



СОКОЛОВ ОЛЕГ ВАДИМОВИЧ

ул. Авиационная, д. 1, корп. Б, кв. 11 г. Сочи, Россия, 354340

email: mail@osokolov.ru, ☎ 8-962-407-111-7

ПРАКТИЧЕСКАЯ РАБОТА № 2

курса «Linux для робототехников».

Основание:

изучение модуля «Основы Linux»
курса «Linux для робототехников».

Исполнитель:

Соколов Олег Вадимович



СОКОЛОВ ОЛЕГ ВАДИМОВИЧ

ул. Авиационная, д. 1, корп. Б, кв. 11 г. Сочи, Россия, 354340

email: mail@osokolov.ru, ☎ 8-962-407-111-7

ПРАКТИЧЕСКАЯ РАБОТА № 2

курса «Linux для робототехников».

СОДЕРЖАНИЕ:

[Задание 1. Запуск ROS-узлов при старте системы.](#)

[1.1. Выполненная работа.](#)

[1.2. Результат.](#)

[Задание 2. Автоматическое удаление файлов.](#)

[2.1. Объект.](#)

[2.2. Выполненная работа.](#)

[2.3. Результат.](#)

[2.4. Примечание.](#)

[Задание 3. Подсчет общей продолжительности видеофайлов.](#)

[3.1. Объект](#)

[3.2. Выполненная работа.](#)

[3.3. Результат.](#)

[Приложение 1. Иллюстративный материал.](#)

Задание 1. Запуск ROS-узлов при старте системы.

1.1. Выполненная работа¹ [[ссылка на содержание](#)]:

- создан .launch-файл² в каталоге `~/Documents/Module2/PW2/Scripts/`;
- создан скрипт запуска³, которому были даны права на исполнение;
- в каталоге `/etc/systemd/system/` был создан systemd unit-файл `ros-demo.service`;
- активирован созданный сервис командой `sudo systemctl enable ros-demo.service`;
- проведена перезагрузка системы;
- проведена проверка статуса службы с помощью команды `systemctl status ros-demo.service` [[см. иллюстрацию 1](#)];
- с помощью команды `rosnode list` был получен список запущенных узлов ROS [[см. иллюстрацию 2](#)];
- проведен контроль публикуемых сообщений с помощью команды `rostopic echo /demo_topic` [[см. иллюстрацию 2](#)].

1.2. Результат [[ссылка на содержание](#)].

Был создан скрипт, позволяющий с использованием systemd автоматически запускать ROS-узлы на роботе при старте системы.

Задание 2. Автоматическое удаление файлов.

2.1. Объект [[ссылка на содержание](#)].

Для имитации накопителя информации робота был использован USB-накопитель, смонтированный в каталог `/media/sapsan/Data4Gb`.

2.2. Выполненная работа [[ссылка на содержание](#)]:

2.2.1. Был создан скрипт для создания файлов `file_creator.sh`⁴;

Здесь и далее в каждом скрипте будут использованы опции для Bash, обеспечивающие строгий режим выполнения скрипта `set -euo pipefail`, где:

- e - выход при ошибке любой команды;
- u - выход при использовании неинициализированной переменной;
- x - вывод команд перед их выполнением для отладки;
- o pipefail - выход при ошибке в любой части конвейера.

В скрипте используются следующие основные операторы и функции:

- `create_files()` - функция для создания файлов в заданном каталоге. Она принимает размер файла и путь к каталогу в качестве параметров. Затем в цикле создает файлы с именами `file1.dat`, `file2.dat` и т.д. с помощью команды `dd`. Если

¹ В соответствии с «Дополнительными материалами к Заданию 1 ПР М2»

² https://drive.google.com/open?id=108nVdIuP_T97Gj4Y06_s_w3MK1zOGB5X&usp=drive_fs

³ https://drive.google.com/open?id=108oIuV_-K2fi4pz5wM70fJ2gzHe3JBYP&usp=drive_fs

⁴ https://drive.google.com/open?id=103AyRejwAC0pZHpTBVAmWLphNpUbiA-0&usp=drive_fs

при создании файла возникает ошибка, она записывается в файл ``creation_log.txt``, иначе выводится сообщение об успешном создании файла. В функции используются следующие операторы:

- ``local target_dir=$2`` - оператор объявляет локальную переменную ``target_dir``, которая принимает значение второго переданного аргумента функции, то есть путь к целевому каталогу;
- ``for ((i=1; i<=count; i++))`` - оператор создает цикл, который будет выполняться от 1 до значения переменной ``count``, которое задается пользователем;
- ``local file_path="${target_dir}/file${i}.dat"`` - оператор формирует путь к файлу, который будет создан в цикле, используя значение ``target_dir`` и номер файла ``i``;
- ``if ! dd if=/dev/zero of="$file_path" bs=1M count=$1 > /dev/null 2>&1; then`` - оператор использует команду ``dd`` для создания файла заданного размера. В случае ошибки при создании файла, выводится сообщение об ошибке и ошибка логируется в файл ``creation_log.txt``.
- ``echo "Создан файл ${file_path} размером $1 мегабайт" | tee -a creation_log.txt`` - оператор выводит сообщение о успешном создании файла и его размере, а также логирует эту информацию в файл ``creation_log.txt``.

- ``prompt_user()`` - функция для запроса данных у пользователя и вызова функции ``create_files()``. Она запрашивает размер файла, количество файлов и путь к целевому каталогу. Проверяет, что введенные значения являются положительными числами и что каталог существует. Если все проверки пройдены успешно, функция записывает начало процесса создания файлов в ``creation_log.txt``, вызывает ``create_files()`` и записывает успешное завершение. В функции используются следующие операторы:

- ``read -p "Введите размер файла в мегабайтах: " size`` - оператор запрашивает у пользователя ввод размера файла в мегабайтах;
- ``read -p "Введите количество файлов: " count`` - оператор запрашивает у пользователя ввод количества файлов;
- ``read -p "Введите название целевого каталога: " target_dir`` - оператор запрашивает у пользователя ввод пути к целевому каталогу;
- ``if ! [["$size" =~ ^[1]+$]] || ! [["$count" =~ ^[1]+$]]; then`` - оператор проверяет, что введенные значения для размера файла и количества файлов являются положительными числами;
- ``if [! -d "$target_dir"]; then`` - оператор проверяет существование указанного целевого каталога;

- ``echo "Начало создания файлов $(date)" | tee creation_log.txt`` - оператор логирует начало процесса создания файлов в файл ``creation_log.txt``;
 - ``create_files $size $target_dir`` - оператор вызывает функцию ``create_files`` с передачей размера файла и пути к каталогу в качестве аргументов;
 - ``echo "Файлы успешно созданы $(date)" | tee -a creation_log.txt`` - оператор логирует успешное завершение процесса создания файлов в файл ``creation_log.txt``.
- ``prompt_user`` - вызов функции ``prompt_user()`` для запуска скрипта.

2.2.2. Был создан скрипт для удаления файлов ``file_eraser_find_awk.sh``⁵, в котором все входные параметры (период времени – 7 дней, порог свободного места – менее 20%, каталог - ``/media/sapsan/Data4Gb``, путь и наименование файла логирования) заданы через переменные в начале скрипта, а также использовались рекомендуемые команды ``find`` и ``awk``.

В скрипте используются две основные функции:

- ``delete_old_files()`` - функция отвечает за удаление файлов, которые старше заданного периода. Внутри функции используется команда ``find`` для поиска файлов в указанной директории (`$FOLDER`), которые были изменены более `$PERIOD` дней назад, и затем эти файлы удаляются с помощью команды ``rm``.

- ``check_disk_space()`` - функция проверяет доступное место на диске. Она сначала получает информацию о диске с помощью команды ``df``, а затем использует ``awk`` для извлечения необходимых данных, таких как общий объем, использованное пространство, доступное пространство и процент заполнения. Если процент заполнения меньше порогового значения (`$THRESHOLD`), то вызывается функция ``delete_old_files`` для удаления старых файлов, и в лог-файл записывается информация о действии.

В скрипте используются следующие операторы:

- операторы присваивания:

- ``PERIOD=7`` - присваивает переменной значение 7;
- ``FOLDER="/media/sapsan/Data4Gb"`` - присваивает переменной путь к директории;
- ``THRESHOLD=20`` - присваивает переменной значение 20;
- ``LOG_FILE="/media/sapsan/Data4Gb/deletion_log.txt"`` - присваивает переменной путь к файлу лога;

- оператор вызова функции:

- ``delete_old_files`` - вызывает функцию `delete_old_files`;
- ``check_disk_space`` - вызывает функцию `check_disk_space`;

⁵ https://drive.google.com/open?id=1-nEs77-KsJfTQG3slQjuUDDZDsP6uX2K&usp=drive_fs

⁶ Написаны заглавными буквами, так как являются неизменяемыми параметрами (константами)

- оператор конвейера:

- - `df -h \$FOLDER | awk 'NR==2{print \$2}` - передает вывод df в awk для обработки;
- - `df -h \$FOLDER | awk 'NR==2{print \$3}` - передает вывод df в awk для обработки;
- - `df -h \$FOLDER | awk 'NR==2{print \$4}` - передает вывод df в awk для обработки;
- - `df -h \$FOLDER | awk 'NR==2{print \$5}' | tr -d '%'` - передает вывод df в `awk`, затем в `tr` для обработки;

- условный оператор `if`:

- - `if [\$percentage -lt \$threshold]; then ... fi` - проверяет, меньше ли процент заполнения диска порогового значения;

- оператор вывода в файл:

- - `echo "Действие: Файлы старше \$period дней удалены." >> \$log_file` - дописывает строку в файл лога;
- - `echo "Информация: Общий объём: \$total, Использовано: \$used, Свободно: \$available, Процент заполнения: \$percentage%." >> \$log_file` - дописывает строку в файл лога;

2.2.3. Проведены следующие действия по подтверждению работоспособности кода:

- в каталоге `/etc/systemd/system/` был создан systemd unit-файл `robot-file-eraser.service`;
- активирован созданный сервис командой `sudo systemctl enable robot-file-eraser.service`;
- деактивирована в настройках операционной системы автоматическая установку даты и времени (выключено подключение к сети Интернет);
- с помощью команды `sudo date -s "2024-05-10"` было изменено системное время операционной системы на период 10 суток назад;
- с помощью скрипта `file_creator.sh` было создано 9 файлов размером 400 Мб [см. иллюстрацию 3];
- проведена проверка лог-файла⁷ и созданных файлов [см. иллюстрацию 4];
- далее было установлено актуальное системное время операционной системы (включен Интернет);
- проведена перезагрузка системы;
- проведена проверка статуса службы с помощью команды `systemctl status robot-file-eraser.service` [см. иллюстрацию 5];

⁷ https://drive.google.com/open?id=1-rBXbMBIWt8g8JcYAEi4ZkHolVfjRwbV&usp=drive_fs

- проведена проверка удаленных файлов, лог-файла⁸ и свободного места после удаления файлов [[см. иллюстрацию 6](#)].

2.3. Результат [[ссылка на содержание](#)].

Создан скрипт, создающий файлы заданного размера с помощью команды ``dd``, имитируя запись данных роботом. Создан скрипт по удалению файлов. Создан и активирован автоматический сервис удаления файлов по заданным параметрам.

2.4. Примечание [[ссылка на содержание](#)].

2.4.1. Столь подробное описание скриптов создано в учебных целях лично для меня, чтоб всегда можно было освежить накопленные знания. Ну, или является следствием моей профессиональной деформации ☺ (я по профессии судебный эксперт).

2.4.2. Видел чате обсуждение о возможности использования в процессе обучения виртуальных машин. В процессе выполнения данного задания столкнулся с проблемами отладки и неработоспособности кода именно из-за виртуальной машины и особенностей обращения виртуальной ОС к внешним устройствам через мосты. На «железе» скрипт отработывал без ошибок, на виртуалке возникали ошибки, причем так до конца и не разобрался почему. Таким образом, я всё-таки согласен с тем, что для процесса обучения лучше использовать «железо».

2.4.3. В процессе изучения материала об удалении файлов подсмотрел интересное решение и в качестве бонуса сделал для себя удобный «скрипт-удалятор» с использованием ``fzf``. Если будет интересно, то ссылка ниже⁹.

Задание 3. Подсчет общей продолжительности видеофайлов.

3.1. Объект [[ссылка на содержание](#)].

Объектом для выполнения задания № 3 были выбраны файлы видеокурса, зарегистрированные в каталоге ``~/Videos/``, древовидная структура которых приведена далее [[см. иллюстрацию 7](#)].

3.2. Выполненная работа [[ссылка на содержание](#)].

3.2.1. Был создан скрипт для подсчета продолжительности каждого видеофайла и продолжительности видео во всем каталоге ``duration_for_file.sh``¹⁰.

3.2.1.1. Описание скрипта:

- переменные:

- ``total_duration`` - переменная для хранения общей продолжительности всех файлов. Инициализируется нулевым значением;
- ``LOG_FILE`` - переменная, содержащая путь к лог-файлу;

⁸ https://drive.google.com/open?id=1012wINxDKG2UI9KIWFG4zeleovbk_eEr&usp=drive_fs

⁹ https://drive.google.com/open?id=102Dnk_u0zsgWzMpnW6VwdjmBmO8285sE&usp=drive_fs

¹⁰ https://drive.google.com/open?id=1-pCnRV1Fy3o3QaxAZ7fg_t_n1RahZ-ig&usp=drive_fs

- команды:

- ``echo "" > "$LOG_FILE"`` - очистка или создание лог-файла. Если файл не существует, он будет создан пустым;
- ``for file in /home/sapsan/Videos/*`` - цикл ``for``, перебирающий все файлы в директории ``/home/sapsan/Videos/``. Переменная ``file`` будет принимать значение каждого файла по очереди;
- ``duration=$(ffprobe -v error -show_entries format=duration -of default=nokey=1:noprint_wrappers=1 "$file")`` - получение продолжительности текущего файла с помощью команды ``ffprobe``. Результат сохраняется в переменную ``duration``;
- ``duration=$(echo "$duration" | awk '{print int($1)}')`` - форматирование продолжительности с помощью ``awk``. Округление до целого числа секунд и сохранение в переменную ``duration``;
- ``echo "$file: $duration seconds" >> "$LOG_FILE"`` - запись информации о файле и его продолжительности в лог-файл;
- ``total_duration=$((total_duration + duration))`` - суммирование продолжительности текущего файла с общей продолжительностью;
- ``echo "Total duration: $total_duration seconds"`` - вывод общей продолжительности после обработки всех файлов;

- операторы:

- ``$()`` - подстановка команды. Результат выполнения команды внутри скобок подставляется в команду;
- ``>>`` - дозапись в файл. Используется для записи результатов в лог-файл;
- ``$((expression))`` - арифметическая подстановка. Используется для суммирования продолжительностей;

- работа скрипта:

Инициализируется переменная ``total_duration`` для хранения общей продолжительности. Определяется путь к лог-файлу ``video_durations.log``. Очищается или создается лог-файл с помощью ``echo "" > "$LOG_FILE"``. Начинается цикл ``for``, перебирающий все файлы в директории ``/home/sapsan/Videos/``. Далее для каждого файла: получается продолжительность файла с помощью ``ffprobe``, продолжительность форматируется в читаемый вид с помощью ``awk``, продолжительность записывается в лог-файл с помощью ``echo``, продолжительность текущего файла суммируется с ``total_duration``, после обработки всех файлов, выводится общая продолжительность с помощью ``echo``.

3.2.1.2. Проверка работоспособности скрипта:

Скрипту были даны права на выполнение с последующим его запуском. Результат выполнения скрипта проиллюстрирован в Приложении № 1 [[см. иллюстрацию 8](#)] и записан в лог-файл `video_durations.log`¹¹.

3.2.2. Далее в качестве бонуса ☺ был создан скрипт для подсчета продолжительности видеофайлов в подкаталогах и продолжительности видео во всем каталоге `duration_for_dir.sh`¹².

3.2.2.1. Описание скрипта:

- переменные:

- `SEARCH_DIR` - переменная, содержащая путь к директории, в которой необходимо найти видеофайлы;
- `total_duration` - переменная для хранения общей продолжительности всех видеофайлов. Инициализируется нулевым значением;
- `LOG_FILE` - переменная, содержащая путь к лог-файлу;

- команды:

- `echo "" > "\$LOG_FILE"` - очистка или создание лог-файла;
- `find "\$SEARCH_DIR" -type d | while read dir; do` - поиск папок в указанной директории;
- `duration=\$(find "\$dir" -maxdepth 1 -type f -name "*.mp4" -exec ffprobe -v error -show_entries format=duration -of default=noprint_wrappers=1:nokey=1 {} \; | awk '{sum += \$1} END {printf "%02d:%02d:%02d\n", int(sum/3600), int((sum%3600)/60), int(sum%60)}')` - подсчёт продолжительности видеофайлов в текущей папке;
- `convert_to_seconds() { ... }` - функция для преобразования формата "ЧЧ:ММ:СС" в секунды;
- `convert_to_hms() { ... }` - функция для преобразования секунд в формат "ЧЧ:ММ:СС";
- `echo "..." >> "\$LOG_FILE"` - запись результатов в лог-файл;
- `echo "..." | tee -a "\$LOG_FILE"` - запись результатов в лог-файл и вывод на экран;
- `done` - окончание цикла поиска папок;

- операторы:

- `IFS=: read -r hours minutes seconds <<< "\$1"` - разделение строки на часы, минуты и секунды;

¹¹ https://drive.google.com/open?id=10AhXdRrAv3Cb7BASDJC_vj_MoiGL93K&usp=drive_fs

¹² https://drive.google.com/open?id=1-oF06wrCQk8aF3kE-50NEmljZGRsF2hx&usp=drive_fs

- ``echo $((10#{hours} * 3600 + 10#{minutes} * 60 + 10#{seconds}))`` - преобразование секунд в формат "ЧЧ:ММ:СС";
- ``printf "%02d:%02d:%02d" $hours $minutes $seconds`` - форматирование продолжительности в формат "ЧЧ:ММ:СС";
- ``total_duration=$((total_duration + duration_in_seconds))`` - суммирование продолжительности;
- ``echo "..." | tee -a "$LOG_FILE"`` - запись результатов в лог-файл и вывод на экран.

- работа скрипта:

Инициализируется переменная ``total_duration`` для хранения общей продолжительности. Определяется путь к лог-файлу ``video_dir_durations.log``. Очищается или создается лог-файл с помощью ``echo "" > "$LOG_FILE"``. Начинается цикл поиска подкаталогов в каталоге ``/home/sapsan/Videos``. Далее для каждого подкаталога: подсчитывается продолжительность видеофайлов с помощью ``find`` и ``ffprobe``; продолжительность форматируется в читаемый вид с помощью ``awk``; продолжительность записывается в лог-файл; продолжительность суммируется с ``total_duration``; после обработки всех папок, выводится общая продолжительность всех видеофайлов. Конвертируется общая продолжительность в формат "ЧЧ:ММ:СС" с помощью функции ``convert_to_hms``. Выводится и записывается общая продолжительность в лог-файл.

3.2.2.2. Проверка работоспособности скрипта:

Скрипту были даны права на выполнение с последующим его запуском. Результат выполнения скрипта проиллюстрирован в Приложении № 1 [\[см. иллюстрацию 9\]](#) и записан в лог-файл ``video_dir_durations.log``¹³.

3.3. Результат [\[ссылка на содержание\]](#).

Создан скрипт для автоматического подсчёта общей продолжительности всех видеофайлов (формата .mp4) в каталоге и его подкаталогах. Продолжительность выведена в формате ``чч:мм:сс``.

Выполнены рекомендации по использованию инструментов ``find``, ``ffprobe``, ``awk`` для достижения цели задания.

По желанию: реализован вывод продолжительности видео отдельно по папкам.

Приложение: иллюстративный материал на 5 л., в 1 экз.

О.В. Соколов

29.05.2024

¹³ https://drive.google.com/open?id=10TaGx-KuxXy-rV-Ud074Sc1_CtZiWyLn&usp=drive_fs

Приложение 1 к практической работе № 2 курса «Linux для робототехников».

ИЛЛЮСТРАТИВНЫЙ МАТЕРИАЛ

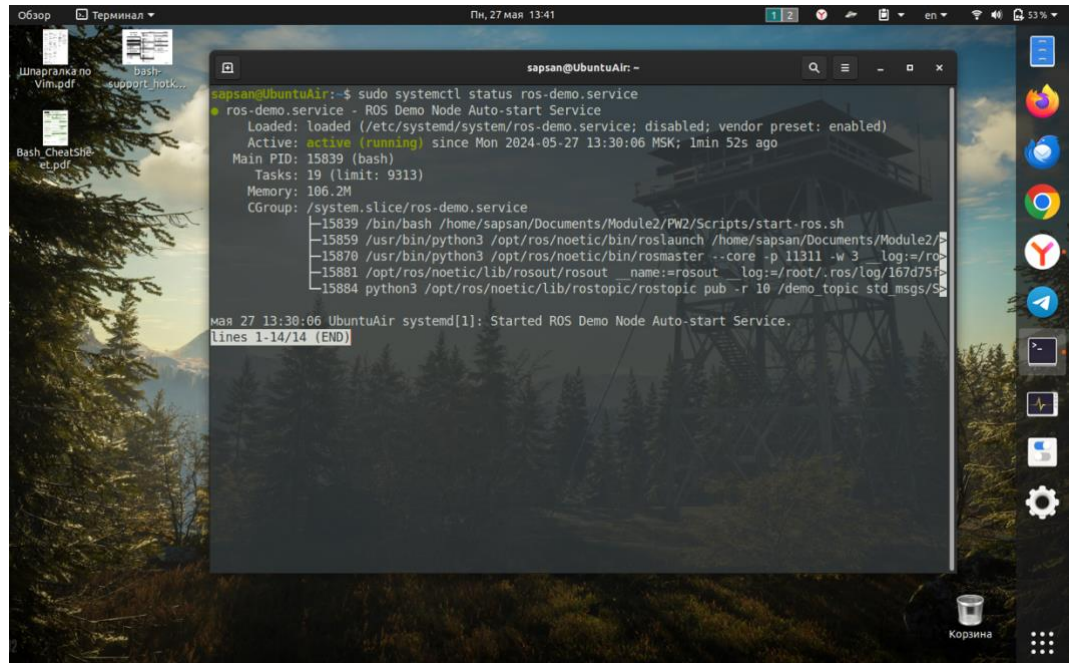


Иллюстрация 1. Проверка статуса службы с помощью команды `systemctl status ros-demo.service` [\[обратная ссылка\]](#).

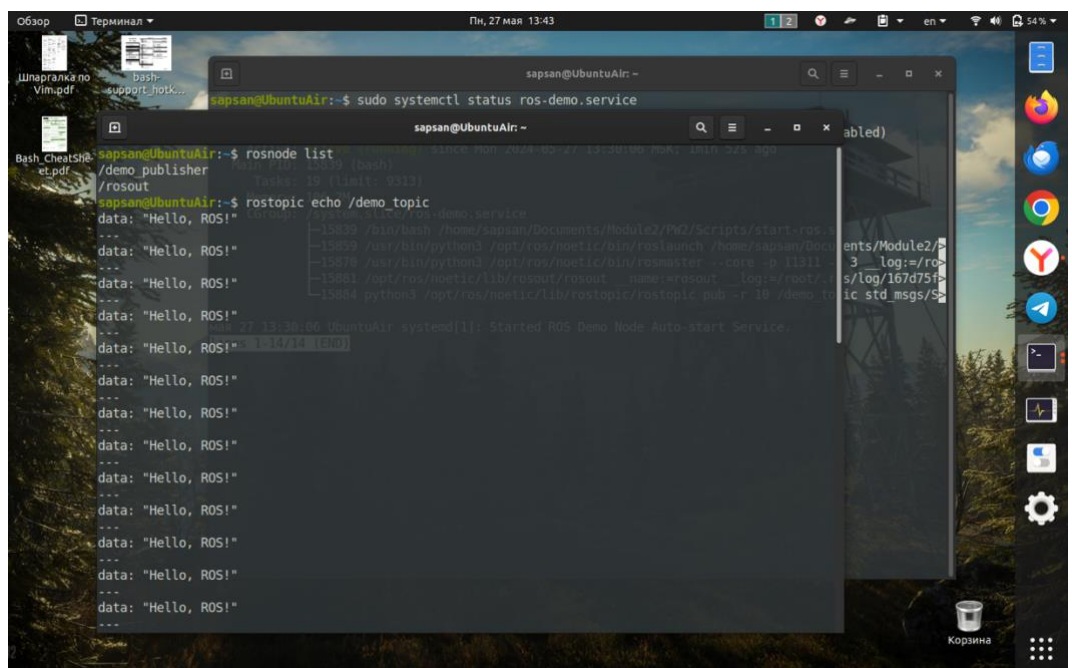


Иллюстрация 2. Список запущенных узлов ROS и контроль публикуемых сообщений [\[обратная ссылка\]](#).

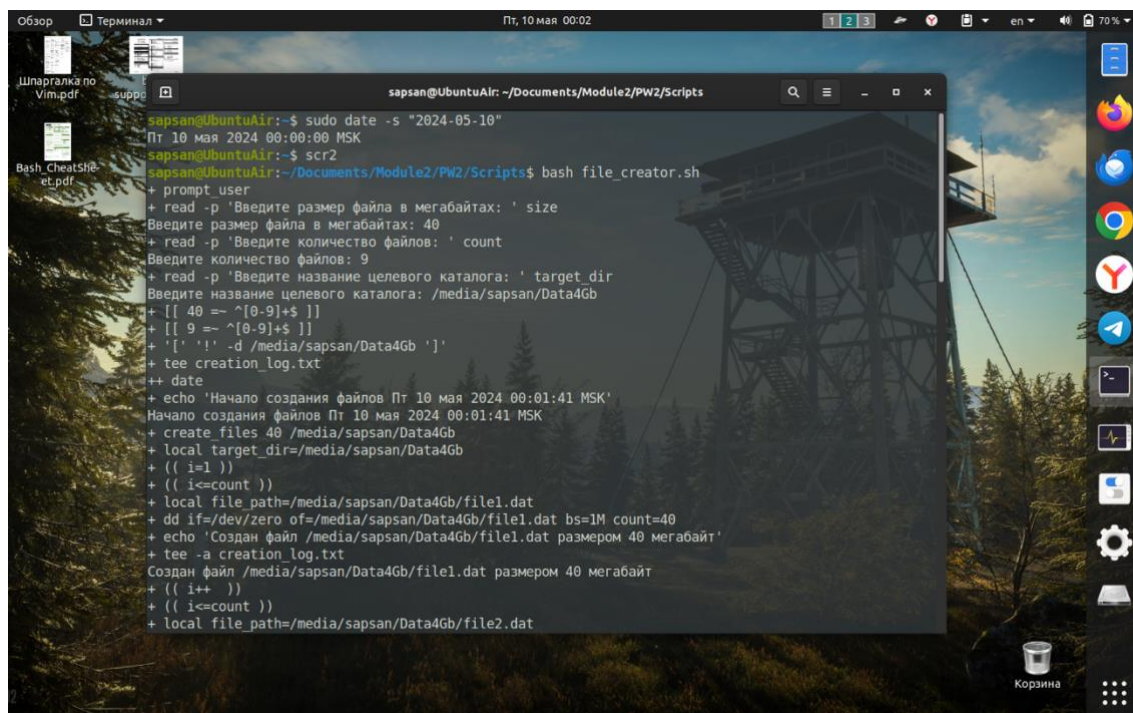


Иллюстрация 3. Изменение системного времени и создание файлов [[обратная ссылка](#)].

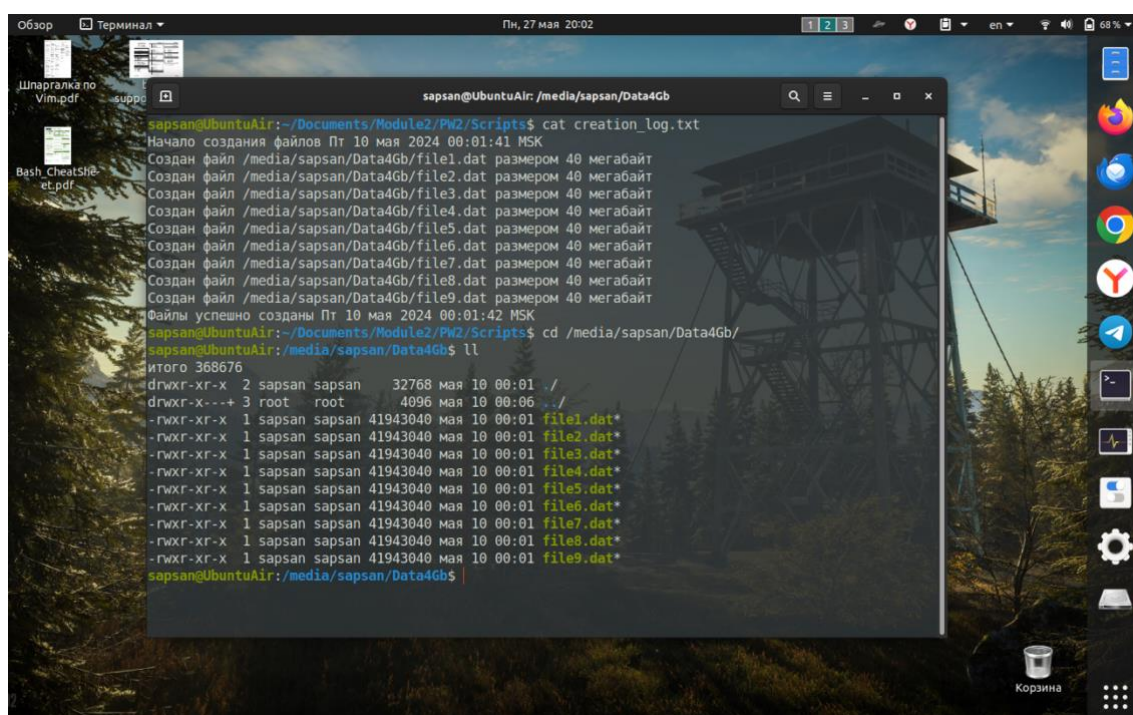


Иллюстрация 4. Проверка файла лога и созданных файлов [[обратная ссылка](#)].

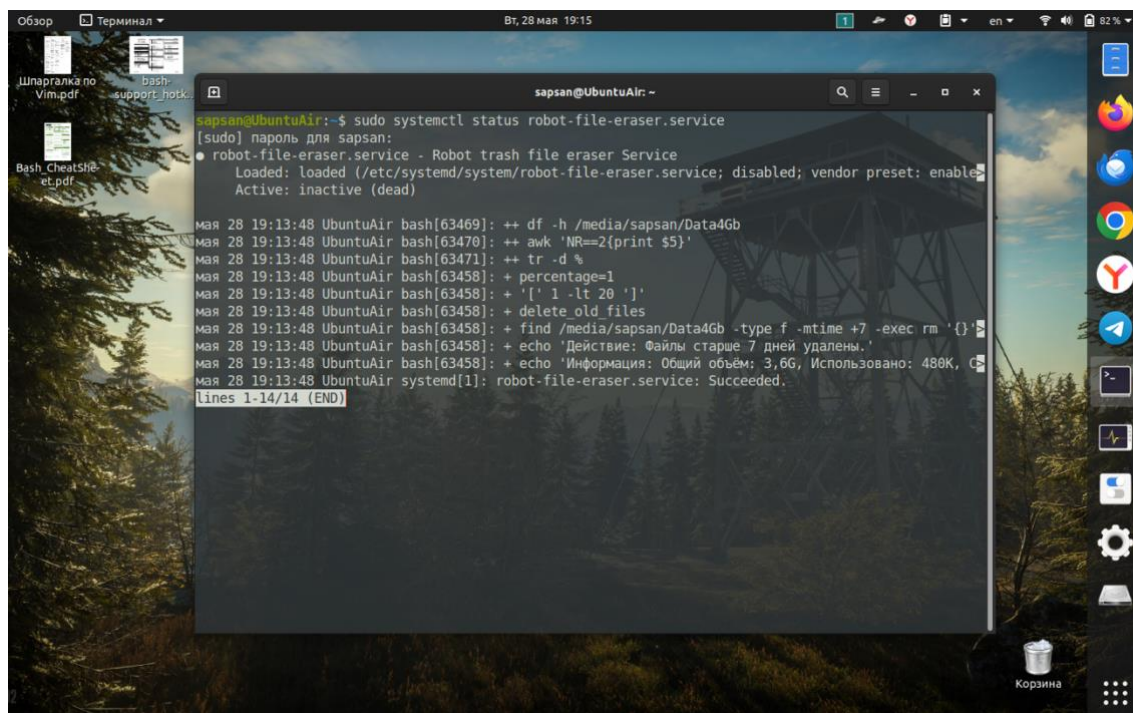


Иллюстрация 5. Проверка статуса службы с помощью команды `systemctl status robot-file-eraser.service` [\[обратная ссылка\]](#).

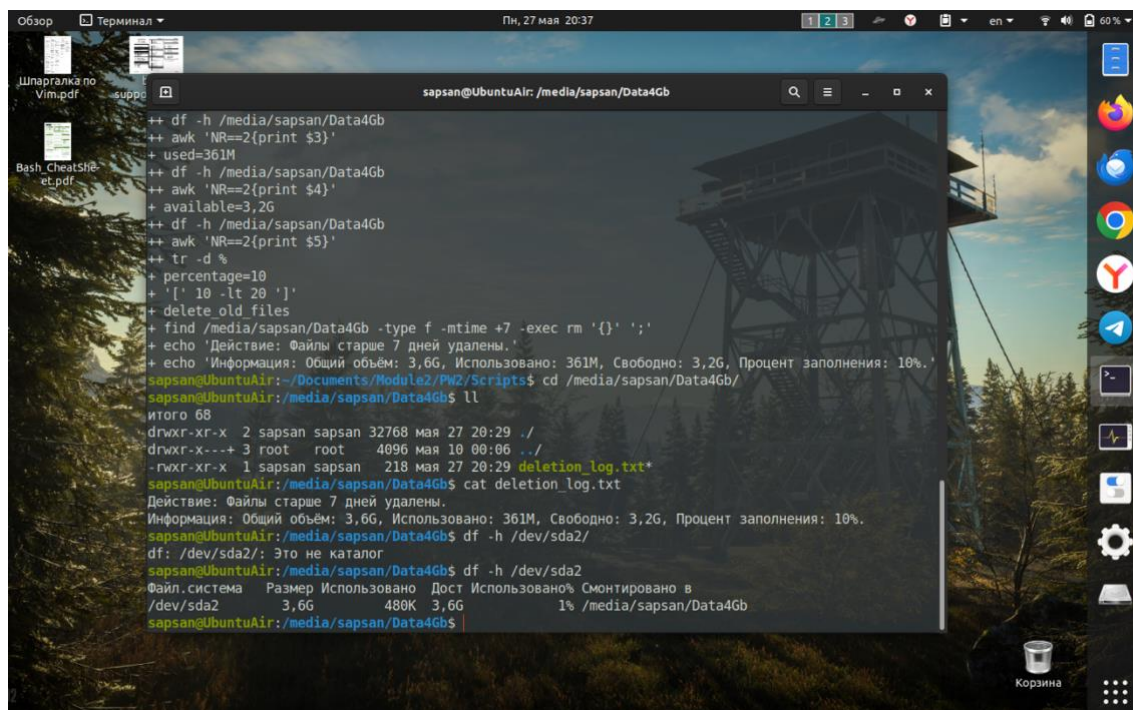


Иллюстрация 6. Проверка удаленных файлов, лог-файла и свободного места после удаления файлов [\[обратная ссылка\]](#).

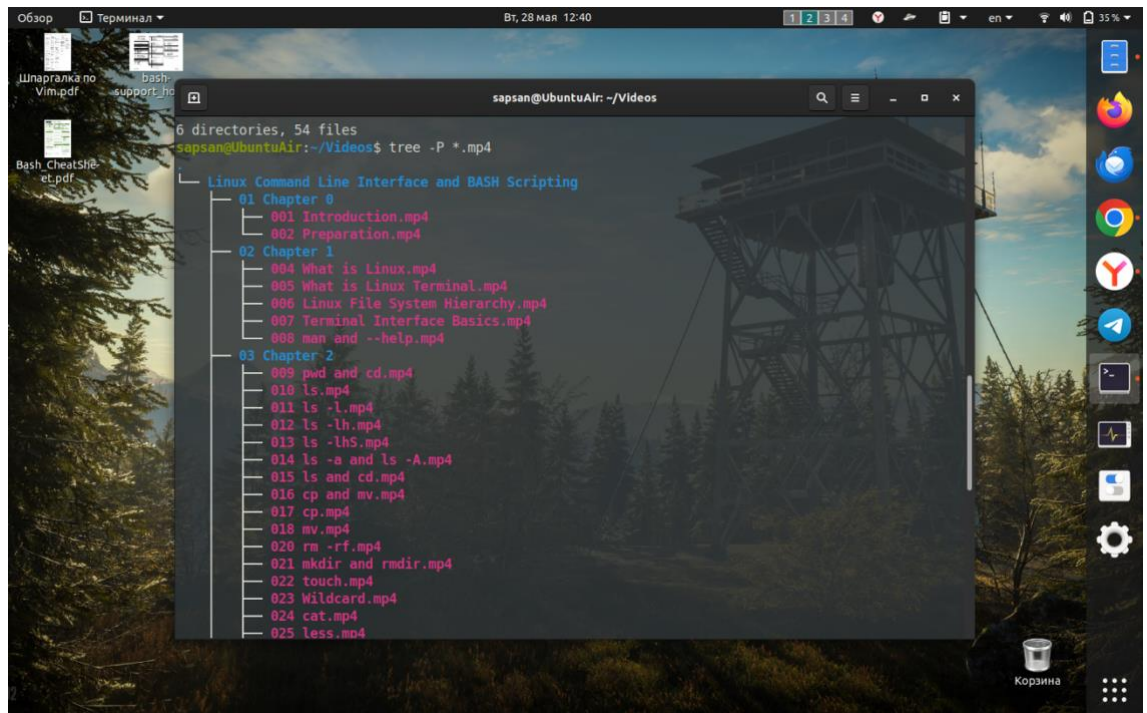


Иллюстрация 7. Фрагмент древовидной структуры каталога `~/Videos`
[\[обратная ссылка\]](#).

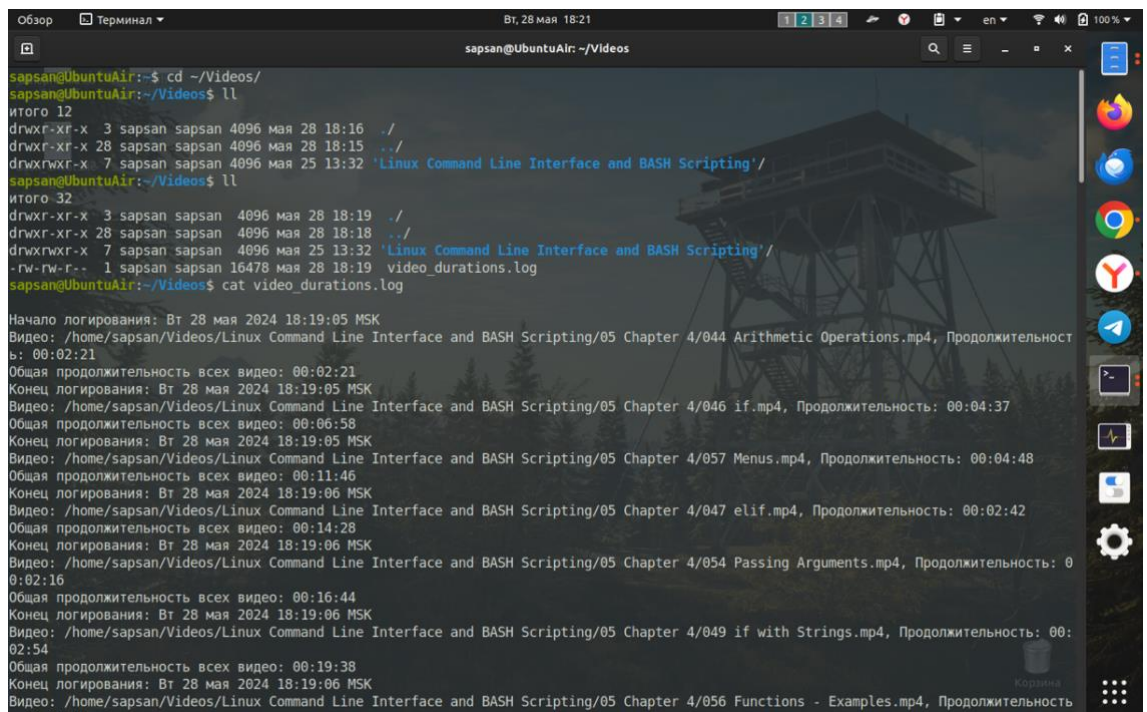


Иллюстрация 8. Проверка работоспособности скрипта `duration_for_file.sh`
[\[обратная ссылка\]](#).

```
Обзор Терминал Бт, 28 мая 18:23 1 2 3 4 en 100%
sapsan@UbuntuAir: ~/Videos

drwxr-xr-x 3 sapsan sapsan 4096 мая 28 18:22 ./
drwxr-xr-x 28 sapsan sapsan 4096 мая 28 18:18 ../
drwxr-xr-x 7 sapsan sapsan 4096 мая 25 13:32 'Linux Command Line Interface and BASH Scripting'/
-rw-rw-r-- 1 sapsan sapsan 2919 мая 28 18:22 video_dir_durations.log
-rw-rw-r-- 1 sapsan sapsan 16478 мая 28 18:19 video_durations.log
sapsan@UbuntuAir:~/Videos$ cat video_dir_durations.log
cat: video_dir_durations.log: No such file or directory
sapsan@UbuntuAir:~/Videos$

Начало логирования: Бт 28 мая 2024 18:22:20 MSK
Обрабатывается папка: /home/sapsan/Videos
Общая продолжительность видео в папке /home/sapsan/Videos: 00:00:00
Общая продолжительность всех видео: 00:00:00
Конец логирования: Бт 28 мая 2024 18:22:20 MSK
Обрабатывается папка: /home/sapsan/Videos/Linux Command Line Interface and BASH Scripting
Общая продолжительность видео в папке /home/sapsan/Videos/Linux Command Line Interface and BASH Scripting: 00:00:00
Общая продолжительность всех видео: 00:00:00
Конец логирования: Бт 28 мая 2024 18:22:20 MSK
Обрабатывается папка: /home/sapsan/Videos/Linux Command Line Interface and BASH Scripting/05 Chapter 4
Общая продолжительность видео в папке /home/sapsan/Videos/Linux Command Line Interface and BASH Scripting/05 Chapter 4: 00:53:16
Общая продолжительность всех видео: 00:53:16
Конец логирования: Бт 28 мая 2024 18:22:21 MSK
Обрабатывается папка: /home/sapsan/Videos/Linux Command Line Interface and BASH Scripting/03 Chapter 2
Общая продолжительность видео в папке /home/sapsan/Videos/Linux Command Line Interface and BASH Scripting/03 Chapter 2: 01:32:53
Общая продолжительность всех видео: 02:26:09
Конец логирования: Бт 28 мая 2024 18:22:23 MSK
Обрабатывается папка: /home/sapsan/Videos/Linux Command Line Interface and BASH Scripting/02 Chapter 1
Общая продолжительность видео в папке /home/sapsan/Videos/Linux Command Line Interface and BASH Scripting/02 Chapter 1: 00:08:00
Общая продолжительность всех видео: 02:34:09
Конец логирования: Бт 28 мая 2024 18:22:23 MSK
Обрабатывается папка: /home/sapsan/Videos/Linux Command Line Interface and BASH Scripting/01 Chapter 0
Общая продолжительность видео в папке /home/sapsan/Videos/Linux Command Line Interface and BASH Scripting/01 Chapter 0: 00:04:08
Общая продолжительность всех видео: 02:38:17
Конец логирования: Бт 28 мая 2024 18:22:23 MSK
Обрабатывается папка: /home/sapsan/Videos/Linux Command Line Interface and BASH Scripting/04 Chapter 3
Общая продолжительность видео в папке /home/sapsan/Videos/Linux Command Line Interface and BASH Scripting/04 Chapter 3: 00:13:27
Общая продолжительность всех видео: 02:51:44
Конец логирования: Бт 28 мая 2024 18:22:23 MSK
sapsan@UbuntuAir:~/Videos$
```

Иллюстрация 9. Проверка работоспособности скрипта `duration_for_dir.sh`
[обратная ссылка].