

Министерство образования Республики Беларусь

Учреждение образования
«Белорусский государственный университет
информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра ПОИТ

Дисциплина «Архитектура компьютерной техники и
операционных систем»

ОТЧЁТ
к лабораторной работе №5

РАБОТА С ФАЙЛОВОЙ СИСТЕМОЙ ОС LINUX. ИЗУЧЕНИЕ
ОСНОВНЫХ КОМАНД И ОПЕРАТОРОВ УПРАВЛЕНИЯ
ИНТЕРПРЕТАТОРА BASH

Вариант 8

Студент группы №351001
Ушаков А.Д.
Преподаватель
Леванцевич В.А.

Минск 2024

СОДЕРЖАНИЕ

Введение.....	3
1 Работа с файловой системой ОС Linux.....	4
1.1 Загрузка пользователем root.....	4
1.2 Переход в конечной каталог и просмотр содержимого.....	4
1.3 Просмотр содержимого каталога файлов физических устройств.....	6
1.4 Просмотр и права доступа к файлу <code>vmlinux</code>	7
1.5 Возврат в директорию пользователя.....	7
1.6 Абсолютный путь до директории пользователя	8
1.7 Создание файлов <code>1.txt</code> , <code>2.txt</code> , <code>3.txt</code> разными способами	8
1.8 Изменение прав доступа к файлу <code>1.txt</code>	11
1.9 Жёсткая и символическая ссылки на <code>2.txt</code>	12
1.10 Создание каталога <code>new</code> в домашней директории.....	13
1.11 Копирование <code>1.txt</code> и перемещение <code>2.txt</code>	13
1.12 Изменение владельца файла или каталога	14
1.13 Поиск с помощью команды <code>find</code>	15
2 Управление интерпретатором BASH.....	16
2.1 Задание	16
2.2 Код скрипта.....	16
2.3 Запуск исполняемого файла.....	17
2.4 Результат работы скрипта	17
Заключение	18
Список использованных источников	19

ВВЕДЕНИЕ

Операционная система Linux создана на основе ОС UNIX и во многом имеет схожую структуру и систему команд. Пользователь может работать в текстовом режиме с помощью командной строки, или с использованием графического интерфейса X Window и одного из менеджеров рабочего стола (например, KDE или GNOME).

Файловая система – это структура, с помощью которой ядро операционной системы предоставляет пользователям (и процессам) ресурсы долговременной памяти системы, т. е. памяти на различного вида долговременных носителях информации – жестких дисках, SSD – дисках, магнитных лентах, CD-ROM и т. п.

С точки зрения пользователя, файловая система – это логическая структура каталогов и файлов. В отличие от Windows, где каждый логический диск хранит отдельное дерево каталогов, во всех UNIX-подобных системах эта древовидная структура растет из одного корня: она начинается с корневого каталога, родительского по отношению ко всем остальным, а физические файловые системы разного типа, находящиеся на разных разделах и даже на удаленных машинах, представляются как ветви этого дерева.

Linux и **Windows** используют различные файловые системы для хранения и организации доступа к информации на дисках. В Linux используются файловые системы – Ext2/Ext3, ReiserFS, FFS и другие. Все файловые системы имеют поддержку журналирования. Журналируемая файловая система сначала записывает изменения, которые она будет проводить в отдельную часть файловой системы (журнал) и только потом вносит необходимые изменения в остальную часть файловой системы. После удачного выполнения всех транзакций, записи удаляются из журнала. Это обеспечивает лучшее сохранение целостности системы и уменьшает вероятность потери данных. Следует отметить, что Linux поддерживает доступ к Windows-разделам.

Файловая система Linux имеет лишь один корневой каталог, который обозначается косой чертой (/). В файловой структуре Linux нет дисков A, B, C, D, ... , а есть только каталоги. В Linux различаются прописные и строчные буквы в командах, именах файлов и каталогов. В Windows у каждого файла существует лишь одно имя, в Linux их может быть много.

Bash – это shell-совместимый интерпретатор командного языка, выполняющий команды, прочитанные со стандартного входного потока или из файла. Скрипт-файл – это обычный текстовый файл, содержащий последовательность команд bash, для которого установлены права на выполнение.

1 РАБОТА С ФАЙЛОВОЙ СИСТЕМОЙ ОС LINUX

1.1 Загрузка пользователем root

В Ubuntu, как правило, по умолчанию не рекомендуется использовать root, поэтому доступ к нему часто возможен через команду `sudo` (см. рисунок 1.1.1).

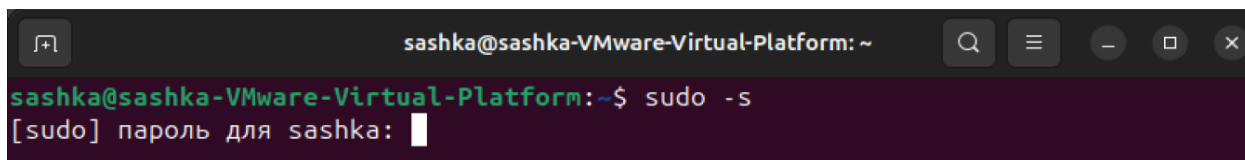


Рисунок 1.1.1 – Команда `sudo` для загрузки пользователем root

После этого терминал откроется под пользователем root, что будет обозначено значком `#` в конце строки ввода (см. рисунок 1.1.2).

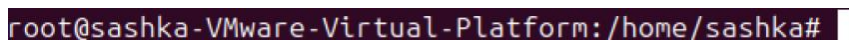
A screenshot of a terminal window showing the prompt 'root@sashka-VMware-Virtual-Platform:/home/sashka#' with a cursor.

Рисунок 1.1.2 – Изменения в терминале после ввода команды `sudo`

1.2 Переход в конечной каталог и просмотр содержимого

Перейти в корневой каталог можно с использованием команды `cd /` (см. рисунок 1.2.1).

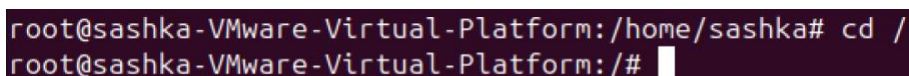
A screenshot of a terminal window showing the prompt 'root@sashka-VMware-Virtual-Platform:/home/sashka#' followed by the command 'cd /'. Below this, it shows the new prompt 'root@sashka-VMware-Virtual-Platform:/#' with a cursor.

Рисунок 1.2.1 – Команда перехода в корневой каталог и результат

Просмотреть содержимое каталога можно с помощью команды `ls -all`. Она отобразит все файлы и папки (включая скрытые) с дополнительными правами и атрибутами (см. рисунок 1.2.2).

Команда `ls -all` состоит из следующих:

- `ls` – команда, которая выводит список файлов и директорий;
- `-all` – флаг, который заставляет команду `ls` выводить все файлы и папки, включая скрытые (те, что начинаются с точки `.`), и показывает дополнительные данные о каждом объекте.

```

root@sashka-VMware-Virtual-Platform:/# ls -all
итого 3654756
drwxr-xr-x  23 root root    4096 окт 26 18:54 .
drwxr-xr-x  23 root root    4096 окт 26 18:54 ..
lrwxrwxrwx   1 root root      7 апр 22  2024 bin -> usr/bin
drwxr-xr-x   2 root root    4096 фев 26  2024 bin.usr-is-merged
drwxr-xr-x   3 root root    4096 окт 26 18:54 boot
dr-xr-xr-x   2 root root    4096 авг 27 19:22 cdrom
drwxr-xr-x  19 root root   4400 ноя  3 13:00 dev
drwxr-xr-x 140 root root  12288 окт 27 18:09 etc
drwxr-xr-x   3 root root    4096 окт 26 19:16 home
lrwxrwxrwx   1 root root      7 апр 22  2024 lib -> usr/lib
lrwxrwxrwx   1 root root      9 апр 22  2024 lib64 -> usr/lib64
drwxr-xr-x   2 root root    4096 апр  8  2024 lib.usr-is-merged
drwx-----  2 root root   16384 окт 26 18:43 lost+found
drwxr-xr-x   3 root root    4096 окт 26 19:16 media
drwxr-xr-x   2 root root    4096 авг 27 18:37 mnt
drwxr-xr-x   2 root root    4096 авг 27 18:37 opt
dr-xr-xr-x 386 root root      0 ноя  3 13:00 proc
drwx-----  7 root root    4096 окт 27 18:09 root
drwxr-xr-x  36 root root    880 ноя  3 13:00 run

```

Рисунок 1.2.2 – Результат работы команды `ls -all`

Каждая строка вывода содержит информацию о конкретном файле или каталоге. Рассмотрим, что означает каждая колонка. Права доступа (например, `drwxr-xr-x`).

Первый символ указывает тип файла:

- `d` – директория (каталог);
- `-` – обычный файл;
- `l` – символическая ссылка.

Следующие три группы символов указывают права доступа для: 1) владельца (первые три символа после `d`, например, `rwx`); 2) группы (вторая тройка, например, `r-x`); 3) остальных пользователей (последняя тройка, например, `r-x`).

Символы:

- `r` – право на чтение (read);
- `w` – право на запись (write);
- `x` – право на выполнение (execute).

Число ссылок – второе значение, например, 23 в первой строке. Это количество ссылок на данный файл или каталог.

Владелец – третья колонка, например, `root`. Указывает пользователя, которому принадлежит файл или каталог.

Группа – четвертая колонка, например, `root`. Указывает группу, которой принадлежит файл или каталог.

Размер – пятая колонка, например, 4096. Размер файла в байтах (если это каталог, то указывается размер его метаданных).

Дата и время последнего изменения – следующие колонки, например, окт 26 18:54. Показывают, когда файл или каталог был в последний раз изменен.

Имя файла или каталога – последняя колонка, например, ., .., bin, и так далее:

- “.”, “..” – специальные каталоги;
- “.” – текущий каталог;
- “..” – родительский каталог.

Синие записи (например, bin -> usr/bin) – это символические ссылки, которые указывают, что данная директория или файл являются ссылками на другие места в файловой системе.

1.3 Просмотр содержимого каталога файлов физических устройств

Перед тем, как просмотреть содержимое каталога с файлами физических устройств (команда `ls -all`), необходимо перейти в каталог `/dev` (включает файлы устройств, представляющие оборудование системы, например, `sda` для диска, `tty` для терминалов, и так далее) с помощью команды `cd /dev` – представлено на рисунках 1.3.1 и 1.3.2.

```
root@sashka-VMware-Virtual-Platform:/# cd /dev
root@sashka-VMware-Virtual-Platform:/dev#
```

Рисунок 1.3.1 – Переход в каталог `/dev`

```
root@sashka-VMware-Virtual-Platform:/dev# ls -all
итого 4
drwxr-xr-x 19 root root    4400 ноя  3 13:00 .
drwxr-xr-x 23 root root   4096 окт 26 18:54 ..
crw-r--r--  1 root root    10, 235 ноя  3 13:00 autofs
drwxr-xr-x  2 root root     600 ноя  3 13:01 block
drwxr-xr-x  2 root root     100 ноя  3 13:00 bsg
crw-----  1 root root   10, 234 ноя  3 13:00 btrfs-control
drwxr-xr-x  3 root root      60 ноя  3 13:00 bus
lrwxrwxrwx  1 root root       3 ноя  3 13:00 cdrom -> sr0
drwxr-xr-x  2 root root   3800 ноя  3 13:01 char
crw-----  1 root root      5,  1 ноя  3 13:00 console
lrwxrwxrwx  1 root root     11 ноя  3 13:00 core -> /proc/kcore
drwxr-xr-x  4 root root      80 ноя  3 13:00 cpu
crw-----  1 root root   10, 123 ноя  3 13:00 cpu_dma_latency
crw-----  1 root root   10, 203 ноя  3 13:00 cuse
drwxr-xr-x 10 root root    200 ноя  3 13:00 disk
drwxr-xr-x  2 root root      60 ноя  3 13:00 dma_heap
crw-rw----  1 root audio  14,  9 ноя  3 13:00 dmide
drwxr-xr-x  3 root root    100 ноя  3 13:00 dri
crw-----  1 root root   10, 125 ноя  3 13:00 ecryptfs
crw-rw----  1 root video  29,  0 ноя  3 13:00 fb0
lrwxrwxrwx  1 root root     13 ноя  3 13:00 fd -> /proc/self/fd
```

Рисунок 1.3.2 – Содержимое каталога файлов физических устройств

1.4 Просмотр и права доступа к файлу vmlinuz

Vmlinuz – это файл ядра Linux, который используется при загрузке системы. Обычно он находится в каталоге /boot. На рисунке 1.4.1 представлено, как можно просмотреть права доступа к этому файлу.

```
root@sashka-VMware-Virtual-Platform:/dev# cd /boot
root@sashka-VMware-Virtual-Platform:/boot# ls -l vmlinuz*
lrwxrwxrwx 1 root root      24 окт 26 18:53 vmlinuz -> vmlinuz-6.8.0-47-generic
-rw----- 1 root root 14956936 сен 27 21:47 vmlinuz-6.8.0-47-generic
lrwxrwxrwx 1 root root      24 окт 26 18:53 vmlinuz.old -> vmlinuz-6.8.0-47-gen
eric
```

Рисунок 1.4.1 – Просмотр прав доступа к файлу vmlinuz

На изображении показаны три строки ответа, потому что в каталоге /boot находятся три объекта, связанные с ядром Linux:

1 Vmlinuz – это символическая ссылка (показана буквой `l` в начале строки) на текущую версию файла ядра `vmlinuz-6.8.0-47-generic`. Символические ссылки позволяют создать ярлык для файла, чтобы к нему можно было обратиться по другому имени. Права доступа: `lrwxrwxrwx` означают, что это символическая ссылка, и все пользователи имеют право на её использование. Указывает на: `vmlinuz-6.8.0-47-generic`.

2 Vmlinuz-6.8.0-47-generic – это сам файл ядра (обычный файл), который содержит ядро операционной системы. Права доступа: `-rw-----` означают, что только пользователь `root` имеет право читать и записывать в этот файл, а остальные пользователи не имеют прав на него. Размер: 14956936 байтов. Дата и время последнего изменения: сен 27 21:47.

3 Vmlinuz.old – это символическая ссылка на старую версию ядра, которая также указывает на `vmlinuz-6.8.0-47-generic`. Обычно это используется для быстрого переключения на предыдущую версию ядра, если текущая вызывает проблемы. Права доступа: `lrwxrwxrwx` означают, что это символическая ссылка, и все пользователи имеют право на её использование. Указывает на: `vmlinuz-6.8.0-47-generic`.

1.5 Возврат в директорию пользователя

Вернуться в директорию пользователя возможно, прописав одну простую команду `exit` (см. рисунок 1.5.1).

```
root@sashka-VMware-Virtual-Platform:/boot# exit
exit
sashka@sashka-VMware-Virtual-Platform:~$
```

Рисунок 1.5.1 – Команда возврата в директорию пользователя

1.6 Абсолютный путь до директории пользователя

В Ubuntu, чтобы определить абсолютный путь до своего домашнего каталога, можно воспользоваться одним из трёх способов:

1 Использование переменной окружения. Переменные окружения хранят различные системные данные. HOME автоматически устанавливается при входе в систему и указывает на домашний каталог пользователя (см. рисунок 1.6.1).

2 Команда `pwd` (print working directory) выводит текущую рабочую директорию. Символ `~` является сокращением для домашнего каталога пользователя (см. рисунок 1.6.2).

3 Можно также воспользоваться командой `pwd` без сокращения, предварительно перейдя в домашний каталог с помощью команды `cd ~` (см. рисунок 1.6.3).

```
sashka@sashka-VMware-Virtual-Platform:~$ echo $HOME
/home/sashka
```

Рисунок 1.6.1 – Первый способ: использование переменной окружения

```
sashka@sashka-VMware-Virtual-Platform:~$ pwd ~
/home/sashka
```

Рисунок 1.6.2 – Второй способ: команда `pwd` с сокращением

```
sashka@sashka-VMware-Virtual-Platform:~$ cd ~
sashka@sashka-VMware-Virtual-Platform:~$ pwd
/home/sashka
```

Рисунок 1.6.3 – Третий способ: команда `pwd` без сокращения

1.7 Создание файлов 1.txt, 2.txt, 3.txt разными способами

Файл 1.txt будет создан с помощью команды `touch`, которая создает новые пустые файлы с указанными именами (см. рисунок 1.7.1).

```
sashka@sashka-VMware-Virtual-Platform:~$ touch ~/1.txt
```

Рисунок 1.7.1 – Создание файла с помощью команды `touch`

В результате в домашнем каталоге появляется файл 1.txt (см. рисунок 1.7.2).

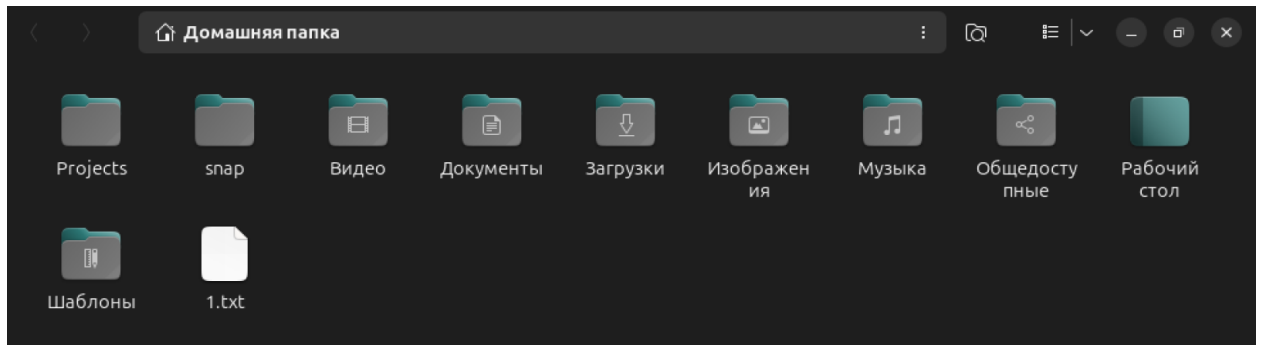


Рисунок 1.7.2 – Домашняя директория после создания 1.txt

Файл 2.txt будет создан с помощью команды `cat >`, при выполнении которой сначала необходимо ввести содержимое текстового документа (см. рисунки 1.7.3, 1.7.4, 1.7.5).

```
sashka@sashka-VMware-Virtual-Platform:~$ cat > ~/2.txt  
Hello, word
```

Рисунок 1.7.3 – Создание файла с помощью команды `cat`

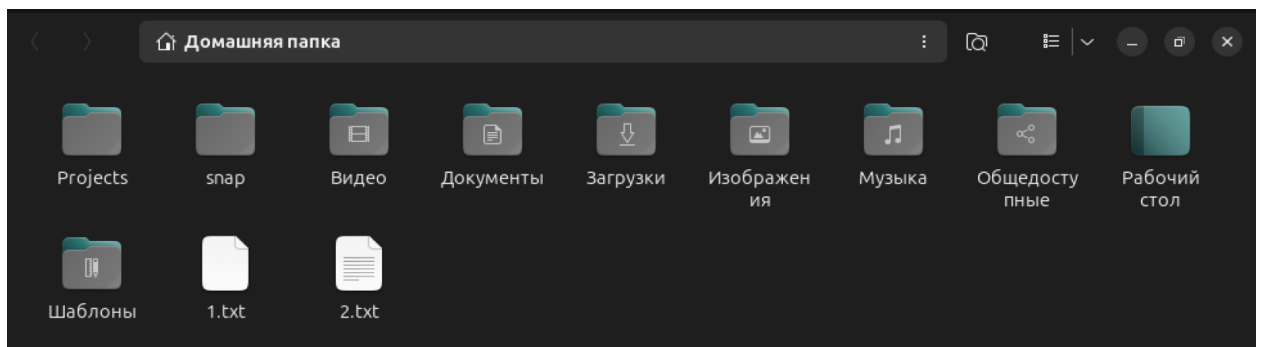


Рисунок 1.7.4 – Домашняя директория после создания 2.txt

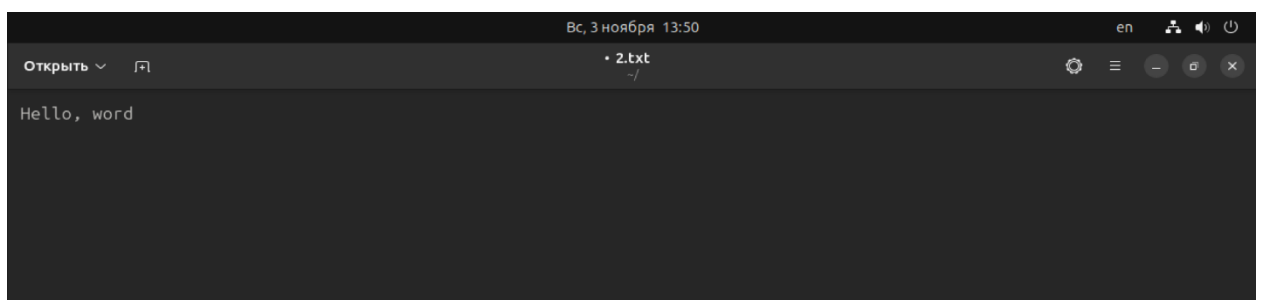


Рисунок 1.7.5 – Содержимое 2.txt после команды `cat`

Файл 3.txt будет создан с помощью текстового редактора nano, в котором изначально можно заполнить содержимое (см. рисунки 1.7.6, 1.7.7, 1.7.8, 1.7.9).

```
sashka@sashka-VMware-Virtual-Platform:~$ nano ~/3.txt
```

Рисунок 1.7.6 – Создание файла с помощью текстового редактора nano

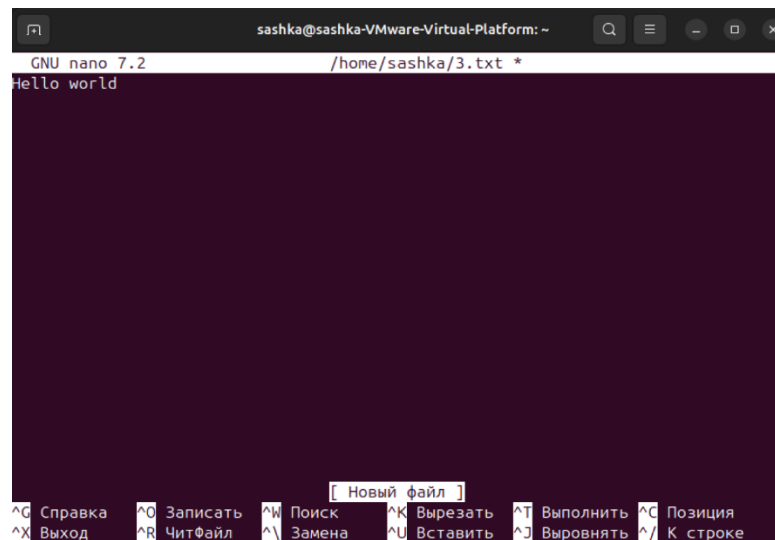


Рисунок 1.7.7 – Текстовый редактор nano

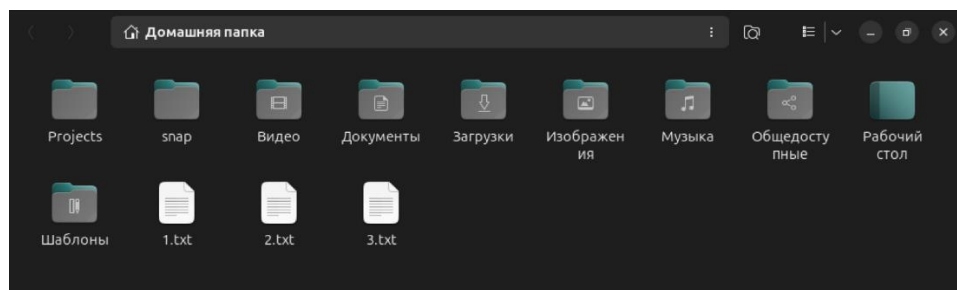


Рисунок 1.7.8 – Домашняя директория после создания 3.txt

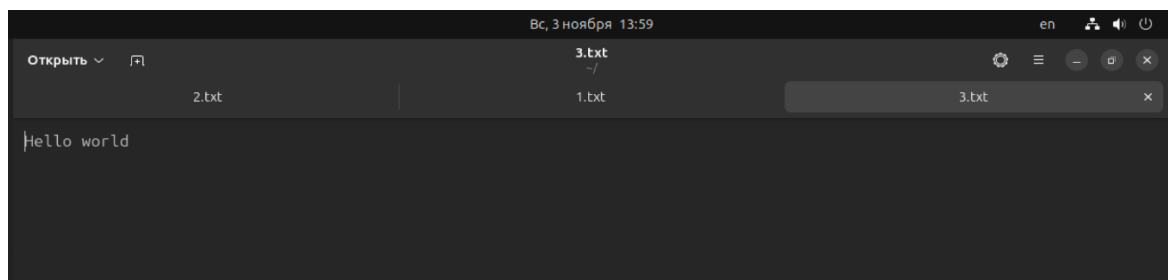


Рисунок 1.7.9 – Содержимое 3.txt

Чтобы просмотреть права доступа к файлам, можно воспользоваться уже известной командой `ls -l` (флаг `-l` выводит содержимое директории в "длинном формате", отображая более подробную информацию о каждом файле и каталоге) – смотреть рисунок 1.7.10.

```
sashka@sashka-VMware-Virtual-Platform:~$ ls -l ~/1.txt ~/2.txt ~/3.txt
-rw-rw-r-- 1 sashka sashka 11 ноя  3 13:55 /home/sashka/1.txt
-rw-rw-r-- 1 sashka sashka 12 ноя  3 13:47 /home/sashka/2.txt
-rw-rw-r-- 1 sashka sashka 12 ноя  3 13:58 /home/sashka/3.txt
```

Рисунок 1.7.10 – Просмотр прав доступа к файлам

1.8 Изменение прав доступа к файлу 1.txt

На рисунке 1.7.10 представлены исходные права доступа файла `1.txt` (смотреть предыдущий подраздел). Последствия изменений прав доступа с помощью команды `chmod` можно увидеть на рисунке 1.8.1. В данном случае была использована `chmod u+w ~/1.txt`, которая добавляет (+) разрешение на запись (`w`) для владельца файла (`u`).

```
sashka@sashka-VMware-Virtual-Platform:~$ chmod a-r ~/1.txt
sashka@sashka-VMware-Virtual-Platform:~$ ls -l ~/1.txt
--w--w---- 1 sashka sashka 11 ноя  3 13:55 /home/sashka/1.txt
```

Рисунок 1.8.1 – Изменение прав доступа к файлу с помощью команды `chmod`

В домашней директории это отображается так, как показано на рисунке 1.8.2.

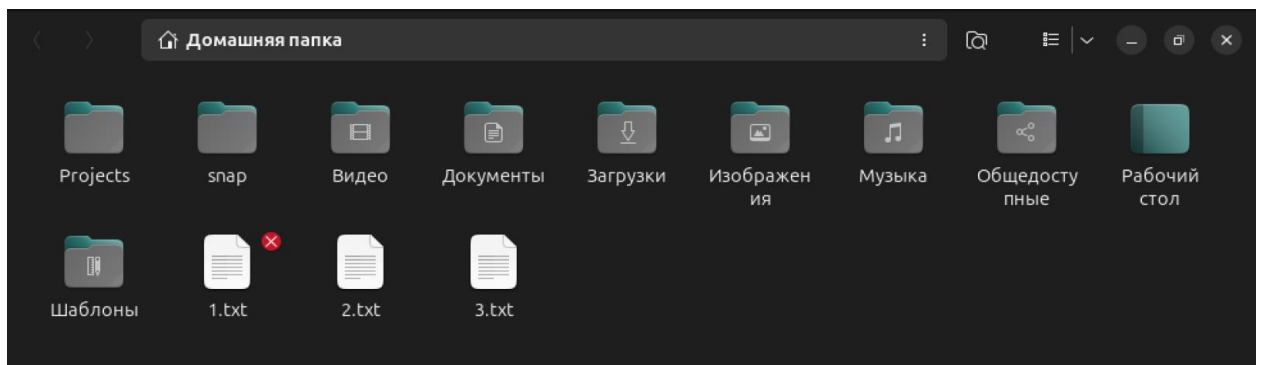


Рисунок 1.8.2 – Внешние изменения в домашней директории после `chmod`

1.9 Жёсткая и символическая ссылки на 2.txt

Жёсткая ссылка создается с помощью команды `ln`. В данном случае `ln ~/2.txt ~/hard_link_to_2.txt` (см. рисунок 1.9.1).

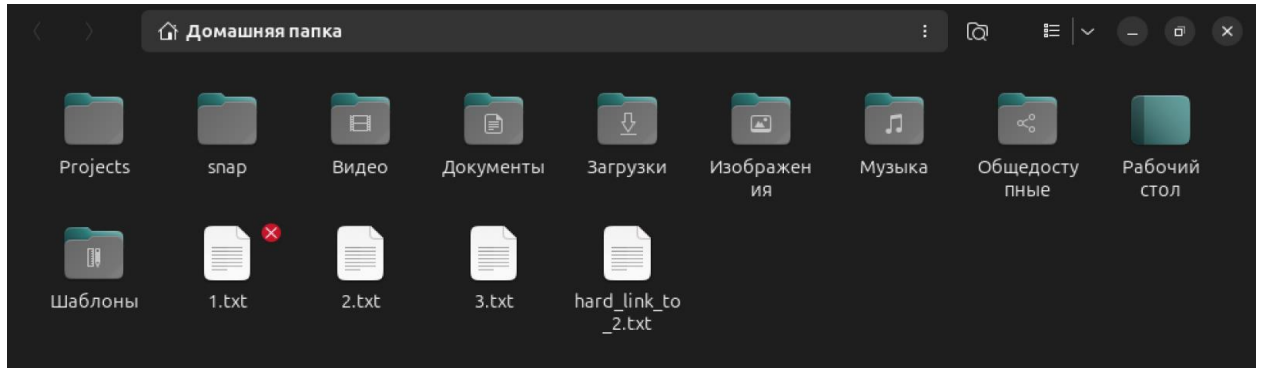


Рисунок 1.9.1 – Домашняя директория после создания жёсткой ссылки на 2.txt

Символическая ссылка создается с помощью команды `ln -s`. В данном случае `ln -s ~/2.txt ~/soft_link_to_2.txt` (см. рисунок 1.9.2). Содержимое созданного текстового файла можно увидеть на рисунке 1.9.3.

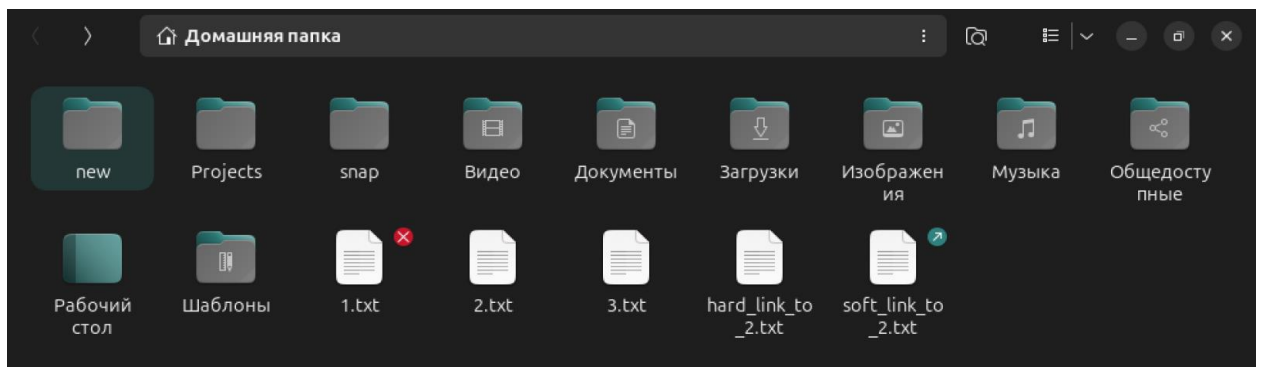


Рисунок 1.9.2 – Домашняя директория после создания символической ссылки на 2.txt

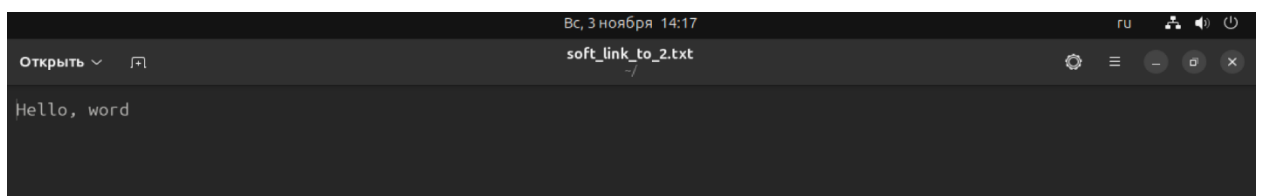


Рисунок 1.9.3 – Содержимое файла символической ссылки

Посмотреть результаты можно с помощью уже известной команды `ls -l ~/2.txt ~/hard_link_to_2.txt ~/soft_link_to_2.txt` (см. рисунок 1.9.4).

```
sashka@sashka-VMware-Virtual-Platform:~$ ls -l ~/2.txt ~/hard_link_to_2.txt ~/soft_link_to_2.txt
-rw-rw-r-- 2 sashka sashka 12 ноя  3 13:47 /home/sashka/2.txt
-rw-rw-r-- 2 sashka sashka 12 ноя  3 13:47 /home/sashka/hard_link_to_2.txt
lrwxrwxrwx 1 sashka sashka 18 ноя  3 14:15 /home/sashka/soft_link_to_2.txt -> /home/sashka/2.txt
```

Рисунок 1.9.4 – Результаты просмотра жёстких и символических ссылок

1.10 Создание каталога new в домашней директории

Создать новый каталог new (см. рис 1.10.1) можно с помощью команды `mkdir ~/new`. Вид домашней директории после создания представлен на рисунке 1.10.2.

```
drwxrwxr-x 2 sashka sashka 4096 ноя  3 14:25 new
```

Рисунок 1.10.1 – Отображения каталога new в сведениях о домашней директории

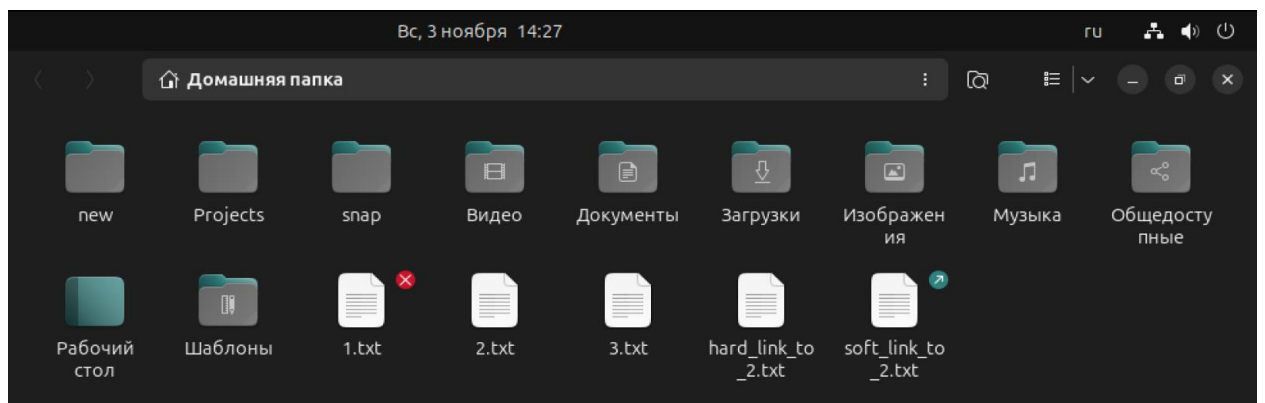


Рисунок 1.10.2 – Вид домашней директории после создания каталога new

1.11 Копирование 1.txt и перемещение 2.txt

Скопировать файл 1.txt в директорию new можно с помощью команды `cp ~/1.txt ~/new/`. Переместить файл 2.txt в директорию new можно с помощью команды `mv ~/2.txt ~/new/`. На рисунках 1.11.1, 1.11.2, 1.11.3 представлены результаты выполнения данных команд

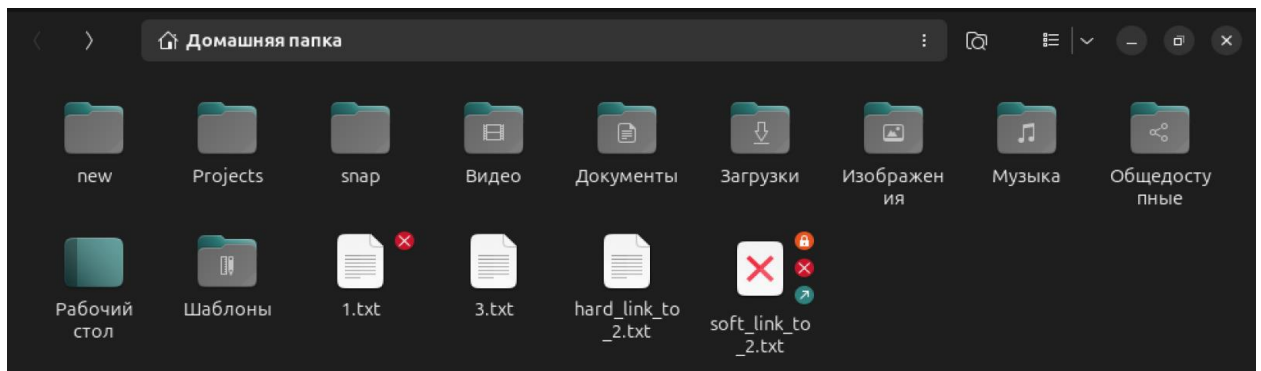


Рисунок 1.11.1 – Вид домашней директории после копирования и перемещения

```
sashka@sashka-VMware-Virtual-Platform:~$ ls ~/new
1.txt
```

Рисунок 1.11.2 – Сведения о каталоге new после копирования

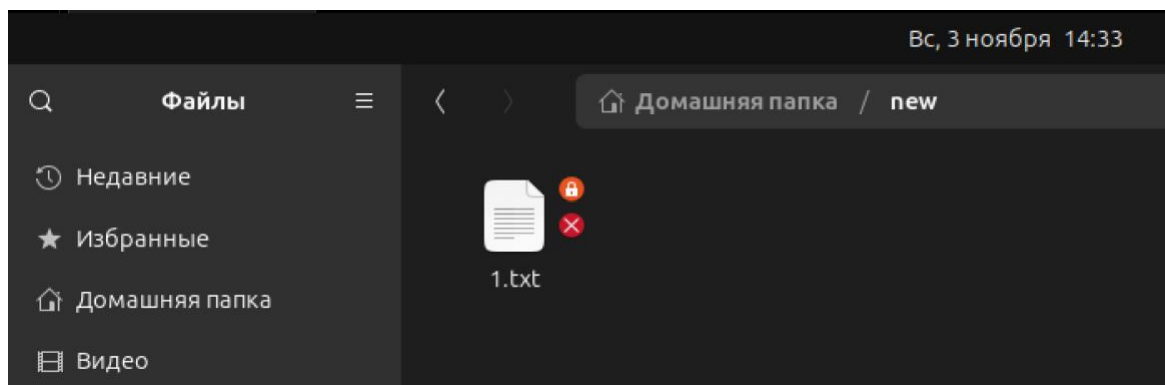


Рисунок 1.11.3 – Внешний вид каталога 1.11.3

1.12 Изменение владельца файла или каталога

С помощью команды `cat /etc/passwd` можно посмотреть список всех возможных пользователей, которые гипотетически могут стать владельцем. Командой `sudo chown new_owner ~/3.txt` меняем владельца (рис. 1.12.1).

```
colord:x:118:120:colord colour management daemon,,,:/var/lib/colord:/usr/sbin/nologin
gnome-initial-setup:x:119:65534::/run/gnome-initial-setup:/bin/false
gdm:x:120:121:Gnome Display Manager:/var/lib/gdm3:/bin/false
nm-openvpn:x:121:122:NetworkManager OpenVPN,,,:/var/lib/openvpn/chroot:/usr/sbin/nologin
sashka:x:1000:1000:sashka:/home/sashka:/bin/bash
sashka@sashka-VMware-Virtual-Platform:~$ ls -l ~/3.txt
-rw-rw-r-- 1 sashka sashka 12 ноя  3 13:58 /home/sashka/3.txt
sashka@sashka-VMware-Virtual-Platform:~$ sudo chown colord ~/3.txt
[sudo] пароль для sashka:
sashka@sashka-VMware-Virtual-Platform:~$ ls -l ~/3.txt
-rw-rw-r-- 1 colord sashka 12 ноя  3 13:58 /home/sashka/3.txt
```

Рисунок 1.12.1 – Изменение владельца файла

Может возникнуть вопрос «-rw-rw-r-- 1 colord sashka 12 ноя 3 13:58 /home/sashka/3.txt: почему в этой строке пишется 2 пользователя?» Первое (colord) – это непосредственно владелец файла, второе (sashka) – это группа, к которой принадлежит файл. Это означает, что все пользователи, входящие в группу sashka, имеют права, указанные для группы (в данном случае чтение и запись).

1.13 Поиск с помощью команды find

На рисунке 1.13.1 представлено использование команды find: файл был выбран случайно.



```
sashka@sashka-VMware-Virtual-Platform:~$ find ~ -name "1.txt"
/home/sashka/1.txt
/home/sashka/new/1.txt
```

Рисунок 1.13.1 – Результат поиска

В ходе поиска мы получили 2 пути одного и то же файла, что означает следующее: когда-то файл с таким же названием был перемещён в корзину и до сих пор остаётся там.

2 УПРАВЛЕНИЕ ИНТЕРПРЕТАТОРОМ BASH

2.1 Задание

Написать скрипт, находящий в заданном каталоге и всех его подкаталогах все файлы заданного расширения и создающий для каждого найденного файла жесткую ссылку в заданном каталоге. Расширение файла и каталог для жестких ссылок задаются в качестве первого и второго аргумента командной строки.

2.2 Код скрипта

```
#!/bin/bash

# Проверка на правильное количество аргументов
if [ "$#" -ne 2 ]; then
    echo "Использование: $0 <расширение> <каталог для ссылок>"
    exit 1
fi

# Получение аргументов
EXTENSION=$1 # Первый аргумент - расширение файла (например, txt)
LINK_DIR=$2  # Второй аргумент - каталог, в котором будут созданы жесткие ссылки

# Проверка, что каталог для ссылок существует
if [ ! -d "$LINK_DIR" ]; then
    echo "Каталог для ссылок $LINK_DIR не существует."
    exit 1
fi

# Поиск файлов с заданным расширением и создание жестких ссылок
find . -type f -name ".*$EXTENSION" | while read FILE; do
    # Получаем имя файла без пути
    BASENAME=$(basename "$FILE")
    # Создаем жесткую ссылку в целевом каталоге
    ln "$FILE" "$LINK_DIR/$BASENAME"
done

#chmod +x /home/sashka/Projects/ЛР5/create_hard_links.sh - сделать
#скрипт исполняемым
#/home/sashka/Projects/ЛР5/create_hard_links.sh txt
#/home/sashka/Projects/ЛР5/Links
# Вывод сообщения об успешном создании ссылок
echo "Жесткие ссылки созданы в каталоге $LINK_DIR."
```

2.3 Запуск исполняемого файла

Чтобы скрипт заработал, необходимо сделать его исполняемым с помощью команды `chmod +x /home/sashka/Projects/ЛР5/create_hard_links.sh`. После этого пользователю требуется запустить скрипт с помощью `./create_hard_links.sh txt /home/sashka/Projects/ЛР5/Links` в интересующей директории. В данном случае `./create_hard_links.sh` – сам исполняемый файл, `txt` – расширение, которое является объектом поиска, `/home/sashka/Projects/ЛР5/Links` – путь, куда будут создаваться жёсткие ссылки.

2.4 Результат работы скрипта

После указанных выше инструкций результат работы скрипта можно увидеть на рисунке 2.4.1.

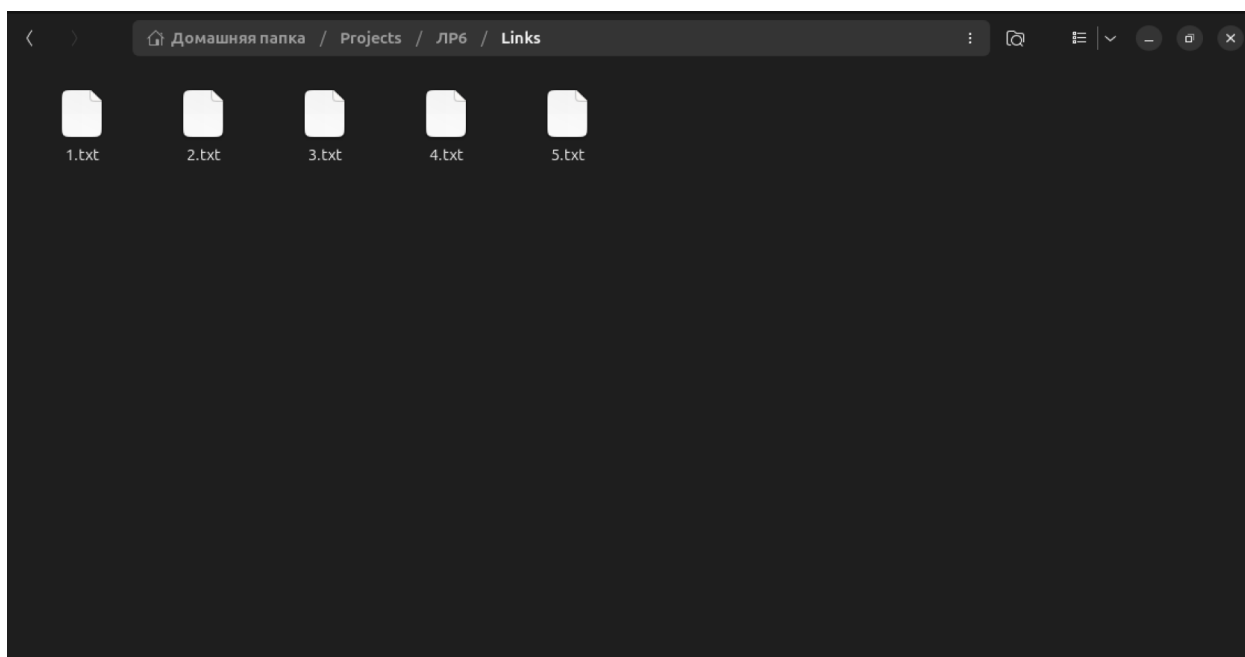


Рисунок 2.4.1 – Папка с созданными жёсткими ссылками

ЗАКЛЮЧЕНИЕ

В данной работе были изучены основы работы с файловой системой Linux и создание bash-скриптов для автоматизации задач. Файловая система Linux организована в виде иерархического дерева, начиная с корневого каталога, что позволяет эффективно управлять данными и находить нужные файлы. Операционная система предоставляет разнообразные команды для создания, удаления, перемещения и копирования файлов, а также изменения прав доступа к ним, что обеспечивает безопасность и удобство работы с данными.

Использование bash-скриптов является ключевым инструментом для автоматизации в Linux. Bash позволяет выполнять команды в виде сценариев, создавая скрипты, которые автоматизируют повторяющиеся задачи и минимизируют риск ошибок. В ходе работы были освоены основные конструкции языка bash, такие как условия, циклы, встроенные команды `find`, `ln`, `echo`, а также управление переменными, что позволяет создавать эффективные и многофункциональные скрипты для ускорения выполнения рутинных операций.

Отдельное внимание в работе уделено использованию ссылок. Linux поддерживает два типа ссылок: жесткие и символические. Жесткие ссылки позволяют создать дополнительный доступ к одному и тому же файлу без дублирования его содержимого, что экономит место и позволяет организовать доступ к данным из разных мест. Символические ссылки, аналогичные ярлыкам, предоставляют гибкость в управлении файловой структурой, позволяя удобно перемещаться по системе.

Таким образом, навыки работы с файловой системой и bash-скриптами позволяют эффективно использовать Linux для управления ресурсами, автоматизации процессов и упрощения повседневных задач. Знание командной строки, структуры файловой системы и принципов создания скриптов значительно повышает эффективность работы и открывает возможности для глубокого взаимодействия с операционной системой.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

[1] Леванцевич, В. А. Работа с файловой системой ОС Linux / В. А. Леванцевич // Лабораторные работы. – 2024. – №5.1.

[2] Леванцевич, В. А. Изучение основных команд и операторов управления интерпретатора BASH и созданию на их основе скрипт-файлов/ В. А. Леванцевич // Лабораторные работы. – 2024. – №5.2.