Оценить показатели качества сжатия можно с помощью информационной энтропии.

1.3 Информационная энтропия

Под информацией понимают сведения, которые являются объектом хранения, передачи и обработки [9]. Формой представления информации является сообщение. Физическую величину, отображающую сообщение, называют сигналом. Передача информации осуществляется следующим образом [10]:

1. Источник информации создает случайное сообщение. В теории информации любой источник информации является стохастическим, его можно описать измеряемыми вероятностными категориям.
2. Сообщение поступает в систему передачи, в которой выполняется кодирование — преобразование сообщения с целью согласования источника информации с каналом связи для увеличения скорости передачи информации или обеспечения заданной помехоустойчивости [9]. Кодирование состоит из шифрования, сжатия и защиты от шума, в результате которых формируется сигнал.
3. Сигнал проходит через канал — среду передачи информации [10]. В канале могут возникать помехи, создаваемые источником шума.
4. Сигнал подается на вход системе приема, которая выполняет декодирование — восстановление исходного сообщения [9].
5. Исходное сообщение передается получателю.

Описанная схема передачи информации представлена на рисунке 1.6.

Сообщение содержит сведения о некоторой физической системе X , которая случайным образом может перейти в какое-либо состояние x_i из конечного множества состояний x_1, x_2, \dots, x_n с вероятностями p_1, p_2, \dots, p_n , где $n \in \mathbb{N}$, $p_i = P(X \sim x_i)$ и $\sum_{i=1}^n p_i = 1$. То есть, для такой системы существует степень неопределенности, которая описывается числом ее возможных состояний и их вероятностями. Сведения из принятого сообщения тем ценнее, чем больше была неопределенность системы до получения сообщения. Специальную характеристику, используемую в качестве меры неопределенности системы, называют информационной энтропией [11]. Информационная энтропия конечной вероятностной схемы определяется по формуле Шеннона:

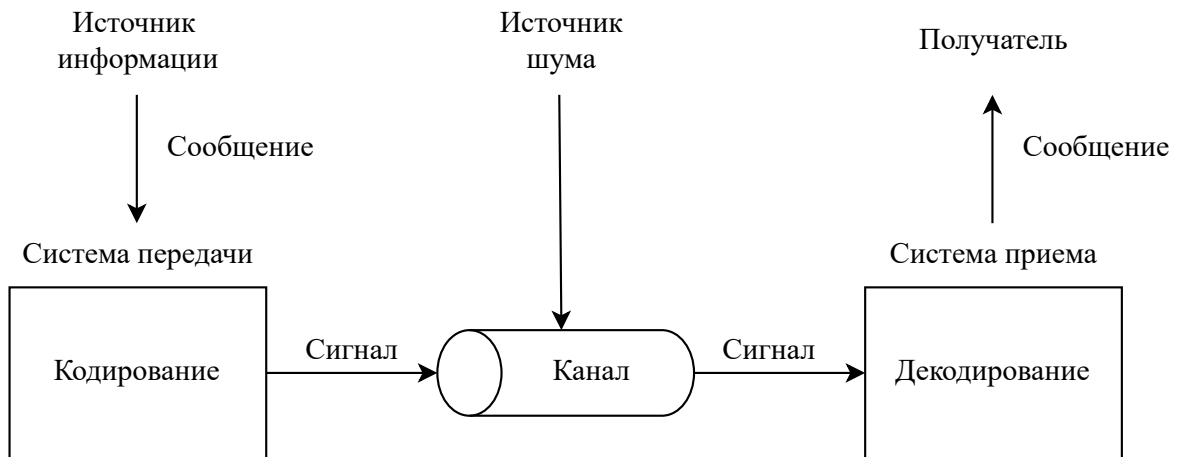


Рисунок 1.6 – Схема передачи информации

$$H(X) = - \sum_{i=1}^n (p_i \cdot \log_a p_i), \quad (1.4)$$

где $p_i \in [0, 1]$, $a > 1$.

Основание логарифма определяет единицы измерения информационной энтропии: при $a = 2$ энтропия измеряется в битах, при $a = 3$ — в тритах, при $a = e$ — в натах.

1.3.1 Свойства информационной энтропии

Информационная энтропия обладает следующими свойствами [12]:

1. Энтропия всегда неотрицательна. Значения $\log_a p_i$ в формуле (1.4) принимают неположительные значения, так как $p_i \in [0, 1]$. Поэтому

$$H(X) = - \sum_{i=1}^n (p_i \cdot \log_a p_i) \geq 0. \quad (1.5)$$

2. Энтропия равна нулю, если состояние системы в точности известно заранее. Если известно состояние x_k , в которое перейдет система X , то вероятность этого состояния p_k равна единице, вероятности других состояний равны нулю. Тогда

$$p_k \cdot \log_a p_k = 1 \cdot \log_a 1 = 1 \cdot 0 = 0. \quad (1.6)$$

В связи с тем, что $\lim_{p \rightarrow 0} (p \cdot \log_a p) = 0$, другие слагаемые суммы в формуле (1.4) равны нулю. В этом случае

$$H(X) = - \sum_{i=1}^n (p_i \cdot \log_a p_i) = 0. \quad (1.7)$$

3. Энтропия принимает наибольшее значение при условии, что все состояния равновероятны, то есть, $p_1 = p_2 = \dots = p_n = \frac{1}{n}$. Тогда

$$H(X) = - \sum_{i=1}^n (p_i \cdot \log_a p_i) = - \sum_{i=1}^n \left(\frac{1}{n} \cdot \log_a \frac{1}{n} \right) = - \log_a \frac{1}{n} = \log_a n. \quad (1.8)$$

1.3.2 Связь со сжатием данных

Данные, обладающие предсказуемой структурой, сокращают неопределенность системы меньше, чем сведения, в которых никакая структура не выделяется. Так как информационная энтропия является мерой неопределенности системы, то данные с выделяемой структурой, имеют низкое значение энтропии. Сведения, в которых закономерности не определяются, имеют высокое значение энтропии [13]. Так, чем меньше избыточность данных, тем выше значение их энтропии. То есть, информационная энтропия сжатых данных выше, чем ее значение до сжатия.

Согласно теореме Шеннона о кодировании источника шифрования для источника сообщений с энтропией H невозможно сжать данные без потери информации так, что среднее число бит на символ будет меньше, чем H . Это означает, что невозможно сжать сообщение без потери данных до размера меньше, чем энтропия источника. Таким образом, на основании информационной энтропии исходных данных определяется теоретическая граница показателей качества сжатия [14].

В связи с применением операций сложения, умножения и логарифмирования при вычислении энтропии по формуле (1.4), число которых растет с увеличением объема данных, встает задача выбора метода подсчета. Исполь-

зование метода влияет на скорость и время определения информационной энтропии.

1.4 Методы подсчета информационной энтропии

1.4.1 Метод скользящего окна

При решении задач, связанных со сжатием, данные X представляют собой массив байтов размером N [15]. Байт состоит из восьми битов, каждый из которых кодирует одно из значений множества $\{0, 1\}$. Поэтому один байт может принимать значения из интервала от 0 до 255 включительно в десятичной системе счисления.

В методе скользящего окна под окном понимают рассматриваемую на текущем этапе подпоследовательность данных размером n [16]. При подсчете энтропии данным методом необходима дополнительная память размером 2^n для хранения числа вхождений подпоследовательностей данных. Так как с увеличением размера окна, растет объем дополнительной памяти, в качестве окна выбирается минимально адресуемая единица памяти, которой является байт [17]. В связи с тем, что на каждом этапе окно смещается на следующие восемь битов, как показано на рисунке 1.7, оно называется скользящим.

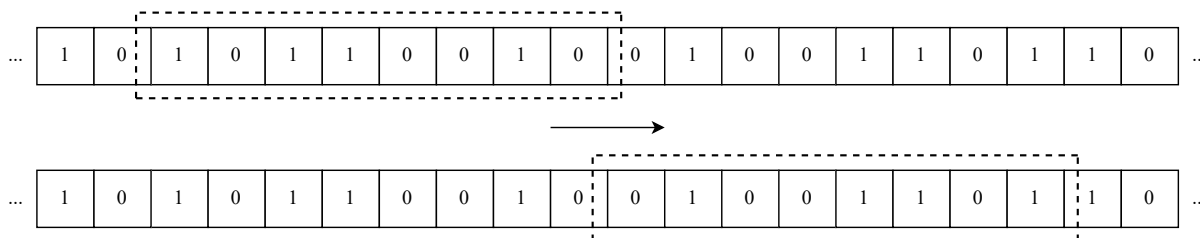


Рисунок 1.7 – Проход по массиву байтов методом скользящего окна

Вычисление информационной энтропии методом скользящего окна включает в себя два шага:

1. Для каждого возможного значения байта подсчитывается число его вхождений k_i в массив байтов, где $i = \overline{0, 255}$.
2. С учетом того, что вероятность появления байта в массиве $p_i = \frac{k_i}{N}$, информационная энтропия вычисляется по следующей формуле:

$$H(X) = - \sum_{i=0}^{255} (p_i \cdot \log_2 p_i). \quad (1.9)$$

Так как первый шаг метода предполагает проход по массиву байтов размером N , а второй шаг — проход по массиву числа вхождений подпоследовательностей данных размером 2^n , временная сложность метода скользящего окна — $O(N + 2^n)$.

Информационная энтропия, подсчитанная методом скользящего окна, принимает значения из интервала $[0, 8]$ битов. При этом согласно свойству 2 из подраздела 1.3.1 информационная энтропия принимает нулевое значение в случае, когда массив данных состоит из одинаковых байтов, и в соответствии со свойством 3 из подраздела 1.3.1 максимальное значение, равное восьми, если все байты в массиве различны.

1.4.2 Биномиальный метод

В биномиальном методе вычисления информационной энтропии [18] данные X рассматриваются как последовательность сообщений, генерируемых бернуллиевским источником — источником информации, порождающим символы из алфавита $\{0; 1\}$ с вероятностями $1 - p$ и p соответственно, причем $p \in (0, 1)$ и может быть неизвестно [19]. То есть, сообщение представляет собой последовательность битов длины n .

Вероятность того, что сообщение содержит k единиц, где $k = \overline{0, n}$, вычисляется следующим образом:

$$P_k = p^k \cdot (1 - p)^{(n-k)}. \quad (1.10)$$

Количество возможных сообщений, содержащих k единиц, определяется как биномиальный коэффициент:

$$C_n^k = \frac{n!}{k! \cdot (n - k)!}. \quad (1.11)$$

В связи с тем, что k может принимать значения из интервала $[0, n]$, число биномиальных коэффициентов, подсчитанных по формуле (1.11), равно $n + 1$. Это означает, что сообщения можно разбить на $n + 1$ классов эквивалентности. Тогда информационная энтропия вычисляется так:

$$H(X) = - \sum_{k=0}^n (C_n^k \cdot P_k \cdot \log_2 P_k). \quad (1.12)$$

В соответствии с формулой (1.12) биномиальный метод подсчета информационной энтропии состоит из следующих этапов:

1. Определяется вероятность p появления единицы в сообщениях.
2. Исходные сообщения разбиваются на $n + 1$ классов эквивалентности, сообщения которых содержат $k = \overline{0, n}$ единиц.
3. Для каждого класса эквивалентности рассчитывается биномиальный коэффициент C_n^k .
4. Вычисляются вероятности P_k по формуле (1.10).
5. Суммируются произведения биномиальных коэффициентов C_n^k , вероятностей P_k и логарифмов вероятностей P_k для всех $n + 1$ значений k .

Схема определения биномиальных коэффициентов C_n^k и вероятностей P_k приведена на рисунке 1.8.

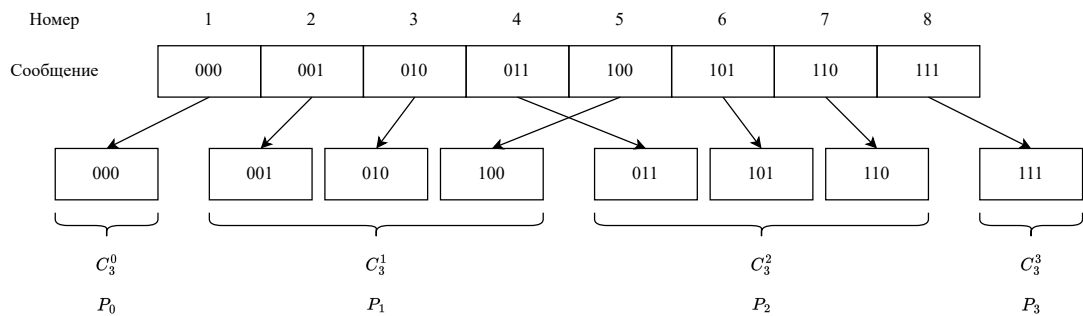


Рисунок 1.8 – Определение биномиальных коэффициентов C_n^k и вероятностей P_k

Этап определения вероятности p появления единицы в сообщениях подразумевает проход по массиву байтов размером N , в котором с единицей сравнивается каждый бит одного байта. Последний этап состоит из $n + 1$ цикла вычисления. При определении вероятности того, что сообщение содержит k единиц, требуются возведение в степень k вероятности появления единицы p и возведение в степень $n - k$ вероятности появления нуля $1 - p$.

Тогда временная сложность биномиального метода подсчета информационной энтропии — $O(N + n^2)$.

В биномиальном методе определяются вероятности появления не подпоследовательности, а единицы в подпоследовательностях. За счет разделения исходной последовательности на классы эквивалентности сокращается число операций сложения.

При рассмотрении бернуллиевского источника информации предполагается, что вероятность появления единицы в подпоследовательности не зависит от вероятностей появления нуля или единицы в битах предыдущей подпоследовательности. При наличии такой зависимости вычисленное значение энтропии будет завышено. Так как рассматриваемое представление данных — последовательность битов, то вероятности появления каждого значения бита независимы.

Недостатком данного метода является необходимость вычисления факториала при определении биномиальных коэффициентов и возведения в степень при определении вероятностей появления сообщений. Для снижения времени подсчета биномиальных коэффициентов можно хранить их значения в дополнительном массиве. Так как для сообщения длины n количество требуемых для определения энтропии биномиальных коэффициентов равно $n + 1$, то для их хранения потребуется дополнительная память размером $n + 1$.

1.4.3 Сравнение существующих методов подсчета

Для сравнения методов подсчета информационной энтропии были выделены следующие критерии оценки:

- К1 — временная сложность;
- К2 — необходимость вычисления факториала;
- К3 — необходимость возведения в степень;
- К4 — возможность распараллеливания вычислений;
- К5 — объем требуемой дополнительной памяти.

Результаты сравнения представлены в таблице 3.1.

Таблица 1.1 – Сравнение методов подсчета информационной энтропии

Метод	K1	K2	K3	K4	K5
Скользящего окна	$O(N + 2^n)$	–	–	+	2^n
Биномиальный	$O(N + n^2)$	+	+	+	n

Таким образом, биномиальный метод подсчета информационной энтропии требует меньших вычислительных затрат по времени и по памяти. При решении задачи оценки качества сжатия помимо трудоемкости вычисления информационной энтропии важна корреляция полученного значения и показателей качества сжатия. При этом не так важна точность значения информационной энтропии.

1.5 Постановка задачи

Из проведенного в разделах 1.2-1.3 анализа сжатия и информационной энтропии можно сделать следующие выводы:

- сжатие и восстановление страниц оперативной памяти должно иметь низкую временную задержку и высокое качество сжатия;
- сжатие может обладать низким качеством, если исходные данные являются сжатыми, зашифрованными или не содержат повторяющихся шаблонов;
- значение информационной энтропии исходных данных позволяет определить теоретическую границу показателей качества сжатия.

Так, сжатие может иметь низкое качество, при этом на него будет тратиться время. Можно сократить временные затраты на сжатие, если определять теоретическую границу качества сжатия до его выполнения, и на основании оценки качества принимать решение, сжимать страницу или нет. Качество сжатия по исходным данным можно оценить с помощью информационной энтропии. Таким образом, необходимо провести оптимизацию сжатия страниц памяти, принимая решение о его выполнении на основании вычисленного значения информационной энтропии исходных данных.

Входными данными поставленной задачи является страница памяти, которая представляет собой вектор $a = (a_1 \ a_2 \ \dots \ a_N)$ 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 $b = (b_1 \ b_2 \ \dots \ b_N)$ размером N , $0 \leq b_i \leq 255$;
- страница памяти с исходными данными, которая представляет собой вектор a .

На рисунке 1.9 показана IDEF0-диаграмма нулевого уровня, формализующая поставленную задачу.

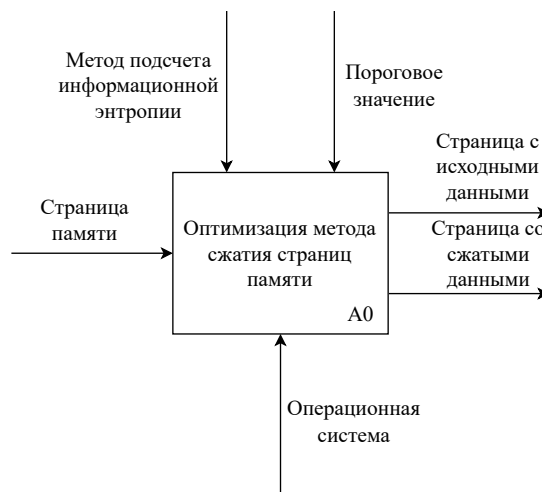


Рисунок 1.9 – IDEF0-диаграмма нулевого уровня

1.6 Сравнение операционных систем

Выполнение оптимизации метода сжатия страниц памяти должно проводиться на операционной системе, удовлетворяющей следующим критериям:

- поддержка сжатия страниц оперативной памяти [3];
- открытый исходный код [20].

Для сравнения, приведенного в таблице 1.2, были выбраны системы, занимающие наибольшие доли рынка операционных систем [21].

Таблица 1.2 – 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 дальнейшем будет рассматриваться ядро операционной системы Linux, так как она удовлетворяет выделенным критериям и занимает долю рынка большую, чем операционная система FreeBSD.

1.7 Сжатие памяти в ядре Linux

Ядро Linux является модульным. Модуль — это часть кода, которую можно загрузить и выгрузить в ядро по необходимости [22]. Модуль может быть встроенным или загружаемым.

Код встроенного модуля встраивается в образ ядра во время компиляции. Если ядру потребуется функциональность этого модуля, оно инициализирует и загрузит модуль. Для того, чтобы изменения такого модуля вступили в силу, необходимо перекомпилировать ядро.

Загружаемый модуль может быть загружен пользователем в ядро во время работы системы. Такие модули используют для добавления требуемой функциональности ядра без перезагрузки системы. Это позволяет избежать пересборки и перезагрузки ядра при каждом добавлении новых функций.

Возможность сжатия страниц оперативной памяти в ядре Linux предоставляют загружаемые модули `zram` и `zswap`.

1.7.1 Модуль `zram`

При загрузке модуля `zram` в оперативной памяти создается блочный специальный файл. Специальные файлы в операционной системе Linux служат для реализации работы с устройствами ввода-вывода как с файлами и

находятся в каталоге `/dev`. Это позволяет драйверам устройств проводить с устройствами ввода-вывода операции чтения и записи, используя те же системные вызовы, которые применяются для чтения и записи файлов. Блочные специальные файлы используются для моделирования устройств, содержащих набор блоков с произвольной адресацией. Открыв такой файл, программа может получить доступ к любому блоку устройства независимо от структуры имеющейся файловой системы [23]. При этом данные могут быть прочитаны или записаны только кратными блоками.

Созданный модулем `zram` блочный специальный файл 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.

На рисунке 1.10 представлена схема работы модуля `zram`.

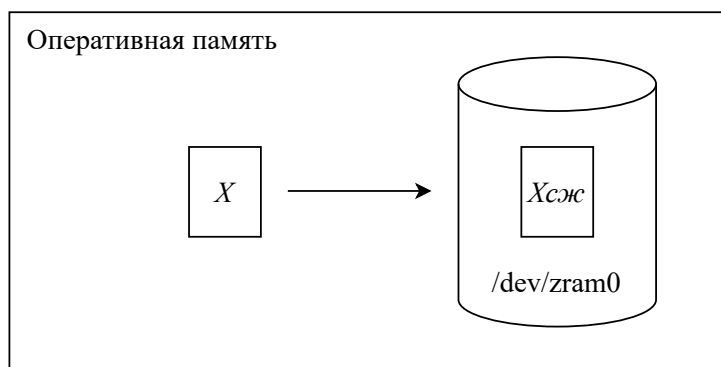


Рисунок 1.10 – Схема работы модуля `zram`

Модуль `zram` дает следующие преимущества [24]:

- 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` [25];
- 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.

1.7.2 Модуль zswap

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.

На рисунке 1.11 показана схема работы модуля zswap: при попытке записи страницы Z в заполненный пул длительно не используемая страница Y перемещается на устройство подкачки.

Максимальный размер сжатого пула можно установить, изменяя значение параметра `max_pool_percent`.

Модуль zswap дает следующие преимущества [26]:

- 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 [25].

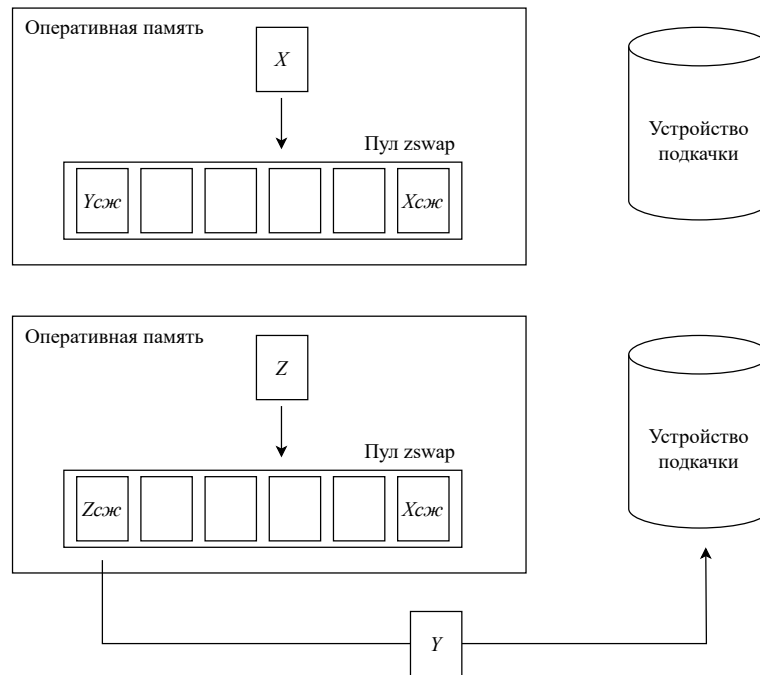


Рисунок 1.11 – Схема работы модуля zswap

Вывод

В данном разделе были рассмотрены такие механизмы подсистемы памяти операционной системы, как виртуальная память и подкачка страниц. В связи с недостатками перемещения данных между вторичным хранилищем и оперативной памятью было изучено сжатие памяти, которое является альтернативой подкачки страниц. Были описаны методы подсчета информационной энтропии, с помощью которой можно оценить качество сжатия. В результате проведенного анализа была поставлена и формализована задача оптимизации метода сжатия страниц оперативной памяти с использованием подсчета информационной энтропии.

На основании выполненного сравнения операционных систем для решения поставленной задачи была выбрана операционная система Linux, поддерживающая сжатие страниц оперативной памяти и имеющая открытый исходный код. Был произведен обзор модулей сжатия в ядре Linux. В данной работе будет выполняться оптимизация сжатия страниц памяти загружаемым модулем 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.

2 Конструкторский раздел

2.1 Функциональная модель оптимизации метода сжатия страниц

Разрабатываемая оптимизация метода сжатия страниц памяти с использованием подсчета информационной энтропии 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.

IDEF0-диаграмма первого уровня, формализующая основные этапы оптимизации сжатия страниц оперативной памяти, приведена на рисунке 2.1.

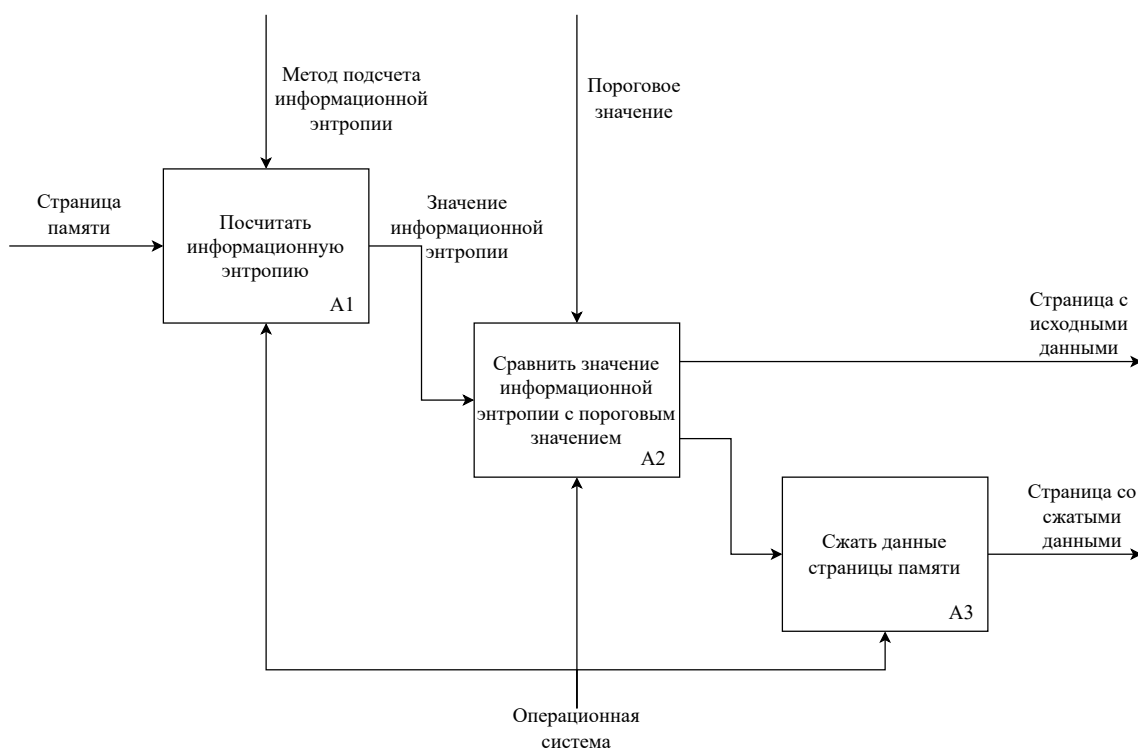


Рисунок 2.1 – IDEF0-диаграмма первого уровня

2.2 Требования к разрабатываемому программному обеспечению

Согласно описанию этапов решения поставленной задачи разрабатываемое программное обеспечение должно:

- вычислять информационную энтропию страницы оперативной памяти, которая является вектором $a = (a_1 \ a_2 \ \dots \ a_N)$ размером N , равным размеру страницы памяти, $0 \leq a_i \leq 255$;
- если вычисленное значение меньше порогового, сжимать данные входной страницы памяти, то есть, получать вектор $b = (b_1 \ b_2 \ \dots \ b_N)$ размером N , $0 \leq b_i \leq 255$;
- если вычисленное значение больше порогового или равно ему, данные входной страницы памяти не должны изменяться.

2.3 Структура разрабатываемого программного обеспечения

Структура загружаемого модуля ядра zgam 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 2.2.

Одним из выделенных к разрабатываемому программному обеспечению требованием является то, что сжатие страницы должно проводиться только в случае, если полученное значение информационной энтропии меньше порогового. Поэтому для выполнения поставленной задачи необходимо изменить функцию записи страницы на диск модуля блочного устройства. Построенная

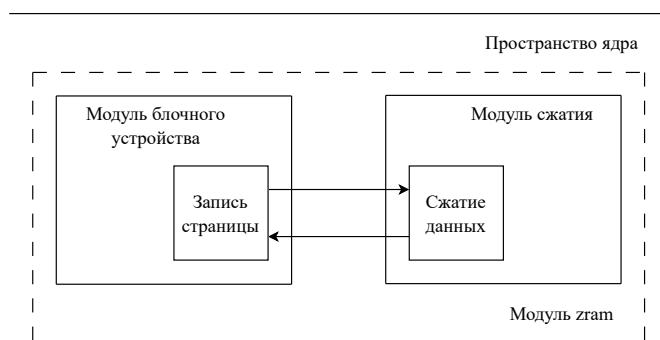


Рисунок 2.2 – Структура модуля zram

в соответствии с этим структура разрабатываемого программного обеспечения показана на рисунке 2.3. Пунктирными стрелками на схеме обозначены возможные переходы.

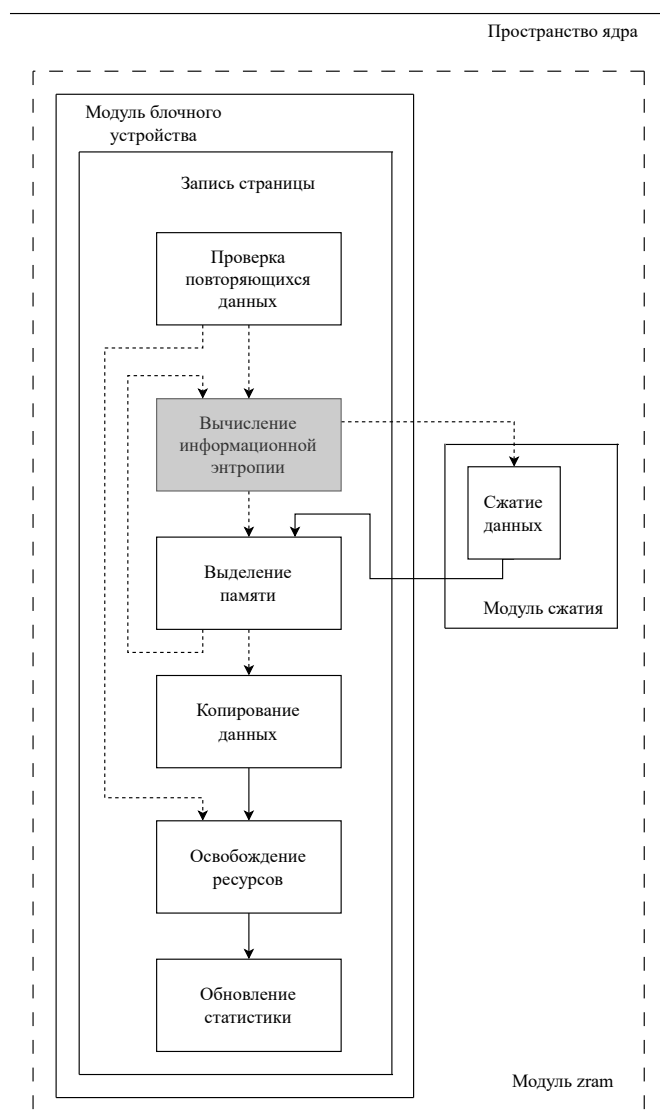


Рисунок 2.3 – Структура разрабатываемого программного обеспечения

2.4 Выбор типов и структур данных

Входными и выходными данными является страница памяти, которая задается вектором размером, меньшим или равным размеру страницы в байтах, состоящим из значений от нуля до 255. Поэтому для представления страницы в методе подсчета информационной энтропии должен использоваться массив типа `unsigned char` размером, равным размеру страницы в байтах.

При выполнении операций с числами с плавающей точкой в режиме пользователя ядро перехватывает системное прерывание и переходит из режима вычислений с целыми числами в режим с плавающей точкой. Если использовать режим с плавающей точкой в пространстве ядра, необходимо сохранять и восстанавливать состояние регистров с плавающей точкой математического сопроцессора. Вычисления с числами с плавающей точкой в режиме ядра выполнять не рекомендуется [27]. Поэтому для представления числовых величин должен использоваться целочисленный тип данных.

2.5 Описание оптимизации метода сжатия страниц

В результате анализа исходного кода загружаемого модуля `zram` [28] и описания этапов разрабатываемой оптимизации метода сжатия страниц памяти была построена схема алгоритма записи страницы на диск, приведенная на рисунке 2.4. Псевдокод данного алгоритма представлен в листинге .

TODO: псевдокод. TODO: написать про пороговое значение.

На разрабатываемый метод подсчета информационной энтропии накладываются следующие ограничения:

- вычисляемое значение может представлять неточное значение информационной энтропии, но должно обладать корреляцией с показателями качества сжатия;
- все числовые операнды в методе должны быть целыми числами.

Разрабатываемый метод является оптимизацией метода скользящего окна. В методе скользящего окна информационная энтропия вычисляется следующим образом:

$$H(X) = - \sum_{i=0}^{255} (p_i \cdot \log_2 p_i), \quad (2.1)$$

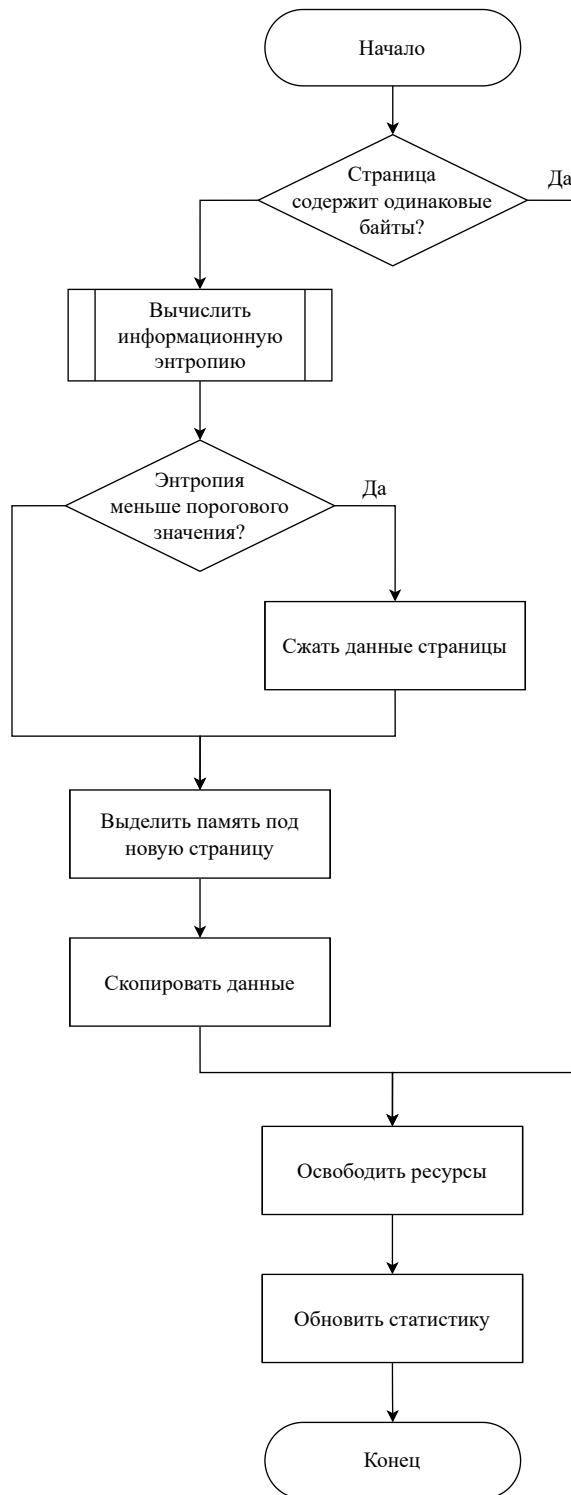


Рисунок 2.4 – Схема алгоритма записи страницы на диск

где k_i — число вхождений значения байта i в массив байтов, представляющий входную страницу, $p_i = \frac{k_i}{N}$ — вероятность появления байта в массиве байтов, $i = \overline{0, 255}$, N — размер страницы в байтах.

Тогда:

$$H(X) = - \sum_{i=0}^{255} \left(\frac{k_i}{N} \cdot \log_2 \frac{k_i}{N} \right). \quad (2.2)$$

В соответствии со свойством логарифма частного можно записать:

$$H(X) = - \sum_{i=0}^{255} \left(\frac{k_i}{N} \cdot (\log_2 k_i - \log_2 N) \right) = \sum_{i=0}^{255} \left(\frac{k_i}{N} \cdot (\log_2 N - \log_2 k_i) \right). \quad (2.3)$$

Так как все числовые операнды являются целыми числами, при определении p_i выполняется целочисленное деление и вычисляемое значение округляется. Для сохранения большей точности результатом разрабатываемого метода будет являться значение информационной энтропии, умноженное на размер страницы:

$$H'(X) = N \cdot H(X) = \sum_{i=0}^{255} (k_i \cdot (\log_2 N - \log_2 k_i)), \quad (2.4)$$

Для сокращения времени вычисления логарифма числа вхождений значения байта i в массив байтов по основанию два в разрабатываемом методе используется дополнительный массив размером $\log_2 N$, для которого верно следующее:

- индекс элемента j является целым значением логарифма числа вхождений значения байта i , $j = \overline{\log_2 1, \log_2 N}$;
- значение элемента с индексом j является наибольшим целым числом x таким, что $\log_2 x = j$.

Для определения $\log_2 k_i$ необходимо взять индекс первого элемента описанного массива, который больше или равен k_i .

Псевдокод алгоритма разрабатываемого метода подсчета информационной энтропии показан в листинге . Соответствующая ему схема алгоритма приведена на рисунке 2.5.

TODO: псевдокод.

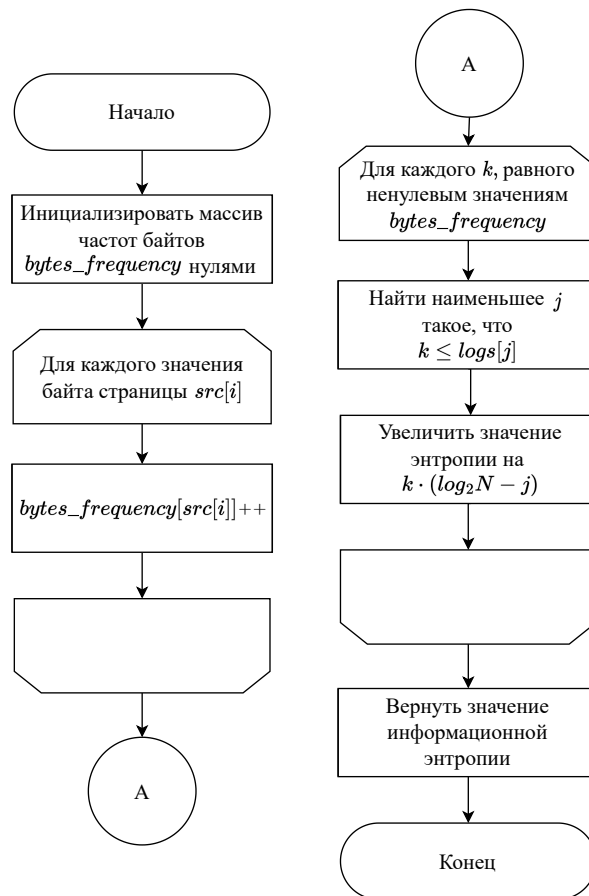


Рисунок 2.5 – Схема алгоритма подсчета информационной энтропии

2.6 Проектирование тестирования программного обеспечения

Разрабатываемое программное обеспечение выполняется при записи данных на диск `zram`. Поэтому для тестирования программного решения необходимо моделировать процесс загрузки страниц памяти на диск. Проверка и отладка разрабатываемого программного обеспечения должны проводиться с помощью ручного тестирования.

Для проведения тестирования выделены следующие классы эквивалентности:

1. Несжимаемые страницы памяти;
2. Сжимаемые страницы памяти.

Несжимаемыми страницами памяти являются страницы с высоким значением информационной энтропии. В разделе 1.3.2 было доказано, что эн-

тропия сжатых данных выше энтропии исходных данных. Поэтому тестовый набор для первого класса эквивалентности должен включать сжатые данные. Примерами таких данных являются файлы форматов .jpg и .pdf [29].

Сжимаемыми страницами памяти являются страницы данных с выделяемой структурой, как было сказано в разделе 1.3.2. Примерами таких данных являются файлы форматов .c, .h и бинарные файлы [30], из которых должен состоять тестовый набор для второго класса эквивалентности.

В результате каждого теста должна быть получена статистика сжатия тестового набора, включающая в себя следующую информацию:

- объем исходных данных в байтах и страницах;
- объем сжатых данных в байтах и страницах;
- число несжимаемых страниц;
- число страниц с одинаковыми байтами.

Система для выполнения тестирования включает следующие компоненты:

- персональный компьютер, который является хостом;
- виртуальная машина, на которой загружен модуль zgam без разрабатываемой оптимизации;
- виртуальная машина, на которой загружен модуль zgam с разрабатываемой оптимизацией.

Тестирование разрабатываемого программного обеспечения состоит из следующих этапов:

1. Загрузить тестовый набор с хоста на диск zgam виртуальной машины с работающим без разрабатываемой оптимизации модулем zgam.
2. Получить статистику выполнения предыдущего шага.
3. Загрузить тестовый набор с хоста на диск zgam виртуальной машины с работающим с разрабатываемой оптимизацией модулем zgam.

4. Получить статистику выполнения предыдущего шага.
5. Сравнить полученные статистики.

Схема тестирующей системы показана на рисунке 2.6.

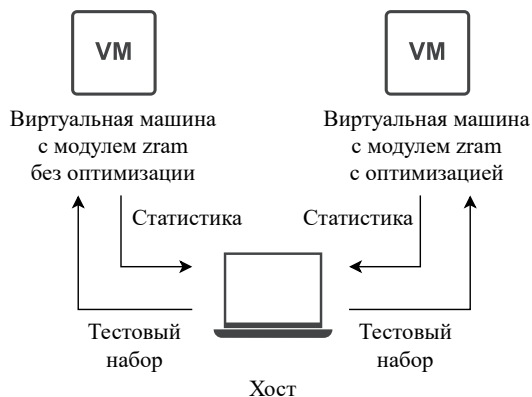


Рисунок 2.6 – Схема системы для тестирования

Вывод

В данном разделе были разработаны основные этапы оптимизации метода сжатия страниц памяти с использованием подсчета информационной энтропии. Их описание было приведено в виде псевдокода и схем алгоритмов. Были сформулированы требования к разрабатываемому программному решению. Взаимодействие компонентов системы было представлено в виде структуры программного обеспечения. Был обоснован выбор целочисленного типа данных и массива в качестве структуры, представляющей страницу памяти. Были выделены классы эквивалентности, описаны тестовые наборы для ручного тестирования, получаемая в результате статистика и тестирующая система.

3 Технологический раздел

3.1 Выбор средств программной реализации

Для выполнения оптимизации сжатия страниц памяти была выбрана операционная система Linux в соответствии со сравнением систем, проведенном в разделе 1.6. Согласно анализу модулей сжатия, представленному в разделе 1.7, был выбран загружаемый модуль zram. В качестве языка программирования был выбран язык C [31], 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 [32], так как с помощью данной утилиты осуществляется сборка загружаемых модулей ядра.

3.2 Детали реализации

В листинге 3.1 приведена реализация разработанного метода подсчета информационной энтропии. Размер страницы в байтах в ядре Linux определяется константой PAGE_SIZE [23].

Листинг 3.1 – Реализация метода подсчета информационной энтропии

```
1 #define PAGE_SIZE_POW 12
2 #define BYTES_NUM 256
3
4 static inline s32 get_entropy(const u8 *src)
5 {
6     u16 logs[PAGE_SIZE_POW + 1] = {1, 3, 7, 15, 31, 63, 127, 255, 511,
7                                     1023, 2047, 4095, 4096};
8     u16 bytes_frequency[BYTES_NUM] = { 0 };
9     u32 i, j;
10
11     for (i = 0; i < PAGE_SIZE; ++i) {
12         bytes_frequency[src[i]]++;
13     }
14
15     s32 entropy = 0;
16
17     for (i = 0; i < BYTES_NUM; ++i) {
18         s32 k = bytes_frequency[i];
19
20         if (k > 0) {
```

```

21
22         while (k > logs[j])
23             j++;
24
25         entropy += k * (PAGE_SIZE_POW - logs[j]);
26     }
27 }
28
29 return entropy;
30 }

```

В листинге 3.2 представлена часть реализации записи страницы на диск в модуле блочного устройства. `ENTROPY_TRESHOLD` — константа, определяющая пороговое значение и устанавливаемая перед сборкой программного обеспечения.

Листинг 3.2 – Часть реализации записи страницы на диск

```

1 static int __zram_bvec_write(struct zram *zram, struct bio_vec *bvec,
2                             u32 index, struct bio *bio)
3 {
4     ...
5     unsigned int comp_len = 0;
6     void *src, *dst, *mem;
7     struct zcomp_strm *zstrm;
8     struct page *page = bvec->bv_page;
9     ...
10
11 compress_again:
12     zstrm = zcomp_stream_get(zram->comp);
13     src = kmap_atomic(page);
14
15     if (get_entropy((const u8 *)src) < ENTROPY_THRESHOLD)
16         ret = zcomp_compress(zstrm, src, &comp_len);
17     else
18         comp_len = PAGE_SIZE;
19     ...
20 }

```

3.3 Конфигурация программного обеспечения

Для сборки и запуска программного обеспечения необходимо выполнить следующие действия:

1. получить исполняемый файл программного обеспечения;
2. загрузить полученный модуль.

При загрузке модуля автоматически создается диск `zram` с идентификатором 0. Для корректной работы программного обеспечения нужно настроить созданный диск:

1. установить алгоритм сжатия из доступных в системе;
2. установить размер диска, для указания единиц измерения которого используются следующие постфиксы: К — размер в килобайтах, М — размер в мегабайт, Г — размер в гигабайтах, размер в байтах устанавливается при отсутствии постфикса.

При получении статистики во время тестирования анализируются файл статистики памяти диска `mm_stat` и сообщения системного журнала.

Для остановки работы программного обеспечения необходимо выгрузить модуль. При этом автоматически удаляются все созданные диски `zram`.

Для выполнения при сборке, запуске, тестировании и остановке программного обеспечения и настройке диска описанных действий был написан `make-файл`, код которого показан в листинге 3.3. В таблице представлено описание доступных команд.

Листинг 3.3 – Конфигурационный файл

```
1 KERNEL_VERSION = $(shell uname -r)
2 MODULE_DIR = $(shell pwd)
3
4 OBJS = zcomp.o zram_drv.o
5 TARGET = zram
6
7 obj-m := $(TARGET).o
8 $(TARGET)-objs := $(OBJS)
9
10 all:
11     $(MAKE) -C /lib/modules/$(KERNEL_VERSION)/build M=$(MODULE_DIR) modules
12
13 $(TARGET).o: $(OBJS)
14     $(LD) -r -o $@ $(OBJS)
15
16 clean:
17     @rm -f *.o *.cmd *.flags *.mod.c *.order
18     @rm -f *.*.cmd *~ *.*~
19     @rm -fR .tmp*
20     @rm -rf .tmp_versions
21
22 distclean:
```

```

23     @rm -f *.ko *.symvers *.mod
24
25 load:
26     sudo insmod $(TARGET).ko
27
28 info:
29     sudo lsmod | grep $(TARGET)
30
31 log:
32     sudo dmesg | grep $(TARGET)
33
34 unload:
35     sudo rmmod $(TARGET).ko
36
37
38 add-disk:
39     cat /sys/class/zram-control/hot_add
40
41 get-comp-algorithm:
42     cat /sys/block/zram$(id)/comp_algorithm
43
44 set-comp-algorithm:
45     echo $(algorithm) > /sys/block/zram$(id)/comp_algorithm
46
47 set-disksize:
48     echo $(disksize) > /sys/block/zram$(id)/disksize
49
50 memory-stat:
51     cat /sys/block/zram$(id)/mm_stat
52
53 reset-disk:
54     echo 1 > /sys/block/zram$(id)/reset
55
56 remove-disk:
57     echo $(id) > /sys/class/zram-control/hot_remove

```

3.4 Тестирование разработанного программного обеспечения

Тестирование разработанного программного обеспечения проводилось с помощью тестовых наборов, которые представляют собой архивы, созданные с помощью утилиты tar [33] следующей командой: `tar cf name.tar name`, где `name` — имя тестового набора. В соответствии с выделенными классами эквивалентности были созданы следующие тестовые наборы:

Таблица 3.1 – Команды конфигурационного файла

Команда	Аргументы	Описание
make	-	собрать загружаемый модуль
make load	-	загрузить собранный модуль
make info	-	проверить, что модуль загружен
make log	-	получить сообщения системного журнала
make unload	-	выгрузить модуль
make add-disk	-	добавить диск, будет выведен его идентификатор
make get-comp-algorithm	id — идентификатор диска	получить список доступных алгоритмов сжатия
make set-comp-algorithm	algorithm — алгоритм сжатия id — идентификатор диска	установить алгоритм сжатия
make set-disksize	disksize — размер диска id — идентификатор диска	установить размер диска
make memory-stat	id — идентификатор диска	получить статистику памяти диска
make reset-disk	id — идентификатор диска	освободить память диска
make remove-disk	id — идентификатор диска	удалить диск

- bin-test.tar — архив, содержащий бинарные файлы;
- text-test.tar — архив, состоящий из файлов с расширением .h;
- pdf-test.tar — архив, включающий в себя файлы с расширением .pdf;
- img-test.tar — архив, составленный из файлов с расширением .jpg.

Загрузка данных в устройство выполнялась следующей командой: `dd if=name.tar of=/dev/zram0`, где name — имя тестового набора.

В результате тестирования были получены следующие результаты.

TODO: добавить статистику по каждому тестовому набору и пояснения к ней.

Вывод

В данном разделе был обоснован выбор языка программирования C и утилиты make в качестве средств программной реализации. Были представлены детали реализации программного решения. Кроме того были изложены команды для сборки и запуска программного обеспечения и настройки устройства zgam. Были описаны результаты тестирования разработанного программного обеспечения.

4 Исследовательский раздел

ЗАКЛЮЧЕНИЕ

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. *Филиппов А. А.* Операционные системы: учебное пособие. — Ульяновск: Изд-во Ульян. тех. ун-та, 2021. — 100 с.
2. *Столлингс В.* Операционные системы: внутренняя структура и принципы проектирования: пер. с англ. — 9-е изд. — СПб.: ООО «Диалектика», 2020. — 1264 с.
3. *Silberschatz A., Galvin P. B., Gagne G.* Operating System Concepts. — 10th. — Hoboken, NJ: Wiley, 2018. — 1278 с.
4. Approximate Memory Compression / A. Ranjan [и др.] // IEEE Transactions on Very Large Scale Integration (VLSI) Systems. — 2020. — т. 28, № 4. — с. 980—991.
5. *Uthayakumar J., Vengattaraman T., Dhavachelvan P.* A survey on data compression techniques: From the perspective of data quality, coding schemes, data type and applications // Journal of King Saud University — Computer and Information Sciences. — 2021. — 119—140 с.
6. *Пантелеев Е. Р., Алыкова А. Л.* Алгоритмы сжатия данных без потерь: учебное пособие для вузов // 2-е изд., стер. — СПб.: Лань. — 2022. — 172 с.
7. L2C: Combining Lossy and Lossless Compression on Memory and I/O / A. Ranjan [и др.] // ACM Trans. Embed. Comput. Syst. — 2022. — т. 21, № 1. — с. 1—27.
8. Данные, не поддерживающие сжатие [Электронный ресурс]. — Режим доступа URL: https://www.ibm.com/docs/ru/informix-servers/12.10?topic=data_that_you_cannot_compress (Дата обращения: 07.05.2023).
9. *Березкин Е. Ф.* Основы теории информации и кодирования: учебное пособие // 3-е изд., стер. — СПб.: Лань. — 2022. — 320 с.
10. *Rodrigues M.* Information-Theoretic Methods in Data Science // Cambridge: Cambridge University Press. — 2021. — 43 с.
11. *Попов И. Ю., Блинова И. В.* Теория информации // 3-е изд., стер. — СПб.: Лань. — 2022. — 160 с.

12. *Осокин А. Н., Мальчуков А. Н.* Теория информации: учебное пособие для вузов // М.: Издательство Юрайт. — 2022. — 205 с.
13. *Zbili M., Rama S.* A Quick and Easy Way to Estimate Entropy and Mutual Information for Neuroscience // Frontiers in Neuroinformatics. — 2021.
14. *Cheng X., Li Z.* How does Shannon’s source coding theorem fare in prediction of image compression ratio with current algorithms? // International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences. — 2020. — 1313—1319 с.
15. *Ryabko B.* Time-Universal Data Compression // Algorithms. — 2019.
16. *Guo H.* File Entropy Signal Analysis Combined With Wavelet Decomposition for Malware Classification // IEEE Access. — 2020. — 158961—158971 с.
17. *Пухальский Г. И., Новосельцева Т. Я.* Проектирование цифровых устройств: учебное пособие для вузов // СПб.: Лань. — 2022. — 896 с.
18. *Borysenko O.* On the binomial method for calculation of entropy // Grail of Science. — 2022. — 113—118 с.
19. *Рябко Б. Я., Фионов А. Н.* Криптография в информационном мире // М.: Горячая линия-Телеком. — 2018. — 300 с.
20. Что представляют инструменты с открытым исходным кодом? [Электронный ресурс]. — Режим доступа URL: <https://aws.amazon.com/ru/what-is/open-source/> (Дата обращения: 07.05.2023).
21. 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).
22. The Linux Kernel Module Programming Guide [Электронный ресурс]. — Режим доступа URL: <https://sysprog21.github.io/lkmpg/#introduction> (Дата обращения: 07.05.2023).
23. *Fox R.* Linux with Operating System Concepts. — 2th. — Boca Raton: CRC Press, 2022. — 100 с.
24. 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).

25. Kernel Crypto API Interface Specification [Электронный ресурс]. — Режим доступа URL: <https://docs.kernel.org/crypto/intro.html> (Дата обращения: 07.05.2023).
26. zswap [Электронный ресурс]. — Режим доступа URL: <https://docs.kernel.org/admin-guide/mm/zswap.html?highlight=zswap> (Дата обращения: 07.05.2023).
27. Лав Р. Ядро Linux: описание процесса разработки. — 3-е изд. — Пер. с англ. — М.: ООО «И.Д. Вильямс», 2013. — 496 с.
28. zram - drivers/block/zram - Linux source code (v5.15.71) - Bootlin [Электронный ресурс]. — Режим доступа URL: <https://elixir.bootlin.com/linux/v5.15.71/source/drivers/block/zram> (Дата обращения: 07.05.2023).
29. Why don't some files compress very much? [Электронный ресурс]. — Режим доступа URL: <https://kb.corel.com/en/125893> (Дата обращения: 07.05.2023).
30. Данные, поддерживающие сжатие [Электронный ресурс]. — Режим доступа URL: https://www.ibm.com/docs/ru/informix-servers/12.10?topic=data_that_you_can_compress (Дата обращения: 07.05.2023).
31. 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).
32. GNU make [Электронный ресурс]. — Режим доступа URL: <https://www.gnu.org/software/make/manual/make.html#Simple-Makefile> (Дата обращения: 07.05.2023).
33. GNU Tar [Электронный ресурс]. — Режим доступа URL: <https://www.gnu.org/software/tar/> (Дата обращения: 07.05.2023).