ИНФОРМАЦИОННЫЕ ТЕХНОЛОГИИ И ТЕЛЕКОММУНИКАЦИИ
INFORMATION TECHNOLOGY AND TELECOMMUNICATIONS

УДК 004. 451

DOI: 10.21822/2073-6185-2025-52-4-118-125 Оригинальная статья/ Original article

Поиск неиспользуемого программного обеспечения в операционных системах LINUX

А.Д. Попов
Воронежский институт МВД России,
394065, г. Воронеж, пр. Патриотов, 53, Россия

Резюме. Цель. Целью исследования является разработка программного обеспечения (ПО) на языке Python для автоматизированного поиска неиспользуемых исполняемых файлов в операционных системах Linux. Актуальность задачи обусловлена необходимостью оптимизации использования памяти, сокращения поверхности атаки и улучшения контроля над системой за счет удаления неиспользуемого ПО. Современные дистрибутивы Linux содержат множество служебных утилит и сторонних программ, которые могут никогда не использоваться, что приводит к нерациональному расходованию ресурсов. **Метод**. Исследование основано на анализе метаданных файлов (время последнего доступа — atime), сканировании каталогов из переменной окружения PATH и учете особенностей файловых систем Linux (режим relatime). **Результат.** Разработанное ПО включает две подсистемы: взаимодействия с пользователем и сбора/анализа данных. Для обеспечения переносимости предложен метод развертывания в контейнеризированной среде Docker. Создано программное решение, которое формирует отчеты о неиспользуемых исполняемых файлах на основе заданной пользователем даты, предоставляет рекомендации по удалению, включая зависимости и способы деинсталляции через менеджеры пакетов, поддерживает динамическое развертывание через Docker, адаптируясь к разным конфигурациям PATH. **Вывод.** Инструмент дополняет существующие утилиты (например, deborphan), фокусируясь на времени использования ПО. Его простота и минималистичность позволяют легко внедрять в процессы администрирования. Перспективы развития включают портирование на Golang с графическим интерфейсом и интеграцию в системы автоматизированного развертывания ОС.

Ключевые слова: Linux, операционная система, файловая система, программное обеспечение, менеджеры пакетов, PATH, relatime, atime, Docker

Для цитирования: А.Д. Попов. Поиск неиспользуемого программного обеспечения в операционных системах LINUX . Вестник Дагестанского государственного технического университета. Технические науки. 2025;52(4):118-125. DOI:10.21822/2073-6185-2025-52-4-118-125

Finding Unused Software in Linux Operating Systems

A.D. Popov
Voronezh Institute of the Ministry of Internal Affairs of Russia,
53 Patriotov Str., Voronezh 394065, Russia

Abstract. Objective. The aim of this study is to develop Python software for the automated search for unused executable files in Linux operating systems. The relevance is due to the need to optimize memory, reduce the attack surface and improve control over the system by removing unused software. Modern Linux distributions contain numerous utility programs and third-party programs that may never be used, leading to inefficient resource usage. **Method.** The study is based on file metadata analysis (last access time - atime), scanning directories from the PATH environment variable, and taking into account the specific features of Linux file systems (relatime mode). **Result.** The developed software includes two subsystems: user interaction and data collection/analysis. To ensure portability, a deployment method in a Docker containerized environment

is proposed. A software solution has been created that generates reports on unused executable files based on a user-specified date, provides removal recommendations, including dependencies and uninstallation methods via package managers, and supports dynamic deployment via Docker, adapting to different PATH configurations. **Conclusion.** The tool complements utilities (such as deborphan), focusing on software usage time. Its simplicity and minimalist design allow for integration into administrative processes. Future development prospects include porting to Golang with a graphical interface and integration into automated OS deployment systems.

Keywords: Linux, operating system, file system, software, package managers, PATH, relatime, atime, Docker

For citation: A.D. Popov. Finding Unused Software in Linux Operating Systems. Herald of the Daghestan State Technical University. Technical Sciences. 2025; 52(4):118-125. (In Russ) DOI:10.21822/2073-6185-2025-52-4-118-125

Введение. Существующая реальность показала, что использование свободно распространяемого ПО в производственной среде является актуальным. Это обосновывается тем, что недобросовестные корпорации могут прекратить поддерживать приобретенное ПО в угоду третьим силам. Данный факт накладывает дополнительные усилия со стороны администраторов систем по поддержке работоспособности инфраструктуры предприятия. В настоящее время можно сказать, что конкуренцию проприетарному ПО, свободно распространяемое проигрывает, как и в некоторых отраслях, так и по функциональности. Но все же наблюдаются значительные тенденции, связанные с переработкой и добавления в штат отечественного. ОС в данной связке играют ключевую роль.

Управление ОС становится актуальной задачей, поскольку количество автоматизированных рабочих мест увеличивается с каждым годом. Основой для организации информационной структуры с точки зрения пользователя и администратора лежит на ОС семейства Linux. Для их управления используются программы позволяющие, автоматизировать развертывание ОС на различную инфраструктуру. К ним относятся Ansible, OpenTofu и др., а также применяются самописные скрипты [1,2]. Но в данном контексте под управлением понимается широкая область администрирования: управление пользователями, сетью, памятью, процессами и др.

Постановка задачи. Тенденцией в построении современных информационных систем является оптимальное использование памяти, в частности, в ОС не должно быть включено ничего лишнего, что не связано с ее функционированием. Поэтому оптимальный состав ПО в ОС Linux может сократить поверхность атаки, сэкономить используемую память и улучшить контроль над состоянием системы в целом.

В настоящее время в ОС систему включены множество служебных утилит, которые могут никогда не использоваться, как и сторонние программы после установки и однократного выполнения. Поэтому можно сказать, что память используется не по назначению до окончания жизненного цикла ОС, инсталлированного на аппаратное обеспечение. Для решения поставленной задачи необходимым навыком является управление программами в ОС Linux. Для его автоматизации актуальна разработка самостоятельного программного решения как подхода к оптимизации использования памяти, который, может являться, одним из инструментов управления ОС семейства Linux в целом.

Методы исследования. Знакомство с управлением программами в ОС Linux традиционно начинается с менеджеров пакетов, которые используют фактически все дистрибутивы. Они используются для установки, удаления и предоставления информации об установленном ПО. Как было сказано выше, другим способом установки ПО является пакет, который представляет собой бинарный файл. Следующим способом установки ПО является установка из исходного кода, где в большей своем содержится файл с инструкцией по установке [3,4]. После установки в основном все программы распределяются по существующим каталогам в ОС Linux по предназначению. Для того чтобы запуск программы был стандартизирован создана переменная окружения PATH, которая содержит информацию

о имеющихся каталогах [4]. При вызове программ происходим обход всех каталогов на наличие сходства и происходит ее запуск. В представленную переменную могут добавляться другие пути к каталогам для запуска программ.

Стандартное содержимое переменной PATH выглядит следующим образом «/usr/local/sbin:/usr/local/bin:/usr/sbin:/usr/bin:/sbin:/bin».

1. Директория /bin предназначена для хранятся основных исполняемых файлов, необходимых для минимальной загрузки и функционирования ОС. Данные файлы включают в себя утилиты командной строки, необходимые для работы системы, к примеру, для работы с файловой системой, команды для управления процессами и др.

2. Директория /usr/bin предназначена для хранятся основных исполняемых файлов и пользовательских программ, а также содержат утилиты и приложения, которые используются пользователями системы. В отличие от /bin, содержимое /usr/bin не является необходимым для загрузки или функционирования базовой системы.

3. В директории /sbin хранятся системные исполняемые файлы, которые используются для администрирования и управления системой. Выполнение представленных файлов обычно требуют повышенных привилегий (как правило, root). Могут быть использованы на ранних стадиях загрузки системы или для восстановления после сбоев, монтирования файловых систем, настройки сети, администрирования учетных записей и др.

4. В директории /usr/sbin хранятся системные исполняемые файлы, которые также предназначены для администрирования и управления, но они обычно не являются критически важными для базового функционирования. Данные файлы могут быть установлены отдельно от основной системы и предназначены для использования системными администраторами. В качестве примера в данный каталог включены утилиты для управления службами, настройки безопасности и др.

5. Директория /usr/local/bin предназначена для хранения исполняемых файлов и пользовательских программ, которые не входят в стандартный набор поставляемых с ОС. Обычно в данную директорию располагают программы, которые не устанавливаются с помощью пакетных менеджеров, а компилируются и устанавливаются вручную или с помощью сторонних инструментов установки. Файлы, расположенные в /usr/local/bin доступны всем пользователям системы.

6. Директория /usr/local/sbin аналогична /usr/local/bin, но предназначена для хранения системных исполняемых файлов пользовательских программ, которые требуют повышенных привилегий для выполнения. Эти файлы также не входят в стандартный набор поставляемых с ОС и устанавливаются вручную или с помощью других инструментов установки. Файлы в /usr/local/sbin обычно доступны только администраторам системы.

В представленный перечень каталогов, содержащийся в переменной PATH отсутствует /opt, в данной директории располагаются дополнительные пакеты. Например, при установке программы из deb пакета, в вышеперечисленные каталоги помещается файл ссылающийся на исходный код (символьная ссылка), который располагается в /opt. Но и как почти во всех правилах есть исключения, и они связаны с термином «тарбол», который представляет собой архив в формате tar, tar.gz и др. Он содержит в себе все необходимые зависимости, исполняемый файл и др. - все то, что необходимо для запуска программы. Ключевая особенность использования тарболов в ОС Linux заключается в том, что эти файлы при разархивировании могут находиться в любой части файловой системы и запускаться только при прямом обращении, тем самым, находиться вне пределов стандартных каталогов, перечисленных выше, что свидетельствует о человеческом факторе и невозможности контроля всего установленного ПО. Для преодоления данной неопределенности за правило необходимо взять администраторам создание символьных ссылок на исполняемый файл, входящий в тербол. Основой для сбора данных об использовании файлов в ОС Linux является анализ метаданных. На пользовательском уровне информация предоставляется при помощи встроенной утилиты командной строки stat.

Ее работа основывается на одноименном системном вызове stat(), который возвращает атрибуты файла для индексного дескриптора (Inode) и включает в себя следующие свойства: **atime** (время последнего доступа); **mtime** (время последней модификации); **ctime** (время последнего изменения статуса).

Для достижения цели статьи, выявление неиспользованного ПО в ОС Linux, свойство atime является определяющим. Время доступа atime показывает, когда в последний раз осуществлялся доступ к файлу и/или каталогу напрямую при помощи процесса в ОС или через утилиты командной строки. Согласно своему предназначению, атрибут atime должен обновляться (то есть записываться на диск) каждый раз, при обращении, даже если это была просто операция чтения. При высоких нагрузках atime может серьезно влиять на производительность файловых систем, особенно на жесткие диски в отличие от твердотельных накопителей. В связи с вышеизложенным поведением atime были придуманы различные опции монтирования файловых систем, связанные со стратегиями функционирования ОС Linux в целом [6-8].

Существуют следующие: **strictatime** – всегда обновляет время доступа; **relatime** – выборочное обновление времени; **nodiratime** – не обновляется время доступа для каталогов; **noatime** – никаких обновлений времени доступа для чего-либо.

Опция relatime является наиболее популярной и используется по умолчанию при монтировании, поэтому рассмотрим ее подробнее. Концептуально постоянная перезапись метаданных осуществляет дополнительную нагрузку, что разработчики ОС посчитали излишним. Для экономии ресурсов определились, что по умолчанию метаданные всех файлов будут меняться в течении 24 часов одновременно. Это улучшит производительность системы, особенно на носителях с ограниченным числом циклов записи. Данное поведение и характеризуется режимом relatime, но при создании файловых систем его можно изменить, указанием представленных выше опций.

Метаданные о времени доступа к файлу (atime) в режиме relatime, обновляются в следующих случаях:

1. Предыдущее значение atime старше, чем текущее значение mtime (время последней модификации файла) или ctime (время последнего изменения состояния файла);
2. Atime не обновлялось в течение определённого периода времени (по умолчанию 24 часа). Если предыдущее время доступа (atime) отстает более чем на 24 часа назад.

Это означает что, если вы часто обращаетесь к файлу, atime будет обновляться не при каждом доступе, а с интервалом в день или при изменении файла, что снижает нагрузку на файловую систему. Опция relatime включена по умолчанию в большинстве современных дистрибутивов Linux и является компромиссом между noatime, который полностью отключает обновление atime, и ранее используемым стандартом strictatime который обновлял atime при каждом доступе к файлу. Особенности изменения метаданных файловых систем в ОС семейства Linux и в целом их функционирования, позволили выделить требования для создания ПО поиска неиспользуемый исполняемых файлов. К ним относится: простота использования, минималистичность, функциональность, интуитивно понятный интерфейс, быстрая установка и перенос в виде скрипта.

Обсуждение результатов. Разработанное ПО состоит из двух подсистем, подсистема взаимодействия с пользователем, подсистема сбора и анализа метаданных файлов. Подсистема взаимодействия с пользователем предоставляет справочную информацию о предназначении установленного ПО, об опциях функционирования ОС, позволяет вводить установочную информацию для поиска необходимых метаданных по дате использования и получать рекомендациях для удаления ПО. На первоначальном этапе происходит сканирование файловой системы для получения опций монтирования при помощи команды «mount | grep " / "» и предоставляется справочная информация по общему количеству исполняемых файлов, находящихся в каталогах переменной PATH (рис. 1). Следующим шагом пользователю доступна справка о предназначении данных каталогов и предлагается ввести дату старше которой будет произведено сканирование исполняемые файлы (рис. 2).

```
xub@xub:~/nupil$ python main_test.py
##################################
###########    NUPIL    ##########
##################################
¯\_(ツ)_/¯ nupil - предназначен для поиска неиспользуемых или последних
исполняемых файлов в *nix операционных системах

Нажмите любую клавишу для продолжения
++++++++++++++++++++++++++++++++++++

Ваш режим доступа к дискам
/dev/sda2 on / type ext4 (rw,relatime,errors=remount-ro)

++++++++++++++++++++++++++++++++++++
Введите 'a' или 'A' для получения справки о режимах доступа к дискам: a
Опции монтирования дисков

Каталоги PATH и количество исполняемых файлов в них
Path: /usr/local/sbin | Number files: 0
Path: /usr/local/bin | Number files: 0
Path: /usr/sbin | Number files: 385
Path: /usr/bin | Number files: 1489
Path: /sbin | Number files: 385
Path: /bin | Number files: 1489
Path: /usr/games | Number files: 0
Path: /usr/local/games | Number files: 0
Path: /snap/bin | Number files: 4
Total file: 3752
```

Рис. 1 – Сводная справка по файловой системе
Fig. 1 – Summary of the file system

```
Введите 'a' или 'A' для получения справки о предназнач
ении каталогов переменной PATH, или любую другую клави
шу для продолжения:
Продолжаем

Для определяния неспользуемых бинарных файлов введите
ДАТУ, старше которой произвести сканирование
Enter a year: 2022
Enter a month: 6
Enter a day: 1
Вы ввели дату 2022-06-01
Нажмите любую клавишу для выполнения сканирования
```

Рис. 2 – Ввод данных
Fig. 2 – Data entry

Подсистема сбора и анализа метаданных файлов на первоначальном этапе выполняет поиск исполняемых файлов согласно указанных пользователем критериев. На следующем этапе происходит предобработка с целью формирования необходимых данных для предоставления пользователю, в удобочитаемом формате. Результатом является структура в виде абсолютного пути до исполняемого файла, дата последнего обращения, размер и каталог его расположения из переменной PATH (рис. 3). Результатом сканирования является отчет о существующих исполняемых файлам, обращение к которым осуществлялось раньше, чем введенная дата пользователем (рис 4).

```
Файл: /bin/xlsfonts 2020-02-29 32K /bin
Файл: /bin/xmag 2020-02-29 56K /bin
Файл: /bin/xman 2020-02-29 92K /bin
Файл: /bin/xmessage 2020-02-29 32K /bin
Файл: /bin/xmodmap 2018-03-19 44K /bin
Файл: /bin/xmore 2020-02-29 24K /bin
Файл: /bin/xrandr 2018-03-19 72K /bin
Файл: /bin/xrefresh 2018-03-19 16K /bin
Файл: /bin/xset 2018-03-19 40K /bin
Файл: /bin/xsetmode 2018-03-19 16K /bin
Файл: /bin/xsetpointer 2018-03-19 16K /bin
```

Рис. 3 – Сводная справка по файловой системе
Fig. 3 – Summary of the file system

```
Вы ввели дату 2023-01-01
Нажмите любую клавишу для выполнения сканирования
Выполняется сканирование...
/home/mywork/.local/bin 0
/home/mywork/.local/bin 0
/usr/local/sbin 0
/usr/local/bin 0
/usr/sbin 71
/usr/bin 398
/sbin 71
/bin 398
/usr/games 41
/usr/local/games 0
/snap/bin 0
Сканирование успешно выполнено.
Общее количесвто неиспользованных бинирных файлов = 979
```

Рис. 4 – Результат сканирования по каталогам в PATH
Fig. 4 – Result of scanning directories in PATH

На следующем этапе снова осуществляет работу подсистема взаимодействия с пользователем, которая позволяет просмотреть результаты сканирования при помощи выбора каталога и указания исполняемого файла, который требуется проанализировать (рис. 5).

```
Для получения информации по кнтретному пакету введите абсолютный путь
Например /sbin/update-xmlcatalog : /usr/bin/htop
++++++++++++++++++++++++++++++++++++
Пакет предназначен
htop (1)              - interactive process viewer
++++++++++++++++++++++++++++++++++++
Зависимости пакета
linux-vdso.so.1 (0x00007fff7d3de000)
        libncursesw.so.6 => /lib/x86_64-linux-gnu/libncursesw.so.6 (0x00007f43a3b5900
0)
        libtinfo.so.6 => /lib/x86_64-linux-gnu/libtinfo.so.6 (0x00007f43a3b29000)
        libm.so.6 => /lib/x86_64-linux-gnu/libm.so.6 (0x00007f43a39da000)
        libc.so.6 => /lib/x86_64-linux-gnu/libc.so.6 (0x00007f43a37e8000)
        libdl.so.2 => /lib/x86_64-linux-gnu/libdl.so.2 (0x00007f43a37e2000)
        /lib64/ld-linux-x86-64.so.2 (0x00007f43a3bdf000)
++++++++++++++++++++++++++++++++++++
Для получения зависимостей пренадлежащих пакету можно использовать дополнительную ком
наду
apt-cache depends --recurse --no-recommends --no-suggests --no-conflicts --no-breaks
--no-replaces --no-enhances htop | grep "^\w" | sort -u
++++++++++++++++++++++++++++++++++++
Для удаления исполняемого файла рекомендуется использовать менеджеры пакетов
apt, dpkg, snap и др.
apt remove htop
dpkg -r htop - оставлет файлы конфигурации
dpkg -purge htop - удаляет файлы конфигурации
snap remove htop
++++++++++++++++++++++++++++++++++++
Для просфотра всех фалов пренадлежащих пакету
whereis htop
locate htop
Понадобятся права суперпользователя
find / -name htop
Нажмите любую клавишу для завершения работы программы
¯\_(ツ)_/¯
########################################
############    NUPIL    ###############
########################################
```

Рис. 5 – Рекомендации по удалению неиспользованного исполняемого файла
Fig. 5 – Recommendations for deleting an unused executable file

В отчете по конкретному исполняемому файлу предоставляется укороченная справка по выборному исполняемому файлу, списку зависимостей, рекомендации по способам удаления через менеджеры пакетов, а также полезные команды для углубленного изучения артефактов пакета, располагаемых в ОС. В отличие от существующего ПО deborphan и его графического аналога gtkorphan, которые направлены на удаление зависших зависимостей. Представленное в статье решение позволяет контролировать существование исполняемых файлов в зависимости от времени их последнего использования или изменения. В связи с этим разработанная программа может дополнять сторонние разработки.

Автор предполагает возможное использование асинхронных подходов к программированию, но не преследует целью использование сторонних библиотек для дополнительного увеличения кодовой базы. В связи с этим представленное программное решение в статье увеличит время использования, чем можно пренебречь в связи небольшим объёмом сканирования объектов файловой системы. Но, с другой стороны, позволит быстро переносить разработанное ПО. Для его корректного функционирования необходим только Python, который предустановлен почти на всех дистрибутивах Linux.

Одним из вариантов применения разработанного ПО в ОС Linux могут считаться контейнеризированные среды на примере Docker. Их преимуществом в контексте представленных результатов будет являться полная изоляция от хостовой системы, что при увеличении кодовой базы может быть востребовано [9-11]. На рис. 6 представлен Dockerfile, позволяющий осуществить сборку контейнера. На рис. 7 скрипт развертывания и запуска контейнера, особенность которого заключается в добавлении манифест файла в скрипт. Этот изысканный подход обеспечит динамическое добавление каталогов, находящихся в переменной PATH (в связи с тем, что их количество на разных системах может отличаться).

```
FROM ubuntu:24.04

RUN apt-get update -y
RUN apt-get install -y python3.12
RUN ln -s /usr/bin/python3.12 /usr/bin/python

WORKDIR /nupil_mount
RUN mkdir /nupil_mount/nupil

COPY main.py /nupil_mount/nupil
```

Рис. 6 – Dockerfile для сборки образа
Fig. 6 – Dockerfile for building the image

```
#!/bin/bash
path_content=$PATH

IFS=':' read -ra path_segments <<< "$path_content"
buildComposeYaml() {
  cat <<HEADER
version: '3'
services:
  nupil:
    image: nupil_ubuntu:0.0.4
    # command: python /nupil_mount/nupil/main.py
    tty: true
    volumes:
HEADER
    for i in "${path_segments[@]}"; do
    cat <<BLOCK
    - $i:/nupil_mount$i:rw

BLOCK
    done
}
buildComposeYaml | docker-compose -f- "$@" up
```

Рис. 7. – Скрипт развертывания и запуска контейнера
Fig. 7. – Script for deploying and launching a container

Вывод. Предполагается, что пользователь использует программу в личных целях для улучшения качества администрирования. ПО предоставляет рекомендации по оптимальному использованию исполняемых файлов в ОС Linux. Тестирования происходило в ОС RedHat и Debian. Исходный код располагается по адресу https://github.com/anton-holmes/nupil.git. Стоит отметить, что разработка не подходит к использованию вместе с декларативными менеджерами пакетов типа Nix [12].

Перспективным направлением развития, представленного ПО может считаться создание исполняемого файла на языке Golang с оконным интерфейсом и дополнительными функциональными возможностями [13]. Перспективным направлением развития исследования является автоматизация сборки ОС с учетом включения в нее только тех исполняемых файлов, которые необходимы для ее минимального функционирования с учетом выполнения всех поставленных перед ОС задач. Полученные результаты могут быть использованы в научной деятельности, связанной с оценкой функционирования и качества ОС, а также в практической связанной с администрированием информационных систем.

Библиографический список:

1. Chavan M.P., Chavan P.M. Rpm packaging for Ansible automation configuration management in Linux // Interantional journal of scientific research in engineering and management. 2022. Т. 06. № 12. P. 1–3
2. Irons L.G., Irons M.A. Terraform sustainability assessment framework for bioregenerative life support systems // Frontiers in Astronomy and Space Sciences. 2021. Т. 8. С. 789563.
3. Кривцова И.Е., Салахутдинова К.И., Юрин И.В. Метод идентификации исполняемых файлов по их сигнатурам // Вестник государственного университета морского и речного флота им. адмирала С.О. Макарова. 2016. № 1 (35). С. 215–-224.

4. Орещенков И. Запуск современных программ на устаревшем дистрибутиве Linux // Системный администратор. 2018. № 12 (193). С. 22–28.
5. Жеряков Д.В. Базовые пакеты по, необходимые для организации работы пользователя Linux // Сборник: Саратовская область: традиции, инновации, стратегии лидерства. Сборник научных трудов Всероссийской научно-практической конференции студентов, магистрантов, аспирантов, посвященной 80-летию Саратовской области. В 2-х частях. 2016. С. 194–195.
6. Пантелеев Н.Н., Панов С.С., Матвеев А.В. Сравнение характеристик и возможностей современных файловых систем Linux: EXT4, XFS, BTRFS // E-Scio. 2022. № 11 (74). С. 125–140.
7. Díaz A.F., Anguita M., Camacho H.E., Nieto E., Ortega Ju. Two-level hash/table approach for metadata management in distributed file systems // The Journal of Supercomputing. 2013. Т. 64. № 1. С. 144–155.
8. URL: https://www.unixtutorial.org/atime-ctime-mtime-in-unix-filesystems/
9. Chae M.S., Lee H.M., Lee K. A. Performance comparison of Linux containers and virtual machines using Docker and Kvm // Cluster Computing. 2019. Т. 22. № Suppl. 1. С. 1765–1775.
10. Захарчёнок В.Ф., Бизюк А.Н. Контейниризация и развертывание приложений с помощью Docker и Docker-compose // В сборнике: Материалы докладов 56-й международной научно-технической конференции преподавателей и студентов. в двух томах. Витебск, 2023. С. 80–82.
11. Громов Н.Д., Платошин А.И. Применение Docker при разработке ПО // Моя профессиональная карьера. 2024. Т. 2. № 60. С. 229–232.
12. Киселёв Д.В., Харламов Д.А. Работа пакетного менеджера Nix // В сборнике: Молодежь и научно-технический прогресс. Сборник докладов XV международной научно-практической конференции студентов, аспирантов и молодых ученых. В 2-х томах. Сост.: Е.Н. Иванцова, В.М. Уваров [и др.]. Губкин, 2022. С. 139–142.
13. Нагаев М.Т., Кожевникова А.В. Анализ эффективности реализации программ на golang // В сборнике: Новые информационные технологии и системы (НИТиС-2023). Сборник научных статей по материалам XX Международной научно-технической конференции, посвященной 80-летнему юбилею Пензенского государственного университета. Пенза, 2023. С. 160–168.
14. Корнеев В.В. Администрирование Linux. Серверное оборудование. - СПб.: БХВ-Петербург, 2021. - 560 с. ISBN 978-5-9775-6635-4.
15. Немет Э., Снайдер Г., Хейн Т.Р. Unix и Linux. Руководство системного администратора. 4-е изд. - М.: Вильямс, 2020. - 1312 с. ISBN 978-5-8459-2109-3.
16. Собель М. Linux. Администрирование и системное программирование. - М.: Вильямс, 2021. - 1088 с. ISBN 978-5-8459-2108-6.
17. Родригес К., Фишер Г., Смолка С. Python и анализ данных. - М.:ДМК Пресс, 2021. - 482 с. ISBN 978-5-97060-799-4.
18. Робачевский А.М. Операционная система UNIX. 2-е изд. - СПб.: БХВ-Петербург, 2020. - 656 с. ISBN 978-5-9775-4099-6.

References:

1. Chavan M.P., Chavan P.M. Rpm packaging for Ansible automation configuration management in Linux // Interantional journal of scientific research in engineering and management. 2022; 6.(12):1–3.
2. Irons L.G., Irons M.A. Terraform sustainability assessment framework for bioregenerative life support systems. *Frontiers in Astronomy and Space Sciences*. 2021; 8:789563.
3. Krivtsova I.E., Salakhutdinova K.I., Yurin I.V. Method of identification of executable files by their signatures. *Bulletin of the Admiral S.O. Makarov State University of Maritime and Inland Shipping*. 2016;1(35): 215–224.
4. Oreshchenkov I. Running modern programs on an outdated Linux distribution. *System administrator*. 2018; 12 (193): 22–28.
5. Zheryakov D.V. Basic software packages necessary for organizing the work of a Linux user // Collection: Saratov region: traditions, innovations, leadership strategies. Collection of scientific papers of the All-Russian scientific and practical conference of students, master's students, and postgraduate students dedicated to the 80th anniversary of the Saratov region. In 2 parts. 2016:194–195.
6. Panteleev N.N., Panov S.S., Matveev A.V. Comparison of characteristics and capabilities of modern Linux file systems: EXT4, XFS, BTRFS. *E-Scio*. 2022;11 (74):125–140. (In Russ).
7. Díaz A.F., Anguita M., Camacho H.E., Nieto E., Ortega Ju. Two-level hash/table approach for metadata management in distributed file systems. *The Journal of Supercomputing*. 2013; 64(1):144–155.
8. URL: https://www.unixtutorial.org/atime-ctime-mtime-in-unix-filesystems/
9. Chae M.S., Lee H.M., Lee K. A. Performance comparison of Linux containers and virtual machines using Docker and Kvm. *Cluster Computing*. 2019;22(Suppl.1):1765–1775.
10. Zakharchonok V.F., Bizyuk A.N. Containerization and deployment of applications using Docker and Docker-compose // In the collection: Proceedings of the 56th international scientific and technical conference of teachers and students. in two volumes. Vitebsk, 2023. pp. 80–82.
11. Gromov N.D., Platoshin A.I. Using Docker in software development. *My professional career*. 2024;2(. 60): 229–232.

12. Kiselev D.V., Kharlamov D.A. Work of the Nix package manager // In the collection: Youth and scientific and technological progress. Collection of reports of the XV international scientific and practical conference of students, graduate students, and young scientists. In 2 volumes. Comp.: E.N. Ivantsova, V.M. Uvarov [et al.]. Gubkin, 2022;139–142.
13. Nagaev M.T., Kozhevnikova A.V. Analysis of the efficiency of program implementation in golang // In the collection: New information technologies and systems (NITiS-2023). Collection of scientific articles based on the materials of the XX International scientific and technical conference dedicated to the 80th anniversary of Penza State University. Penza, 2023:160–168.
14. Korneev V.V. Linux Administration. Server equipment. - St. Petersburg: BHV-Petersburg, 2021:560 p. ISBN 978-5-9775-6635-4.
15. Nemeth E., Snyder G., Hein T.R. Unix and Linux. System Administrator's Guide. 4th ed. - Moscow: Williams, 2020:1312. ISBN 978-5-8459-2109-3.
16. Sobel M. Linux. Administration and system programming. - Moscow: Williams, 2021; 1088. ISBN 978-5-8459-2108-6.
17. Rodriguez C., Fisher G., Smolka S. Python and data analysis. - Moscow: DMK Press, 2021; 482. ISBN 978-5-97060-799-4.
18. Robachevsky A.M. Operating system UNIX. 2nd ed. - SPb.: BHV-Peterburg, 2020: 656 ISBN 978-5-9775-4099-6. (In Russ).

Сведения об авторе:

Попов Антон Дмитриевич, кандидат технических наук, доцент кафедры автоматизированных информационных систем органов внутренних дел; anton.holmes@mail.ru

Information about the author:

Anton D. Popov, Cand. Sci. (Eng.), Assoc. Prof., Department of automated information systems of internal organs; anton.holmes@mail.ru.

Конфликт интересов/Conflict of interest.

Автор заявляет об отсутствии конфликта интересов/The author declare no conflict of interest.

Поступила в редакцию/ Received 15.07.2025.

Одобрена после рецензирования/Reviced 30. 08.2025.

Принята в печать /Accepted for publication 24.10.2025.