

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ



**ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО
ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ ГОСУДАРСТВЕННЫЙ
ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»**

Институт

компьютерных наук

Кафедра

автоматизированных систем управления

ЛАБОРАТОРНАЯ РАБОТА №2

По дисциплине "Операционные системы Linux"

На тему "Работа с файловой системой ОС Linux"

Студент

ПИ-22-1

подпись, дата

Кистерёв В.А.

Руководитель

канд. техн. наук, доцент

ученая степень, ученое звание

подпись, дата

Кургасов В.В.

Липецк, 2024 г.

Оглавление

Цель работы	3
Ход работы	4
1. Общая часть	4
2. Файлы и каталоги	15
3. Пользователи и группы	18
4. Архивация и поиск	21
5. Подсистема инициализации и управления службам	23
Контрольные вопросы.....	25
Вывод.....	29

Цель работы

Приобрести опыт работы с файлами и каталогами в ОС Linux, настройки прав на доступ к файлам и каталогам.

Ход работы

1. Общая часть

1.1. Ознакомиться со структурой системных каталогов ОС Linux на рабочем месте. Изучить стандарт (Filesystem Hierarchy Standard).

FHS (Filesystem Hierarchy Standard) — это стандарт, определяющий расположение каталогов и файлов в операционных системах на базе UNIX, таких как Linux. Цель стандарта заключается в установлении структуры файловой системы, чтобы обеспечить совместимость программного обеспечения и упрощение администрирования.

Основные положения:

Корневой каталог / — корень файловой системы, где расположены все другие файлы и каталоги.

Каталог /bin — содержит необходимые исполняемые файлы, которые используются как обычными пользователями, так и системными процессами. Например, команды вроде ls, cp, mv.

Каталог /sbin — системные утилиты, которые предназначены для выполнения задач администратора (например, управление системой).

Каталог /usr — хранит второстепенные программы и библиотеки. Он часто используется для размещения программ, не требующихся в режиме восстановления системы.

Каталог /lib — библиотеки, необходимые для запуска программ в /bin и /sbin.

Каталог /etc — файлы конфигурации системы.

Каталог /var — файлы, которые изменяются во время работы системы, например, логи, временные файлы.

Каталог /tmp — временные файлы, которые удаляются после перезагрузки системы.

1.2. Изучить и привести в отчёте перечень основных каталогов с указанием их назначения.

Командой `ls -l /` (/ – корневая директория) посмотрим перечень основных каталогов (рисунок 1).

```
root@debian:~# ls -l /
итого 60
lrwxrwxrwx  1 root root    7 окт  6 11:11 bin -> usr/bin
drwxr-xr-x  3 root root 4096 окт  6 11:19 boot
drwxr-xr-x 17 root root 3340 окт 12 18:37 dev
drwxr-xr-x 71 root root 4096 окт 12 18:37 etc
drwxr-xr-x  7 root root 4096 окт  7 20:14 home
lrwxrwxrwx  1 root root    30 окт  6 11:14 initrd.img -> boot/initrd.img-6.1.0-26-amd64
lrwxrwxrwx  1 root root    30 окт  6 11:12 initrd.img.old -> boot/initrd.img-6.1.0-25-amd64
lrwxrwxrwx  1 root root    7 окт  6 11:11 lib -> usr/lib
lrwxrwxrwx  1 root root    9 окт  6 11:11 lib64 -> usr/lib64
drwx----- 2 root root 16384 окт  6 11:11 lost+found
drwxr-xr-x  3 root root 4096 окт  6 11:11 media
drwxr-xr-x  2 root root 4096 окт  6 11:11 mnt
drwxr-xr-x  2 root root 4096 окт  6 11:11 opt
dr-xr-xr-x 151 root root    0 окт 12 18:37 proc
drwx----- 4 root root 4096 окт 11 18:43 root
drwxr-xr-x 17 root root  520 окт 12 18:37 run
lrwxrwxrwx  1 root root    8 окт  6 11:11/sbin -> usr/sbin
drwxr-xr-x  2 root root 4096 окт  6 11:11 srv
dr-xr-xr-x 13 root root    0 окт 12 18:37 sys
drwxrwxrwt  8 root root 4096 окт 12 18:37 tmp
drwxr-xr-x 12 root root 4096 окт  6 11:11 usr
drwxr-xr-x 11 root root 4096 окт  6 11:11 var
lrwxrwxrwx  1 root root   27 окт  6 11:14 vmlinuz -> boot/vmlinuz-6.1.0-26-amd64
lrwxrwxrwx  1 root root   27 окт  6 11:12 vmlinuz.old -> boot/vmlinuz-6.1.0-25-amd64
root@debian:~#
```

Рисунок 1 – Перечень основных каталогов

Назначение каталогов:

Каталог **/bin** содержит основные исполняемые файлы (бинарные программы), которые необходимы как для обычных пользователей, так и для системных процессов. В него входят такие команды, как `ls`, `cp`, `mv` и другие.

Каталог **/boot** хранит файлы, необходимые для загрузки системы, включая ядро Linux и начальные образы загрузки (например, файлы `vmlinuz` и `initrd.img`).

Каталог **/dev** содержит файлы, представляющие физические и виртуальные устройства, такие как жесткие диски, порты ввода/вывода, виртуальные устройства и другие.

Каталог **/etc** предназначен для хранения конфигурационных файлов системы. Здесь находятся настройки различных служб, программ и компонентов системы.

Каталог **/home** хранит домашние каталоги всех пользователей системы, где размещаются их личные файлы и настройки.

Файл **initrd.img** используется на этапе загрузки системы для создания временной файловой системы, необходимой перед запуском основного ядра.

Файл **initrd.img.old** — это предыдущая версия **initrd.img**, используемая для восстановления в случае проблем с текущим ядром.

Каталог **/lib** содержит системные библиотеки, которые используются программами и исполняемыми файлами в **/bin** и **/sbin**.

Каталог **/lib64** — это версия каталога библиотек, которая используется в 64-битных системах для хранения специфических библиотек.

Каталог **/lost+found** используется для хранения поврежденных или утраченных файлов, которые могут быть восстановлены после сбоя файловой системы.

Каталог **/media** содержит точки монтирования съемных носителей, таких как CD-ROM, USB-накопители и другие.

Каталог **/mnt** используется для временного монтирования файловых систем.

Каталог **/opt** предназначен для установки дополнительных программ, не входящих в стандартные пакеты системы.

Каталог **/proc** представляет собой виртуальную файловую систему, которая предоставляет информацию о запущенных процессах и конфигурации ядра. Она динамически обновляется системой.

Каталог **/root** — это домашний каталог пользователя root (администратора системы), где он хранит свои личные файлы.

Каталог **/run** содержит временные данные, которые используются процессами во время работы системы, и эти данные не сохраняются после перезагрузки.

Каталог **/sbin** хранит системные программы и утилиты, которые обычно используются администраторами для управления системой.

Каталог **/srv** содержит данные, которые используются сервисами и серверами, запущенными на данной системе.

Каталог **/sys** предоставляет информацию о подключенных устройствах, драйверах и некоторых параметрах ядра, что также реализовано как виртуальная файловая система.

Каталог **/tmp** используется для временного хранения файлов. Содержимое этого каталога обычно удаляется при перезагрузке системы.

Каталог **/usr** — это место для хранения пользовательских программ, библиотек и документации, которые не являются критическими для загрузки системы.

Каталог **/var** содержит переменные данные, такие как журналы системы, буферы, временные файлы и кэш, которые могут изменяться во время работы системы.

Файл **vmlinuz** — это сжатое ядро Linux, которое загружается во время старта системы.

Файл **vmlinuz.old** — это предыдущая версия ядра, которая сохраняется для того, чтобы можно было вернуться к старому ядру, если новое работает нестабильно.

1.3. Зайти в терминал под root.

Командой **su** зайдём в терминал под root (рисунок 1).

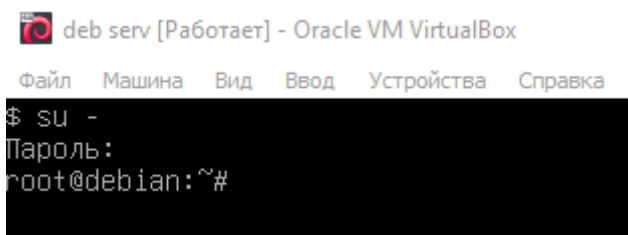


Рисунок 2 – Вход в терминал под пользователем root

1.4. Просмотреть содержимое каталога файлов физических устройств. В отчете привести перечень файлов физических устройств на рабочем месте с указанием назначения файлов.

С помощью команды **ls /dev** посмотрим содержимое каталога файлов физических устройств (рисунок 3).

```
root@debian:~# ls /dev
autofs      cuse      hwrng      loop5      port      sda1      stderr    tty14      tty23      tty32      tty41      tty50      tty6      ttyS2      vcs2      vcsa5      vga_arbiter
block       disk      initctl    loop6      ppp       sda2      stdin     tty15      tty24      tty33      tty42      tty51      tty60      ttyS3      vcs3      vcsa6      vhci
bsg         dri       input      loop7      psaux     sda5      stdout    tty16      tty25      tty34      tty43      tty52      tty61      uhid      vcs4      vcsu      vhost-net
cifsfs-control fb0       kmsg       loop-control ptmx      sda6      tty       tty17      tty26      tty35      tty44      tty53      tty62      uinput    vcs5      vcsu1     vhost-vsock
bus         fd        log        mapper     pts       sg0       tty0       tty18      tty27      tty36      tty45      tty54      tty63      urandom   vcs6      vcsu2     zero
cdrom       full      loop0      mem        random    sg1       tty1       tty19      tty28      tty37      tty46      tty55      tty7      userfaultfd vcsa      vcsu3
char        fuse      loop1      mqueue     rfkill    shm       tty10      tty2       tty29      tty38      tty47      tty56      tty8      vboxguest  vcsa1     vcsu4
console     hidraw0   loop2      net        rtc       snapshot  tty11      tty20      tty3       tty39      tty48      tty57      tty9      vboxuser   vcsa2     vcsu5
core        hpet      loop3      null       rtc0      snd       tty12      tty21      tty30      tty4       tty49      tty58      tty30     vcs       vcsa3     vcsu6
cpu_dma_latency hugepages loop4      nvram      sda       sr0       tty13      tty22      tty31      tty40      tty5       tty59      ttyS1     vcs1      vcsa4     vfi0
root@debian:~#
```

Рисунок 3 – Содержимое каталога dev

Назначение файлов:

Файл **console** представляет устройство, на которое отправляются системные сообщения и где происходит авторизация в режиме единственного пользователя.

Каталог **dri** содержит интерфейс, предоставляющий возможность пользовательским приложениям получать прямой доступ к видеоустройствам, что используется для ускорения работы с графикой.

Файл **fb0** — это первый кадровый буфер, абстрактный слой между программным обеспечением и графическим оборудованием. Он обеспечивает отображение графической информации на экране.

Файл **fd** представляет дисковод для гибких дисков (дискет).

Файл **full** является специальным устройством, запись в которое всегда вызывает ошибку "недостаточно места". Оно используется для тестирования программ на обработку ошибок.

Файлы **loop0-7** позволяют монтировать файлы как виртуальные блочные устройства, отображая их как отдельные файловые системы. Это полезно при работе с образами дисков.

Файл **mem** представляет собой специальный файл, который является образом физической памяти компьютера, позволяя получать доступ к содержимому памяти напрямую.

Файл **null** — это пустое устройство, куда можно записывать данные, но они будут немедленно "теряться". Оно используется для отбрасывания ненужного вывода.

Файл **ptmx** — мультиплексорное устройство псевдотерминала. Он используется для создания пары основного и подчиненного псевдотерминалов, необходимых для работы терминалов.

Каталог **pts** представляет собой виртуальный каталог для временных файлов, связанных с псевдотерминалами.

Файл **random** является псевдоустройством, предоставляющим доступ к системному генератору случайных чисел, который используется для криптографии и других задач.

Файл **urandom** — это псевдоустройство, которое предоставляет интерфейс к системному генератору псевдослучайных чисел, быстрее, но менее энтропийное, чем **random**.

Файл **rftkill** предоставляет интерфейс для управления радиопередатчиками, такими как WiFi и Bluetooth, позволяя включать и отключать их.

Файл **rtc** предоставляет интерфейс для работы с драйверами часов реального времени (RTC), которые обеспечивают точное отслеживание времени на системном уровне.

Файл **sda** представляет собой устройство, которое предоставляет доступ к жестким дискам с интерфейсом SCSI. Буквы после **sda** указывают на новые устройства, а числа после них обозначают номера разделов.

Файл **sr0** представляет привод оптических дисков, таких как CD или DVD.

Файл **stderr** хранит стандартный поток ошибок, который используется для вывода сообщений об ошибках в программах.

Файл **stdin** хранит стандартный поток ввода, который используется для передачи данных программам.

Файл **stdout** хранит стандартный поток вывода, куда программы записывают свои результаты.

Файл **tty** представляет собой устройства, связанные с пользовательскими терминалами (консолями).

Файлы **ttyS** представляют собой устройства для работы с последовательными портами, используемыми для подключения модемов и другого оборудования.

Файл **zero** является специальным устройством, которое всегда предоставляет нулевые байты при чтении.

1.5. Перейти в директорию пользователя root. Просмотреть содержимое каталога. Просмотреть содержимое файла `vmlinuz`. Просмотреть и пояснить права доступа к файлу `vmlinuz`.

В домашнем каталоге пользователя root посмотрим содержимое и права доступа файла `vmlinuz`. Для этого воспользуемся командой `ls -l /boot/vmlinuz-6.1.0-25-amd64`, представленной на рисунке 4.

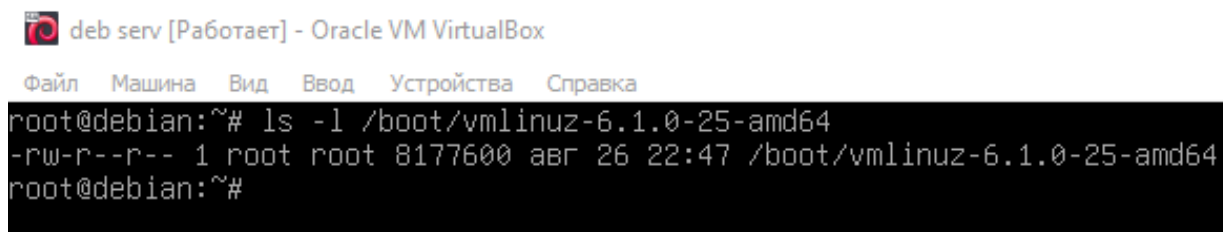


Рисунок 4 – Просмотр прав доступа файла `vmlinuz`

`vmlinuz` — это сжатое ядро Linux, которое загружается при старте системы. Файл `vmlinuz` играет ключевую роль в процессе загрузки операционной системы. При загрузке компьютер читает этот файл, распаковывает его и загружает ядро Linux в память, чтобы система могла начать работу. Он содержит всю логику ядра, которая управляет процессами, памятью, устройствами и всеми основными аспектами операционной системы.

Пояснение прав доступа:

Права `-rw-r--r--` позволяют владельцу читать и изменять файл, а остальным пользователям — только читать его. Выполнение файла запрещено для всех, потому что `vmlinuz` не является исполняемым пользователем файлом в привычном смысле (например, как программа). Это сжатое ядро Linux, которое загружается на этапе загрузки системы специальным загрузчиком (таким как GRUB). Загрузчик распаковывает и запускает ядро, поэтому нет необходимости предоставлять право выполнения пользователям напрямую.

1.6. Создать нового пользователя root.

Создадим нового пользователя `user` командой `useradd (-m: создание домашнего каталога пользователя)`, командой `passwd` зададим новому пользователю пароль, командой `su` переключимся на созданного пользователя. Пример использования команд представлен на рисунке 5.

```

root@debian:/# useradd -m user
root@debian:/# passwd user
Новый пароль:
Повторите ввод нового пароля:
passwd: пароль успешно обновлён
root@debian:/# su - user
$ _

```

Рисунок 5 – Пример использования команд useradd, passwd и su

1.7. Создать в директории пользователя user три файла 1.txt, 2.txt и 3.txt, используя команды touch, cat и текстовый редактор (на выбор vi/nano). Просмотреть и пояснить права доступа к файлам.

Командами touch, cat и nano создадим 3 файла. Пример создания представлен на рисунке 6.

```

$ touch 1.txt
$ cat > 2.txt
$ nano 3.txt

```

Рисунок 6 – Пример создания файлов

Командой ls с ключом -l выведем подробный список файлов и их атрибутов с правами доступа (рисунок 7).

```

$ ls -l
итого 0
-rw-r--r-- 1 user user 0 окт 6 12:05 1.txt
-rw-r--r-- 1 user user 0 окт 6 12:05 2.txt
-rw-r--r-- 1 user user 0 окт 6 12:06 3.txt

```

Рисунок 7 – Вывод списка файлов

Пояснение прав доступа для созданных файлов:

Первая группа (для владельца файла): rw-

r: Право на чтение.

w: Право на запись.

-: Нет права на выполнение.

Вторая группа (для группы): r--

r: Право на чтение.

-: Нет права на запись.

-: Нет права на выполнение.

Третья группа (для остальных пользователей): r--

r: Право на чтение.

-: Нет права на запись.

-: Нет права на выполнение.

1.8. Перейти в директорию пользователя root. В отчёте описать результат.

Командой `su` - переключимся на пользователя root, командой `ls -a` выведем содержимое директории пользователя root (рисунок 8).

```
$ su -  
Пароль:  
root@debian:~# ls -a  
.  ..  .bash_history  .bashrc  .profile  .ssh  
root@debian:~#
```

Рисунок 8 – Вывод содержимого директории

Содержимое директории:

`.bash_history` – файл хранит историю команд, введённых в оболочке `bash`. Он позволяет пользователю просматривать ранее выполненные команды.

`.bashrc` – скрипт, который выполняется каждый раз при запуске интерактивной оболочки `bash`. Он используется для настройки окружения пользователя, например, для установки переменных среды и настройки функций.

`.profile` – файл загружается при входе в систему и предназначен для настройки пользовательского окружения. Он может содержать настройки, специфичные для пользователя, такие как переменные среды и пути к программам.

`.ssh` – директория, которая содержит файлы, связанные с SSH (Secure Shell). Она обычно включает файлы аутентификации, такие как ключи и конфигурации для подключения к удалённым системам.

1.9. Изменить права доступа на файл `1.txt` в директории пользователя `user`.

Командой `chmod 777` (полные права доступа на файл для владельца, группы и остальных пользователей) изменим права доступа на файл `1.txt` (рисунок 9).

```
root@debian:/home/user# chmod 777 1.txt  
root@debian:/home/user# ls -l  
итого 0  
-rwxrwxrwx 1 user user 0 окт  6 12:05 1.txt  
-rw-r--r-- 1 user user 0 окт  6 12:05 2.txt  
-rw-r--r-- 1 user user 0 окт  6 12:06 3.txt  
root@debian:/home/user# _
```

Рисунок 9 – Изменение прав доступа

1.10. Создать жесткую и символическую ссылки на файл 2.txt.

Просмотреть и описать полученные результаты.

Жёсткие ссылки создаются с помощью команды `ln`. Они указывают на один и тот же inode в файловой системе, что означает, что обе ссылки указывают на один и тот же файл. Если файл, на который ссылается жёсткая ссылка, будет удалён, данные останутся доступными через жёсткую ссылку.

Символические ссылки создаются с помощью команды `ln` с ключом `-s`. Они являются отдельными файлами, которые содержат путь к оригинальному файлу. Если оригинальный файл будет удалён, символическая ссылка станет недействительной.

Пример создания жёсткой и символической ссылки на файл 2.txt представлен на рисунке 10.

```
$ ln 2.txt hard_link.txt
$ ln -s 2.txt symbolic_link.txt
$ ls -a
.  ..  1.txt  2.txt  3.txt  .bash_logout  .bashrc  hard_link.txt  .local  .profile  symbolic_link.txt
$ _
```

Рисунок 10 – Создание ссылок

1.11. Создать каталог new в каталоге пользователя user.

Командой `mkdir` создадим новый каталог new в директории пользователя user (рисунок 11).

```
$ mkdir new
$ ls -a
.  ..  1.txt  2.txt  3.txt  .bash_logout  .bashrc  hard_link.txt  .local  new  .profile  symbolic_link.txt
$ _
```

Рисунок 11 – Создание нового каталога

1.12. Скопировать файл 1.txt в каталог new.

Командой `cp` скопируем файл 1.txt в каталог new (рисунок 12).

```
$ cp 1.txt new
$ cd new
$ ls -a
.  ..  1.txt
$ _
```

Рисунок 12 – Копирование файла

1.13. Переместить файл 2.txt в каталог new.

Командой `mv` переместим файл 2.txt в каталог new (рисунок 13).

```
$ mv 2.txt new
$ ls -a
.  ..  1.txt  3.txt  .bash_logout  .bashrc  hard_link.txt  .local  new  .profile  symbolic_link.txt
$ cd new
$ ls -a
.  ..  1.txt  2.txt
$ _
```

Рисунок 13 – Перемещение файла

1.14. Изменить владельца файла 3.txt и каталога new.

Командой `chown` изменим владельца файла 3.txt и каталога new на root (рисунок 14).

```
root@debian:/home/user# chown root 3.txt
root@debian:/home/user# chown root new
root@debian:/home/user# ls -l
итого 4
-rwxrwxrwx 1 user user    0 окт  6 12:05 1.txt
-rw-r--r-- 1 root user    0 окт  6 12:06 3.txt
-rw-r--r-- 2 user user    0 окт  6 12:05 hard_link.txt
drwxr-xr-x 2 root user 4096 окт  6 12:40 new
lrwxrwxrwx 1 user user    5 окт  6 12:33 symbolic_link.txt -> 2.txt
root@debian:/home/user# _
```

Рисунок 14 – Изменение владельца файла

1.15. Удалить файл 1.txt в каталоге new.

Для удаления файла 1.txt в каталоге new используем команду `rm` (рисунок 15).

```
root@debian:/home/user# cd new
root@debian:/home/user/new# rm 1.txt
root@debian:/home/user/new# ls -a
.  ..  2.txt
root@debian:/home/user/new# _
```

Рисунок 15 – Удаление файла

1.16. Удалить каталог new.

Для удаления каталога new используем команду `rm -r` (рекурсивно, для удаления вместе с содержимым). Пример использования команды представлен на рисунке 16.

```
root@debian:/home/user/new# cd ..
root@debian:/home/user# rm -r new
root@debian:/home/user# ls -a
.  ..  1.txt  3.txt  .bash_logout  .bashrc  hard_link.txt  .local  .profile  symbolic_link.txt
root@debian:/home/user#
```

Рисунок 16 – Удаление каталога

2. Файлы и каталоги

2.1. Создайте 3 текстовых файла разными способами: посредством редакторов vi, mc и команды tee (предварительно изучите справку по команде). Файлы должны содержать от 5 до 8 строк осмысленного текста (например, стихи).

Создадим 3 текстовых файла редакторами vi, mc и команды tee, в файлы внесём текст стихотворения. Пример создания текстового файла 1.txt редактором vi представлен на рисунке 17.

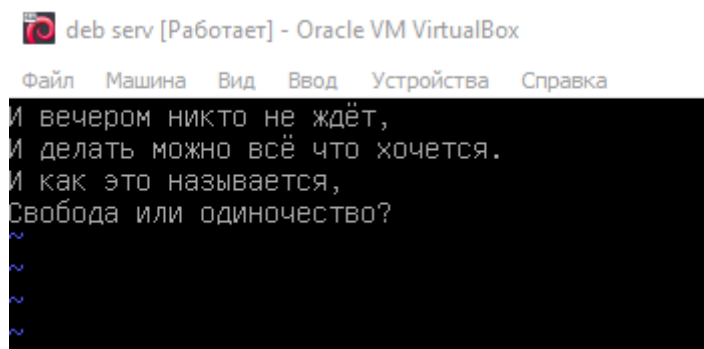


Рисунок 17 – Создание файла в редакторе nano

Для сохранения файла и выхода из редактора нужно нажать Esc для перехода в режим команд, а после – :wq.

2.2. Создайте структуру каталогов в соответствии с вариантом.

Создадим структуру каталогов в соответствии с вариантом 2 (рисунок 18). Файлы создадим копированием ранее созданных файлов командой cp. Ссылки создаются командой ln, символические ссылки командой ln с ключом -s.

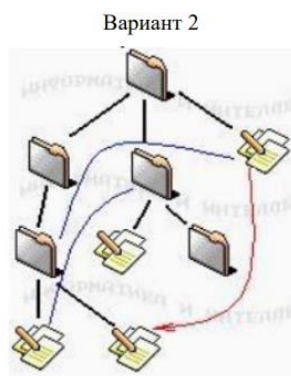


Рисунок 18 – Структура каталогов для варианта 2

Процесс создания структуры каталогов и древовидная форма созданной структуры представлена на рисунке 19.

```

$ ls -a
.  ..  1.txt  2.txt  3.txt  .bash_logout  .bashrc  .cache  .config  .local  .profile  .selected_editor
$ mkdir 1
$ cd 1
$ mkdir 2
$ cd 2
$ mkdir 4
$ cd ../..
$ cp 1.txt 1/2/4
$ cp 2.txt 1/2/4
$ cd 1
$ mkdir 3
$ cd ../
$ cp 3.txt 1/3
$ cd 1/3
$ mkdir 5
$ ln ../2/4/1.txt 1_hard.txt
$ cd ../
$ ln -s 2/4/2.txt 2_s.txt
$ cd 2/4
$ ln ../../2_s.txt 2_s_hard.txt
$ cd ../..
$ tree 1
1
├── 2
│   ├── 4
│   │   ├── 1.txt
│   │   ├── 2_s_hard.txt -> 2/4/2.txt
│   │   └── 2.txt
│   └── 2_s.txt -> 2/4/2.txt
├── 3
│   ├── 1_hard.txt
│   ├── 3.txt
│   └── 5
└── 5 directories, 6 files

```

Рисунок 19 – Процесс создания структуры каталогов

2.3. Провести ряд экспериментов, иллюстрирующих доступ к файлам по основным именам, по ссылкам и по символическим ссылкам. Для доступа использовать команду `cat` или редактор `vi`.

По символической ссылке `2_s.txt` попробуем командой `cat` вывести содержимое файла `2.txt` (рисунок 20).

```

$ cd 1
$ ls -a
.  ..  2  2_s.txt  3
$ cat 2_s.txt
И вечером никто не ждёт,
И делать можно всё что хочется.
И как это называется,
Свобода или
$ _

```

Рисунок 21 – Пример обращения по символической ссылке

По жёсткой ссылке `1_hard.txt` попробуем командой `cat` вывести содержимое файла `1.txt` (рисунок 22).

```

$ cat 1_hard.txt
И вечером никто не ждёт,
И делать можно всё что хочется.
И как это называется,
Свобода или одиночество?
$ _

```

Рисунок 22 – Пример обращения по жёсткой ссылке

2.4. Провести ряд экспериментов, иллюстрирующих реакцию системы на удаление файла, на который имеются ссылки, и файла, на который имеются символические ссылки. Проверять результаты командой `ls -la`.

Удалим файл `1.txt`, на который есть жёсткая ссылка `1_hard.txt`, командой `ls -la` посмотрим результат удаления (рисунок 23).

```
$ rm 1.txt
$ cd ../../3
$ ls -la
итого 20
drwxr-xr-x 3 user user 4096 окт 6 13:49 .
drwxr-xr-x 4 user user 4096 окт 6 13:51 ..
-rw-r--r-- 1 user user 186 окт 6 13:46 1_hard.txt
-rw-r--r-- 1 user user 100 окт 6 13:48 3.txt
drwxr-xr-x 2 user user 4096 окт 6 13:48 5
$ cat 1_hard.txt
И вечером никто не ждёт,
И делать можно всё что хочется.
И как это называется,
Свобода или одиночество?
$
```

Рисунок 23 – Результат удаления

Жёсткая ссылка по-прежнему работает.

Удалим файл `2.txt`, на который есть символическая ссылка `2_s.txt`, командой `ls -la` посмотрим результат удаления (рисунок 24).

```
$ cat 2_s.txt
cat: 2_s.txt: Нет такого файла или каталога
$ ls -la
итого 16
drwxr-xr-x 4 user user 4096 окт 6 13:51 .
drwxr-xr-x 6 user user 4096 окт 6 13:45 ..
drwxr-xr-x 3 user user 4096 окт 6 13:46 2
lrwxrwxrwx 2 user user 9 окт 6 13:51 2_s.txt -> 2/4/2.txt
drwxr-xr-x 3 user user 4096 окт 6 13:49 3
$
```

Рисунок 24 – Результат удаления

Символическая ссылка остается, но больше не имеет действительного объекта, на который она указывает.

2.5. Уничтожить созданные подкаталоги и файлы в них, сохранив исходные 3 файла.

Командой `rm -r 1` удалим созданные каталоги и их содержимое (рисунок 25).

```
$ rm -r 1
$ ls -a
.  ..  1.txt  2.txt  3.txt  .bash_logout  .bashrc  .cache  .config  .local  .profile  .selected_editor
$
```

Рисунок 25 – Удаление подкаталогов

3. Пользователи и группы

3.1. Создайте пользователя в ОС UBUNTU с именем <Ваше Имя Группа>.

Создайте пользователя в ОС UBUNTU с именем <Фамилия Имя Отца>.

Создадим пользователей VictorPI221 и KisterevAlex командой useradd, установим им пароли командой passwd (рисунок 26).

```
root@debian:~# useradd -m VictorPI221
root@debian:~# passwd VictorPI221
Новый пароль:
Повторите ввод нового пароля:
passwd: пароль успешно обновлён
root@debian:~# useradd -m KisterevAlex
root@debian:~# passwd KisterevAlex
Новый пароль:
Повторите ввод нового пароля:
passwd: пароль успешно обновлён
root@debian:~#
```

Рисунок 26 – Пример создания пользователей

3.2. Войдите в систему под созданным пользователем <Ваше Имя Группа>.

Командой su войдём в систему под пользователем VictorPI221 (рисунок 27).

```
root@debian:~# su - VictorPI221
$
```

Рисунок 27 – Пример использования команды su

3.2.1. Создайте файл с именем: <ВашеИмяГруппа>. Откройте созданный файл в удобном вам текстовом редакторе (Vi/VIM/NANO/ Sublime) etc. Внесите в него текст: echo "This is test page <Ваше ФИО>". Сохраните изменения в файле.

Создадим файл с именем VictorPI221 командой nano, внесём в него текст echo "This is test page Кистерёв Виктор Александрович" (рисунок 28), сохраним файл.

```
GNU nano 7.2
echo "This is test page Кистерёв Виктор Александрович"
```

Рисунок 28 – Создание файла в редакторе nano

Командой chmod добавим права на изменение файла VictorPI221 другими пользователями (рисунок 29).

```
$ chmod 646 VictorPI221
$ _
```

Рисунок 29 – Добавление прав на изменение файла

3.3. Переместите файл <ВашеИмяГруппа> в домашний каталог пользователя <ФамилияИмяОтца>.

Командой mv перенесём файл VictorPI221 в домашний каталог пользователя KisterevAlex от пользователя root (рисунок 30).

```
$ su
Пароль:
root@debian:/home/VictorPI221# mv VictorPI221 ../KisterevAlex
```

Рисунок 31 – Перенос файла командой mv

3.4. Выполните вход в систему от имени пользователя <ФамилияИмяОтца>.

Командой su войдём в систему под пользователем KisterevAlex (рисунок 32).

```
root@debian:/home/KisterevAlex# su - KisterevAlex
$ _
```

Рисунок 32 – Вход в систему под другим пользователем

3.4.1. Откройте файл, перемещённый туда в пункте 3. Добавьте в файл строчку: echo "Test page edited by user" <ФамилияИмяОтца> и сохраните изменения.

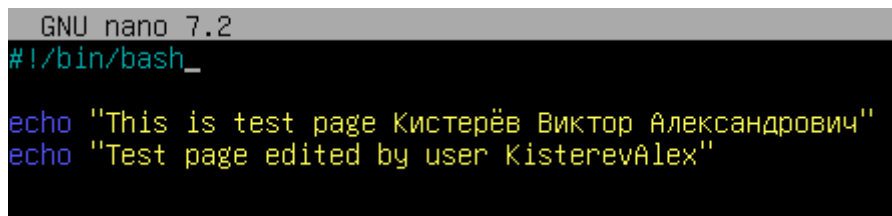
Откроем файл VictorPI221 командой nano, добавим текст echo "Test page edited by user KisterevAlex" (рисунок 33).

```
GNU nano 7.2
echo "This is test page Кистерёв Виктор Александрович"
echo "Test page edited by user KisterevAlex"
```

Рисунок 33 – Редактирование файла в редакторе nano

3.4.2. Переместите файл обратно в папку пользователя «ВашеИмяГруппа». Добавьте в начало документа следующий текст: #!/bin/bash.

Командой mv перенесём файл VictorPI221 обратно в папку пользователя VictorPI221 (от пользователя root), добавим текст #!/bin/bash (рисунок 334).

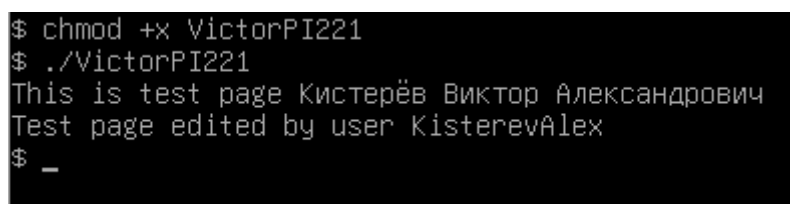


```
GNU nano 7.2
#!/bin/bash_
echo "This is test page Кистерёв Виктор Александрович"
echo "Test page edited by user KisterevAlex"
```

Рисунок 34 – Редактирование файла в редакторе nano

3.5. Зайдите в систему снова пользователем <ВашеИмяГруппа>, сделайте файл исполняемым и запустите.

Командой `su` войдём в систему снова под пользователем VictorPI221, сделаем файл исполняемым командой `chmod` и запустим его (рисунок 35).



```
$ chmod +x VictorPI221
$ ./VictorPI221
This is test page Кистерёв Виктор Александрович
Test page edited by user KisterevAlex
$ _
```

Рисунок 35 – Пример запуска исполняемого файла

4. Архивация и поиск

Создать архив arh2.tar.gz, состоящий из нескольких файлов. Вывести список файлов архива. Вывести содержимое файлов архива, без его распаковки. Распаковать архив. В указанном каталоге (без обработки подкаталогов) найти все обычные файлы, имеющие расширение.

4.1. Создадим папку с файлом, добавим эту папку в архив командой `tar -czf` (`c` — создать новый архив, `z` — указывает на то, что архив сжат с помощью `gzip`, `f` — указывает имя файла архива) (рисунок 36).

```
$ mkdir 1
$ cd 1
$ touch test.txt
$ cd ../
$ tar -czf arh2.tar.gz 1
$ ls -a
.  ..  1  arh2.tar.gz  .bash_logout  .bashrc  .local  .profile
$ _
```

Рисунок 36 – Добавление папки в архив

4.2. Командой `tar -tzf` (`t` — отображает содержимое архива, `z` — указывает на то, что архив сжат с помощью `gzip`, `f` — указывает имя файла архива) выведем содержимое архива (рисунок 37).

```
$ tar -tzf arh2.tar.gz
1/
1/test.txt
$ _
```

Рисунок 37 – Вывод содержимого архива

4.3. Командой `tar -xzf` (`x` — распаковывает файлы из архива, `z` — указывает на то, что архив сжат с помощью `gzip`, `f` — указывает имя файла архива) распакуем архив (рисунок 38).

```
$ rm -r 1
$ ls -a
.  ..  arh2.tar.gz  .bash_logout  .bashrc  .local  .profile  VictorPI221
$ tar -xzf arh2.tar.gz
$ ls -a
.  ..  1  arh2.tar.gz  .bash_logout  .bashrc  .local  .profile  VictorPI221
$
```

Рисунок 38 – Распаковка архива

4.4. Командой `find 1 -maxdepth 1 -type f -name "*.*"` (`1` — путь к каталогу; `-maxdepth 1` — ограничивает поиск только текущим каталогом (без

подкаталогов); -type f — ищет только обычные файлы; -name "*.*)" — ищет файлы, содержащие точку в имени (файлы с расширением)) выполним поиск по каталогу (рисунок 39).

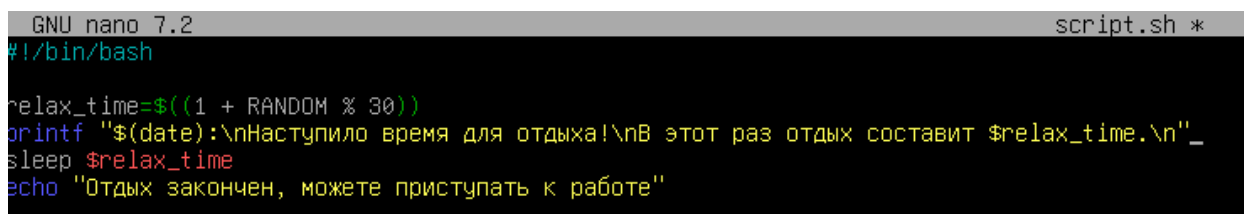
```
$ find 1 -maxdepth 1 -type f -name "*.*)"
1/test.txt
$ _
```

Рисунок 39 – Выполнение поиска по каталогу

5. Подсистема инициализации и управления службам

Написать демон, представляющий собой программу для отдыха глаз. Демон будет показывать уведомление о начале отдыха раз в заданный промежуток времени (repetition period) и уведомление об окончании отдыха через заданный промежуток времени (relax time). Временные промежутки задаются случайным образом.

С помощью редактора nano напишем скрипт для демона, представленный на рисунке 40.



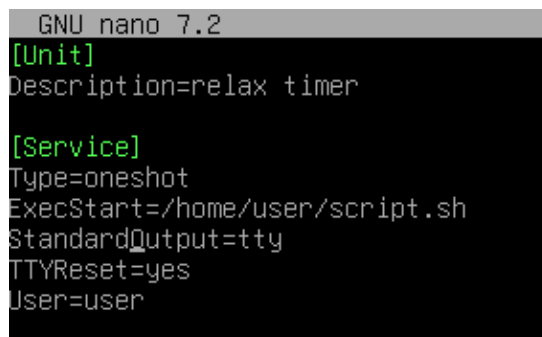
```
GNU nano 7.2 script.sh *
#!/bin/bash

relax_time=$((1 + RANDOM % 30))
printf "$(date):\nНаступило время для отдыха!\nВ этот раз отдых составит $relax_time.\n"
sleep $relax_time
echo "Отдых закончен, можете приступить к работе"
```

Рисунок 40 – Скрипт для демона

Сохраним скрипт и командой `chmod u+x` сделаем его исполняемым.

Создадим сервис для запуска демона в директории `/etc/systemd/system/`. Содержимое файла `relax.service` представлено на рисунке 41.



```
GNU nano 7.2
[Unit]
Description=relax timer

[Service]
Type=oneshot
ExecStart=/home/user/script.sh
StandardOutput=tty
TTYReset=yes
User=user
```

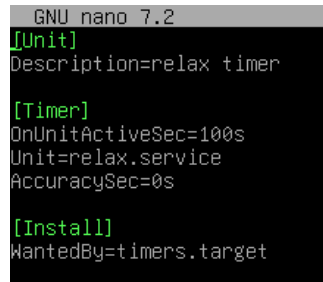
Рисунок 41 – Созданный сервис

Описание файла `relax.service`:

[Unit] – секция для описания юнита, Description – краткое описание юнита.

[Service] – секция с параметрами запуска службы, Type – тип службы, ExecStart – запуск службы (исполняемый файл), StandardOutput – управление выводом стандартного потока (tty – вывод в терминал), TTYReset – сброс терминальных настроек перед запуском сервиса (предотвращает случайное форматирование текста). User – пользователь, от чьего имени будет запущен исполняемый файл.

Создадим таймер `relax.timer` для запуска сервис через определенный промежуток времени. Пример созданного таймера представлен на рисунке 42.



```
GNU nano 7.2
[Unit]
Description=relax timer

[Timer]
OnUnitActiveSec=100s
Unit=relax.service
AccuracySec=0s

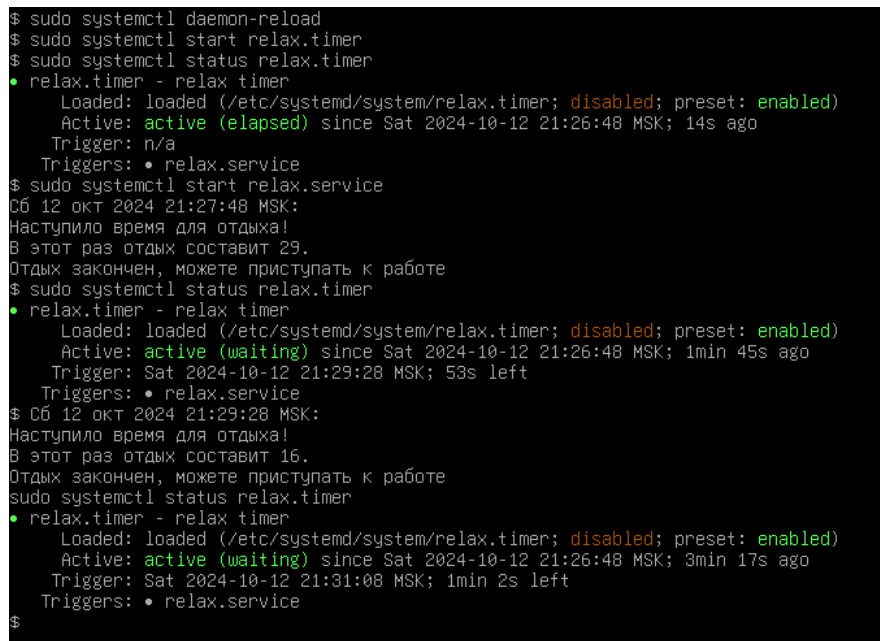
[Install]
WantedBy=timers.target
```

Рисунок 42 – Созданный таймер

Описание таймера:

[Timer] – секция для задания времени активации, OnUnitActiveSec – отсчет относительно момента запуска юнита, Unit – ссылка на сервис, который будет запущен таймером, AccuracySec – точность таймера, [Install] – секция для установки таймера, WantedBy – определяет таргет, при активации которого будет запущен таймер.

Командой `sudo systemctl daemon-reload` перечитаем все файлы конфигурации юнитов. Запустим таймер с помощью команды `sudo systemctl start relax.timer`, после этого запустим сервис командой `sudo systemctl start relax.service`. Пример использования команд и работы демона представлен на рисунке 43.



```
$ sudo systemctl daemon-reload
$ sudo systemctl start relax.timer
$ sudo systemctl status relax.timer
• relax.timer - relax timer
  Loaded: loaded (/etc/systemd/system/relax.timer; disabled; preset: enabled)
  Active: active (elapsed) since Sat 2024-10-12 21:26:48 MSK; 14s ago
  Trigger: n/a
  Triggers: • relax.service
$ sudo systemctl start relax.service
Сб 12 окт 2024 21:27:48 MSK:
Наступило время для отдыха!
В этот раз отдых составит 29.
Отдых закончен, можете приступить к работе
$ sudo systemctl status relax.timer
• relax.timer - relax timer
  Loaded: loaded (/etc/systemd/system/relax.timer; disabled; preset: enabled)
  Active: active (waiting) since Sat 2024-10-12 21:26:48 MSK; 1min 45s ago
  Trigger: Sat 2024-10-12 21:29:28 MSK; 53s left
  Triggers: • relax.service
$ Сб 12 окт 2024 21:29:28 MSK:
Наступило время для отдыха!
В этот раз отдых составит 16.
Отдых закончен, можете приступить к работе
$ sudo systemctl status relax.timer
• relax.timer - relax timer
  Loaded: loaded (/etc/systemd/system/relax.timer; disabled; preset: enabled)
  Active: active (waiting) since Sat 2024-10-12 21:26:48 MSK; 3min 17s ago
  Trigger: Sat 2024-10-12 21:31:08 MSK; 1min 2s left
  Triggers: • relax.service
$
```

Рисунок 43 – Пример работы демона

Контрольные вопросы

1. Что такое файловая система?

Файловая система — это способ организации, хранения и управления данными на диске или другом носителе информации. Она определяет структуру каталогов и файлов, доступ к ним, их имена и другие метаданные. Примеры файловых систем: ext4, NTFS, FAT32.

2. Права доступа к файлам.

В Linux права доступа делятся на три группы: для владельца файла, для группы и для всех остальных пользователей. Права могут быть на чтение (r), запись (w) и выполнение (x). Эти права можно управлять с помощью команд `chmod` и `chown`.

3. Что такое символическая ссылка?

Символическая ссылка (symlink) — это файл, который ссылается на другой файл или каталог, указывая на его путь. Она похожа на ярлык в Windows и указывает на оригинальный объект.

4. Что такое жесткая ссылка?

Жесткая ссылка — это альтернативное имя для существующего файла. В отличие от символической ссылки, жесткая ссылка указывает на тот же inode, что и исходный файл, и продолжает существовать даже после удаления исходного файла.

5. Команда поиска в Linux. Основные сведения.

Основная команда для поиска в Linux — это `find`. Она позволяет искать файлы и каталоги по различным критериям, таким как имя, размер, дата изменения и т. д. Пример: `find /path -name filename`.

6. Перечислите основные команды работы с каталогами.

`cd` — смена текущего каталога.

`ls` — отображение содержимого каталога.

`mkdir` — создание каталога.

`rmdir` — удаление пустого каталога.

`pwd` — вывод текущего пути.

`cp` — копирование файлов и каталогов.

`mv` — перемещение или переименование файлов и каталогов.

7. Чем отличается вывод команд `ls -F` и `ls -la`?

`ls -F` добавляет символы к именам файлов, указывающие на их тип (например, `/` для каталогов, `*` для исполняемых файлов).

`ls -la` выводит подробную информацию о файлах, включая права доступа, владельца, группу, размер и дату изменения, а также отображает скрытые файлы (начинающиеся с точки).

8. С помощью какой команды можно переместить файл в другой каталог?

Для перемещения файлов используется команда `mv`. Пример: `mv filename /new/directory/`.

9. Куда вы переходите, выполнив команду `cd` без параметров?

Команда `cd` без параметров возвращает пользователя в домашний каталог (`/home/username`).

10. Как осуществить просмотр каталогов и их содержимого?

Для просмотра содержимого каталогов используется команда `ls`. Для просмотра содержимого в скрытых и детальных режимах можно использовать `ls -la`.

11. Как осуществить создание нового каталога и необходимых каталогов рекурсивно?

Команда `mkdir -p /path/to/directory` создаст новый каталог, включая все недостающие родительские каталоги.

12. Как осуществить рекурсивное копирование всех файлов из одного каталога в другой?

Для этого используется команда `cp -r`. Пример: `cp -r /source/directory /destination/directory`.

13. Как рекурсивно удалить все файлы и подкаталоги в определенном каталоге?

Используйте команду `rm -r`. Пример: `rm -r /path/to/directory` удалит каталог и все его содержимое рекурсивно.

14. Перечислите основные ключи команды `ls` с их назначением.

-l — вывод в длинном формате с подробной информацией.

-a — отображение всех файлов, включая скрытые.

-r — вывод в обратном порядке.

-t — сортировка по времени изменения.

-F — добавление символов, указывающих на тип файла.

15. Команды `tee` и `cat`. Назначение и применение. Чем `cat` отличается от `more` и `less`?

`cat` — выводит содержимое файлов на экран или объединяет несколько файлов.

`tee` — считывает данные из стандартного ввода и одновременно записывает их в файл и выводит на экран.

`more` и `less` позволяют постранично просматривать содержимое файла, в отличие от `cat`, который выводит его сразу.

16. Что такое демон?

Демон – это процесс, который работает в фоновом режиме без прямого участия пользователя.

17. Для чего в операционной системе Linux применяется подсистема `systemD`?

`SystemD` — это система инициализации и управления службами в Linux. Она отвечает за запуск, остановку и управление службами, а также за отслеживание состояния системы.

18. Какие типы юнитов вы знаете?

В `SystemD` есть несколько типов юнитов:

`service` — для управления службами (демонами).

`socket` — для управления сокетами.

`target` — для группировки юнитов.

`device` — для управления устройствами.

`mount` — для управления файловыми системами.

19. С помощью каких команд осуществляется управление демонами?

Для управления демонами используются команды `systemctl`, такие как:

`systemctl start` — запуск службы.

`systemctl stop` — остановка службы.

`systemctl restart` — перезапуск службы.

`systemctl status` — проверка состояния службы.

20. Чем корутина отличается от потоков?

Корутины — это функции, которые могут быть приостановлены и возобновлены без создания нового потока. Они легче и эффективнее потоков, так как не требуют переключения контекста ядра.

21. Почему корутины должны быть реализованы как выделенная языковая возможность?

Реализация корутин как встроенной возможности языка позволяет более эффективно управлять потоками выполнения и их синхронизацией, упрощает их использование и уменьшает накладные расходы.

22. Какая польза от корутины?

Корутины позволяют писать асинхронные программы, которые легче поддерживать, чем традиционные многопоточные программы. Они помогают избежать блокировок и эффективно использовать процессорное время.

23. Перечислите команды, используемые вами при выполнении данной лабораторной работы, и кратко поясните назначение каждой из них.

`touch` — Используется для создания пустых файлов или для установки времени последнего изменения файла.

`ln` — Создание жестких или символических ссылок.

`cp` — Осуществляет копирование файлов.

`mv` — Осуществляет перемещение или переименования файлов.

`rm` — Используется для удаления файлов.

`mkdir` — Создание новой директории.

`echo` — Вывод строки текста на стандартное устройство вывода.

`chmod` — Изменение прав доступа к файлам и каталогам.

`tar` — Команда для работы с архивами (создание, распаковка и прочее).

ls — Вывод содержимого каталога и информации о файлах.

su — Смена пользователя.

cd — Изменение рабочего каталога.

sudo — Позволяет строго определенным пользователям выполнить указанные команды с административными привилегиями.

adduser — Создание новой учетной записи (автоматически настраивает нового пользователя с настройками конфигурации по умолчанию).

cat — Читает данные из файла или стандартного ввода и выводит их на экран.

Вывод

В ходе выполнения лабораторной работы был приобретён опыт работы с файлами и каталогами в ОС Linux, настроены права на доступ к файлам и каталогам.

Получены практические навыки по работе с подсистемой инициализации и управления службами.