Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение

высшего образования

«МОСКОВСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

(МОСКОВСКИЙ ПОЛИТЕХ)

Лабораторная работа 2

По теме: «Программирование на Bash в среде Linux»

По дисциплине:

«Администрирование серверов»

Выполнил

Студент 3 курса

Кирксова Кристина С.

Гр 231-321

Проверил

Гневшев А.Ю.

Москва 2025

Задание 1. Шахматная доска в консоли

Требования:

1. Пользователь вводит размер доски (например, 8 → 8x8).
2. Доска отображается в консоли с чередованием цветов (ANSI-коды).
3. Поддерживаются чётные и нечётные размеры.

Описание реализации:

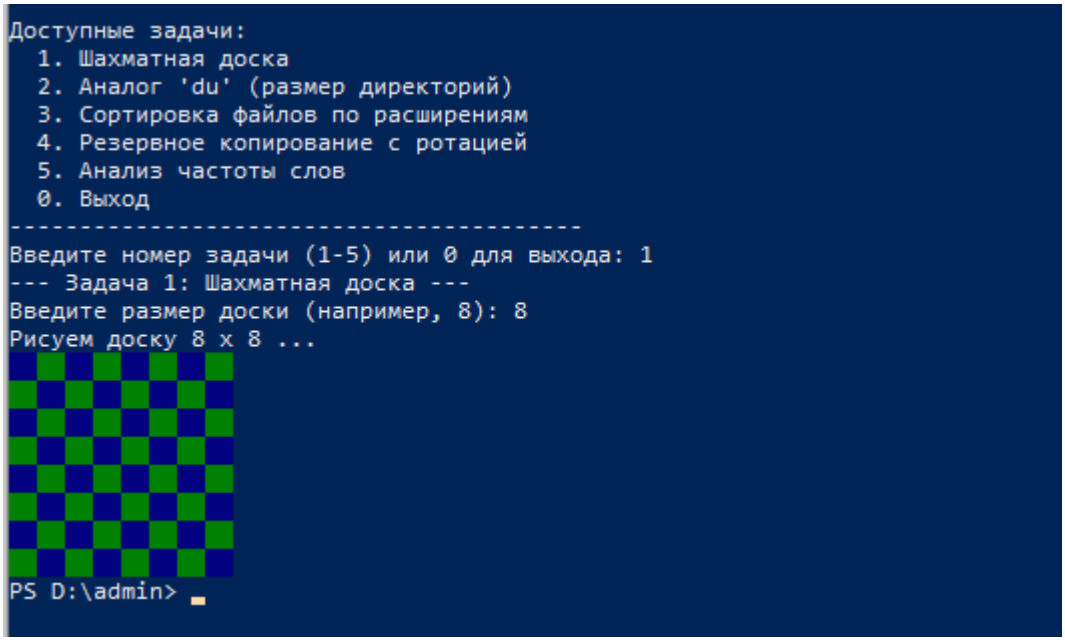
- Используется двойной цикл для строк и столбцов.
- Цвета реализованы с помощью ANSI-кодов.
- После каждой строки выполняется сброс цвета.
- Добавлена проверка корректности введённого размера.

Фрагмент кода:

```
#
============================================================
==================
# ЗАДАНИЕ 1: ШАХМАТНАЯ ДОСКА (Функция)
#
============================================================
==================
draw_chessboard() {
  local size="$1"
  if ! [[ "$size" =~ ^[1-9][0-9]*$ ]]; then
    error_exit "Внутренняя ошибка: размер доски должен быть числом."
  fi

  echo "Рисуем доску $size x $size ..."
  local color1='\e[44m'
  local color2='\e[42m'
  local reset_color='\e[0m'

  for (( i=0; i<size; i++ )); do
    for (( j=0; j<size; j++ )); do
      local total=$((i + j))
      if [ $((total % 2)) -eq 0 ]; then
        echo -en "${color1}  ${reset_color}"
      else
        echo -en "${color2}  ${reset_color}"
      fi
    done
    echo
```

```
  done
}
```

Скриншот 1: запуск программы и выбор пункта 1 в меню/вывод шахматной доски



Задание 2. Аналог du (подсчёт дискового пространства)

Требования:

- Скрипт принимает путь к директории.
- Рекурсивно подсчитывает объём файлов и поддиректорий.
- Выводит результат в человеко-читаемом формате (G, M, K).
- Использует find и stat.

Описание реализации:

- Суммирование размеров файлов происходит через stat -c %s.
- Вывод форматируется функцией format_size().
- Добавлена проверка существования директории.
- Обрабатываются ошибки доступа к файлам.

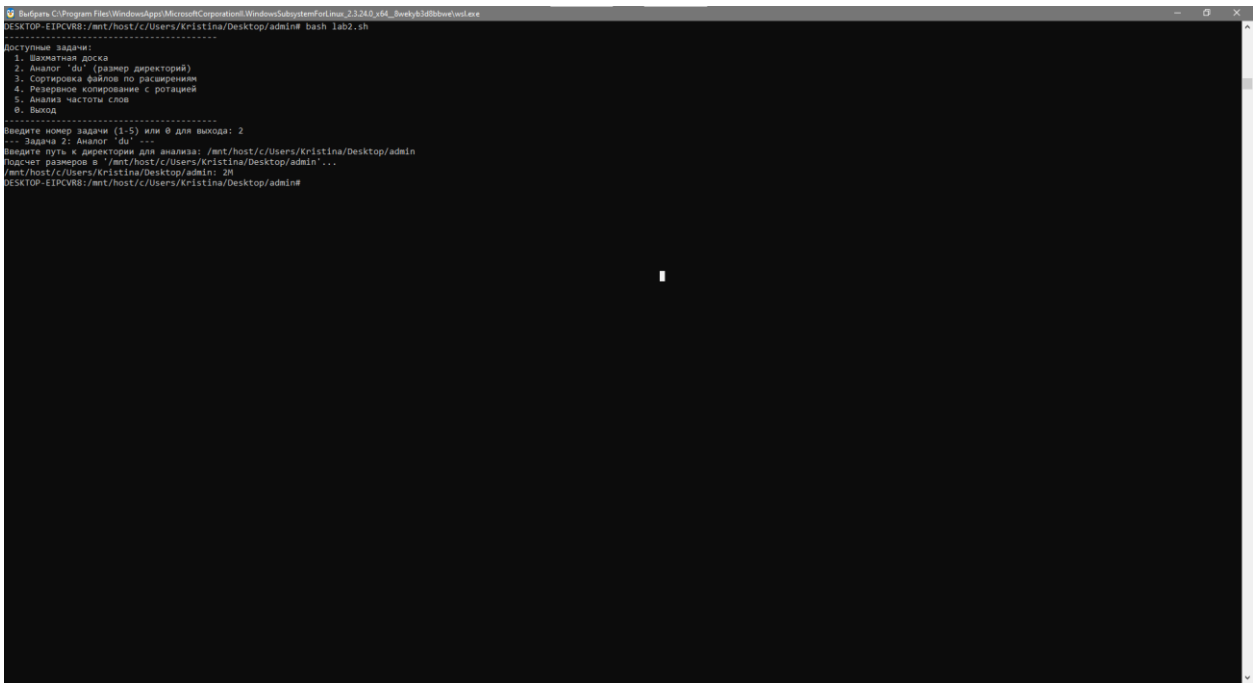
Фрагмент кода:

```
format_size_simple() {
  local bytes=$1
  local kb=$((1024))
  local mb=$((kb * 1024))
```

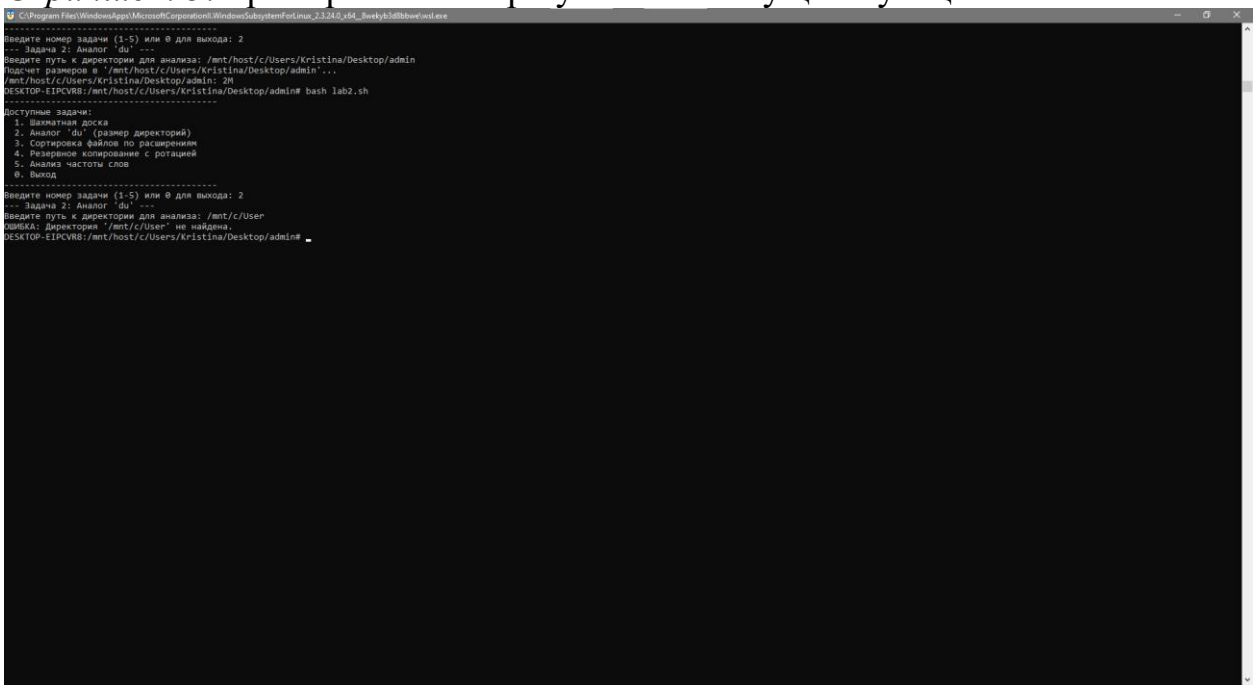
```
  local gb=$((mb * 1024))

  if (( bytes >= gb )); then
    echo "$((bytes / gb))G"
  elif (( bytes >= mb )); then
    echo "$((bytes / mb))M"
  elif (( bytes >= kb )); then
    echo "$((bytes / kb))K"
  else
    echo "${bytes}B"
  fi
}

calculate_sizes_recursive() {
  local dir_path="$1"
  local total_size=0
  local current_files_size=0

  while IFS= read -r -d $'\0' entry; do
    if [ -f "$entry" ] && [ ! -L "$entry" ]; then
      local file_size
      file_size=$(stat -c %s "$entry" 2>/dev/null)
      if [ $? -eq 0 ] && [[ "$file_size" =~ ^[0-9]+$ ]]; then
        current_files_size=$((current_files_size + file_size))
      else
        echo "Предупреждение: Не удалось получить размер '$entry'" >&2
      fi
    elif [ -d "$entry" ] && [ ! -L "$entry" ]; then
      local sub_dir_size
      sub_dir_size=$(calculate_sizes_recursive "$entry")
      if [ $? -eq 0 ] && [[ "$sub_dir_size" =~ ^[0-9]+$ ]]; then
        total_size=$((total_size + sub_dir_size))
      else
        echo "Предупреждение: Ошибка обработки поддиректории '$entry'" >&2
      fi
  fi
  done < <(find "$dir_path" -maxdepth 1 -mindepth 1 -print0 2>/dev/null)

  total_size=$((total_size + current_files_size))
  echo "$total_size"
  return 0
}

run_du_analog() {
  local target_dir="$1"
```

```
  target_dir="${target_dir%/}"

  if [ ! -d "$target_dir" ]; then
    error_exit "Директория '$target_dir' не найдена."
  fi
  if [ ! -r "$target_dir" ] || [ ! -x "$target_dir" ]; then
    error_exit "Нет прав на чтение или вход в директорию '$target_dir'."
  fi

  echo "Подсчет размеров в '$target_dir'..."
  local total
  total=$(calculate_sizes_recursive "$target_dir")
  if [[ "$total" =~ ^[0-9]+$ ]]; then
    local formatted_size
    formatted_size=$(format_size_simple "$total")
    echo "$target_dir: $formatted_size"
  else
    error_exit "Ошибка: невозможно определить размер директории."
  fi
}
```

Скриншот 2: запуск задачи 2 и ввод пути к директории/пример вывода результата в консоли.



Скриншот 3: пример ошибки при указании несуществующей папки.



Задание 3. Сортировка файлов по расширениям

Требования:

- Принимает путь к папке.
- Создаёт поддиректории по расширениям (txt, jpg, no_extension).
- Перемещает файлы в соответствующие папки.

Описание реализации:

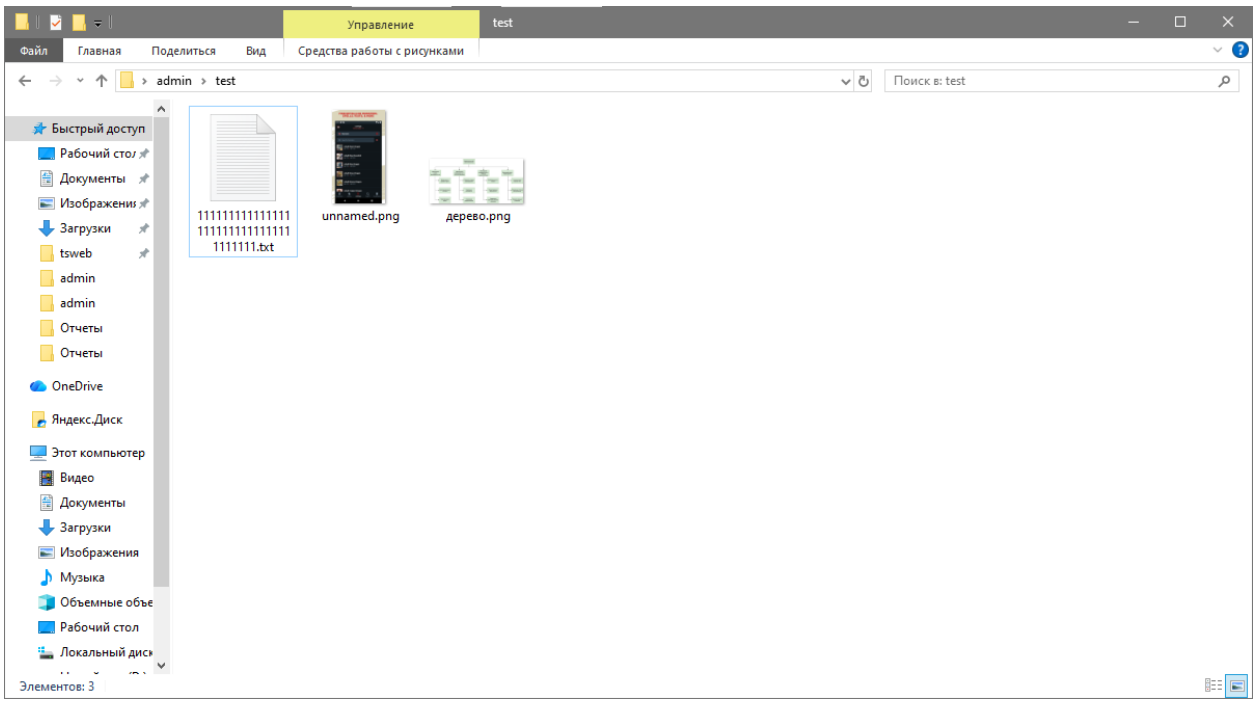
- Поиск файлов реализован через find с ограничением -maxdepth 1.
- Расширение определяется через ${filename##*.}.
- Создаются подпапки и перемещение файлов выполняется с помощью mv.

Фрагмент кода:

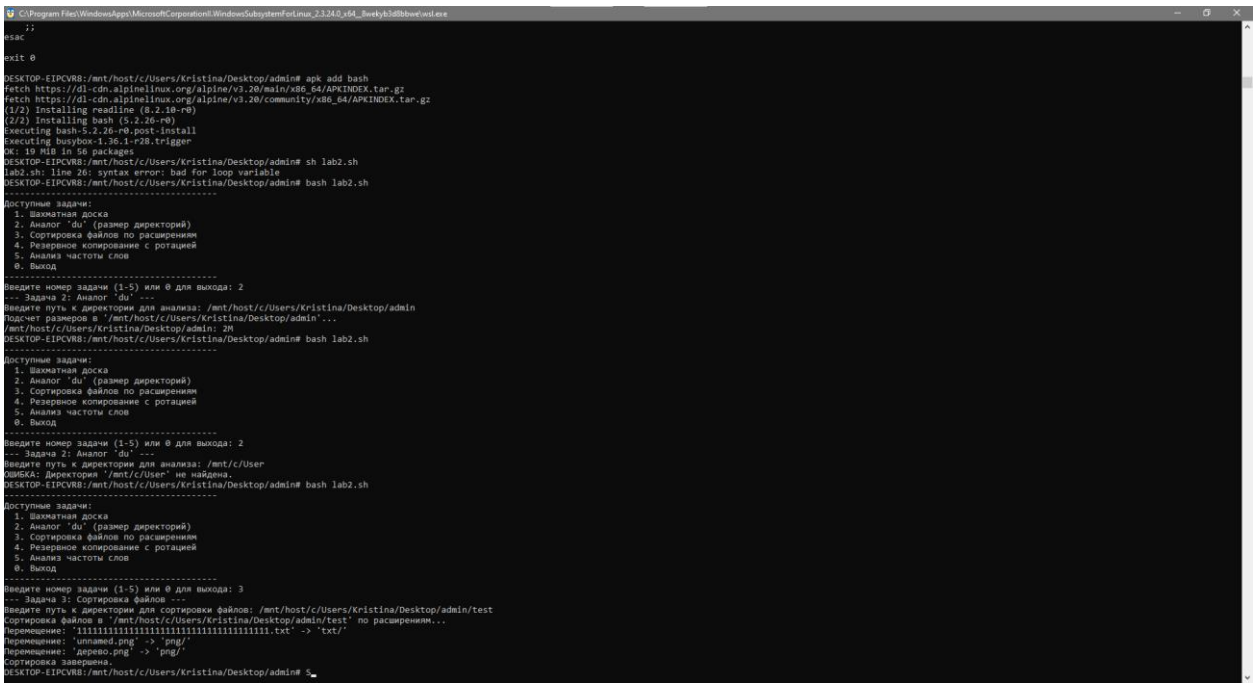
```
sort_files_by_extension() {
  local target_dir="$1"
  target_dir="${target_dir%/}"

  if [ ! -d "$target_dir" ]; then
    error_exit "Директория '$target_dir' не найдена."
  fi
  if [ ! -r "$target_dir" ] || [ ! -w "$target_dir" ] || [ ! -x "$target_dir" ]; then
    error_exit "Нет прав на чтение/запись/вход в директорию '$target_dir'."
  fi

  echo "Сортировка файлов в '$target_dir' по расширениям..."
  find "$target_dir" -maxdepth 1 -type f -print0 | while IFS= read -r -d $'\0'
file_path; do
    local filename
    filename=$(basename "$file_path")
    local extension="${filename##*.}"
    local subdir_name

    if [[ "$filename" == "$extension" ]] || [[ "$filename" == .* && "${filename#.*}"
== "" ]]; then
        subdir_name="no_extension"
    else
        subdir_name="${extension,,}"
    fi
```

```
    local dest_dir="$target_dir/$subdir_name"
    mkdir -p "$dest_dir"
    if [ $? -ne 0 ]; then
        echo "Предупреждение: Не удалось создать папку '$dest_dir'. Пропуск файла '$filename'." >&2
        continue
    fi

    echo "Перемещение: '$filename' -> '$subdir_name/'"
    mv -f "$file_path" "$dest_dir/"
    if [ $? -ne 0 ]; then
        echo "Предупреждение: Не удалось переместить '$filename' в '$dest_dir'." >&2
    fi
  done
  echo "Сортировка завершена."
}
```

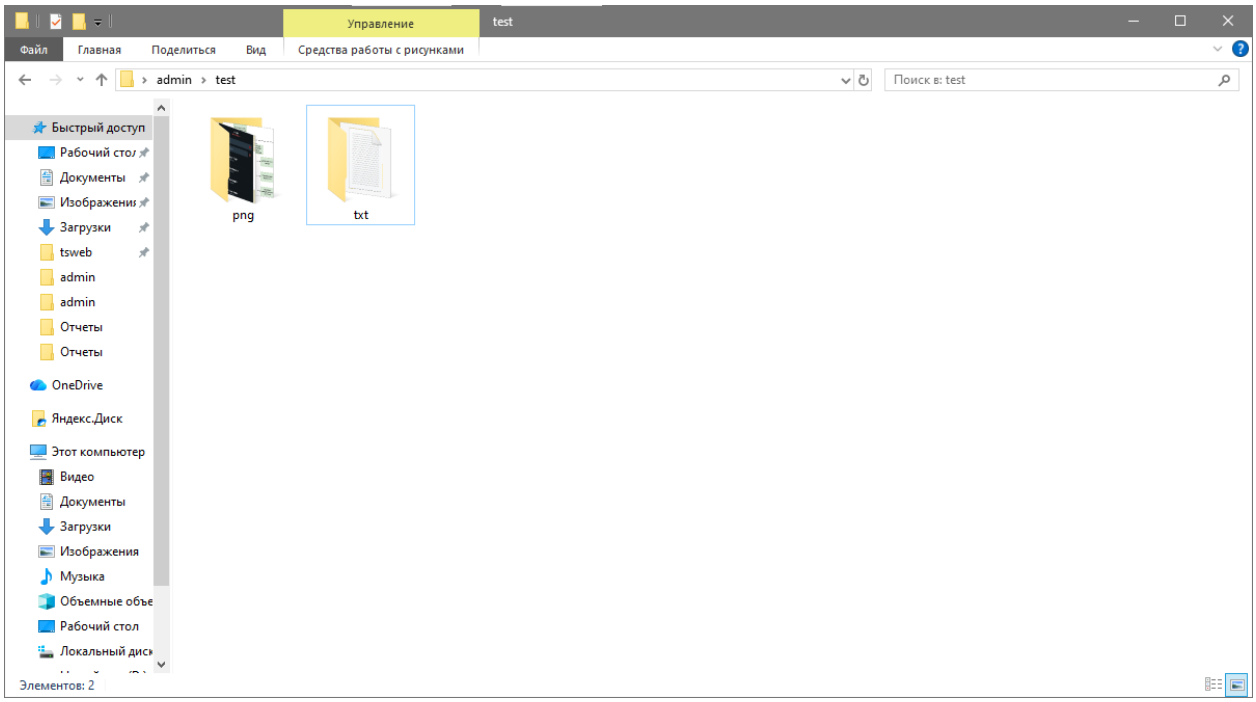
Скриншот 4: структура исходной папки до сортировки.



Скриншот 5: выполнение скрипта (пункт 3).



Скриншот 6: папка после сортировки по расширениям.



Задание 4. Резервное копирование с ротацией

Требования:

- Создание архива (tar.gz) указанной директории с датой в названии.
- Удаление архивов старше 7 дней.
- Проверка успешности создания архива.

Описание реализации:

- Используется команда tar -czf с параметром -C.
- После создания архива выполняется проверка кода возврата.
- Реализована автоматическая ротация старых архивов через find -mtime +7 -delete.
- Добавлена проверка на совпадение исходной и целевой папки.

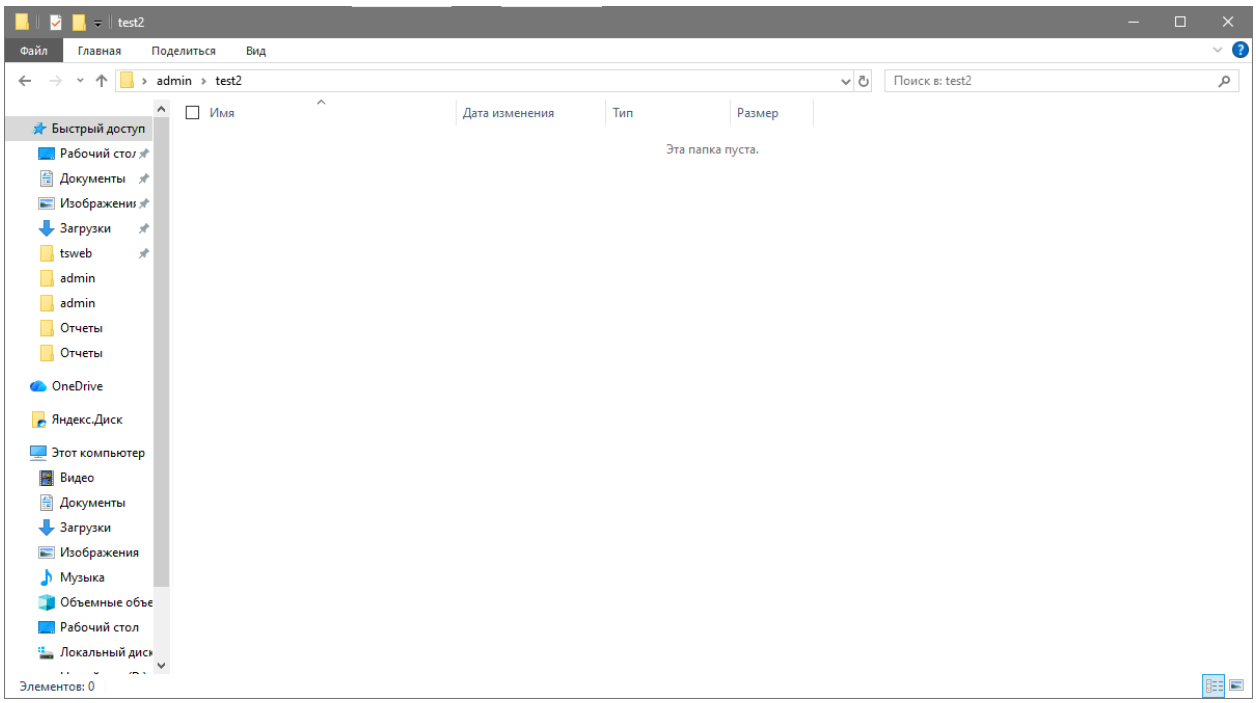
Фрагмент кода:

```
create_backup_with_rotation() {

  local source_dir="$1"

  local backup_dir="$2"

  source_dir="${source_dir%/}"

  backup_dir="${backup_dir%/}"

  local days_to_keep=7



  # Проверка на совпадение путей

  if [ "$source_dir" = "$backup_dir" ]; then

    error_exit "Исходная директория и директория для бэкапов не могут совпадать."

  fi



  # Проверка что backup_dir не находится внутри source_dir

  if [[ "$backup_dir" == "$source_dir"/* ]]; then

    error_exit "Директория для бэкапов не может находиться внутри исходной директории."

  fi



  if [ ! -d "$source_dir" ]; then

    error_exit "Исходная директория '$source_dir' не найдена."
```

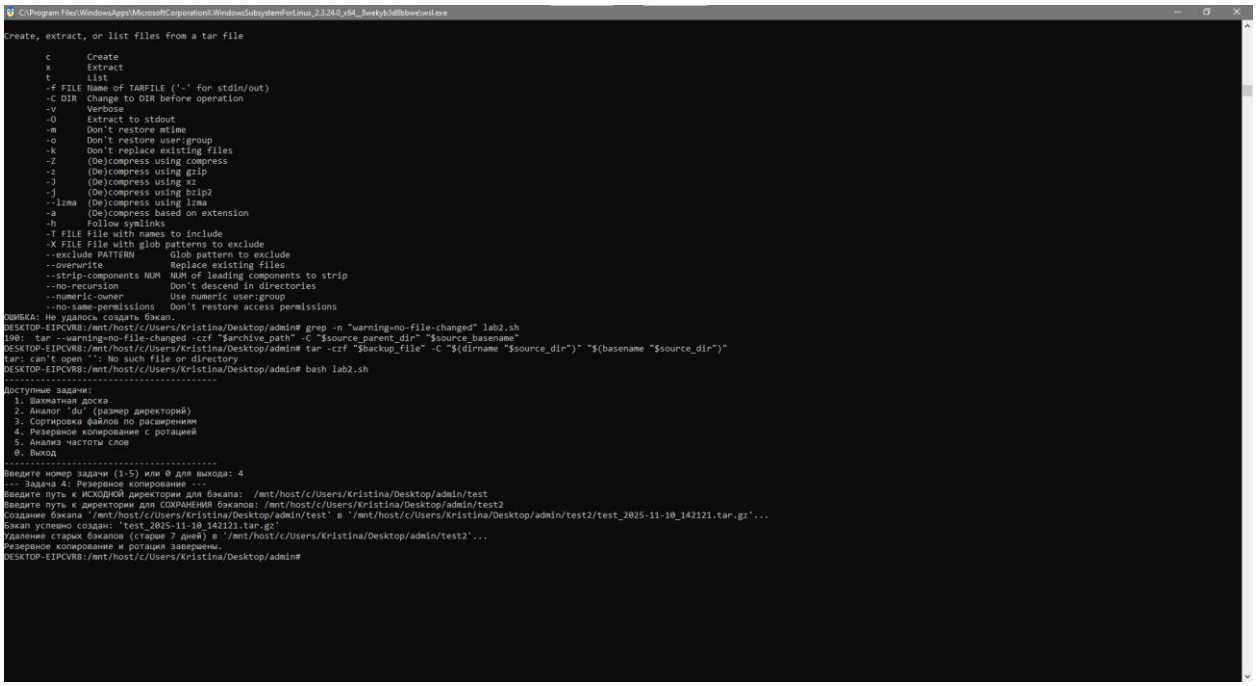
```
 fi

 if [ ! -r "$source_dir" ] || [ ! -x "$source_dir" ]; then

   error_exit "Нет прав на чтение или вход в исходную директорию
'$source_dir'."

 fi



 if [ ! -d "$backup_dir" ]; then

   echo "Инфо: Создаю директорию для бэкапов '$backup_dir'..."

   mkdir -p "$backup_dir" || error_exit "Не удалось создать директорию
'$backup_dir'."

 fi

 if [ ! -w "$backup_dir" ] || [ ! -x "$backup_dir" ]; then

   error_exit "Нет прав на запись или вход в директорию для бэкапов
'$backup_dir'."

 fi



 local datestamp

 datestamp=$(date +%Y-%m-%d_%H%M%S)

 local source_basename

 source_basename=$(basename "$source_dir")

 local archive_name="${source_basename}_${datestamp}.tar.gz"

 local archive_path="$backup_dir/$archive_name"

 local source_parent_dir

 source_parent_dir=$(dirname "$source_dir")

 echo "Создание бэкапа '$source_dir' в '$archive_path'..."

 tar -czf "$archive_path" -C "$source_parent_dir" "$source_basename"
```

```
 if [ $? -eq 0 ]; then

   echo "Бэкап успешно создан: '$archive_name'"

 else

   rm -f "$archive_path"

   error_exit "Не удалось создать бэкап."

 fi



 echo "Удаление старых бэкапов (старше $days_to_keep дней) в
'$backup_dir'..."

 find "$backup_dir" -name "${source_basename}_*.tar.gz" -type f -mtime
"+$((days_to_keep - 1))" -print -delete

 if [ $? -ne 0 ]; then

   echo "Предупреждение: Возникли ошибки при поиске или удалении
старых бэкапов." >&2

 fi

 echo "Резервное копирование и ротация завершены."
```

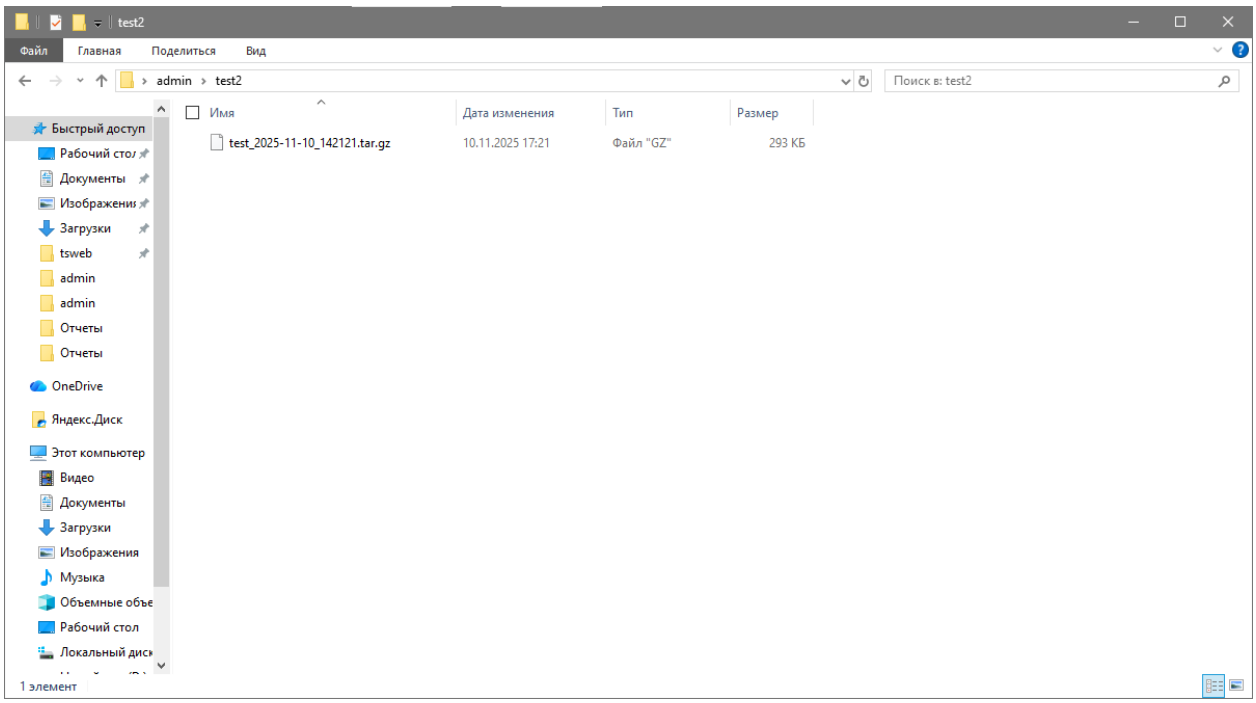
}*Скриншот 7:* запуск задачи 4 и ввод путей к исходной и целевой папке.



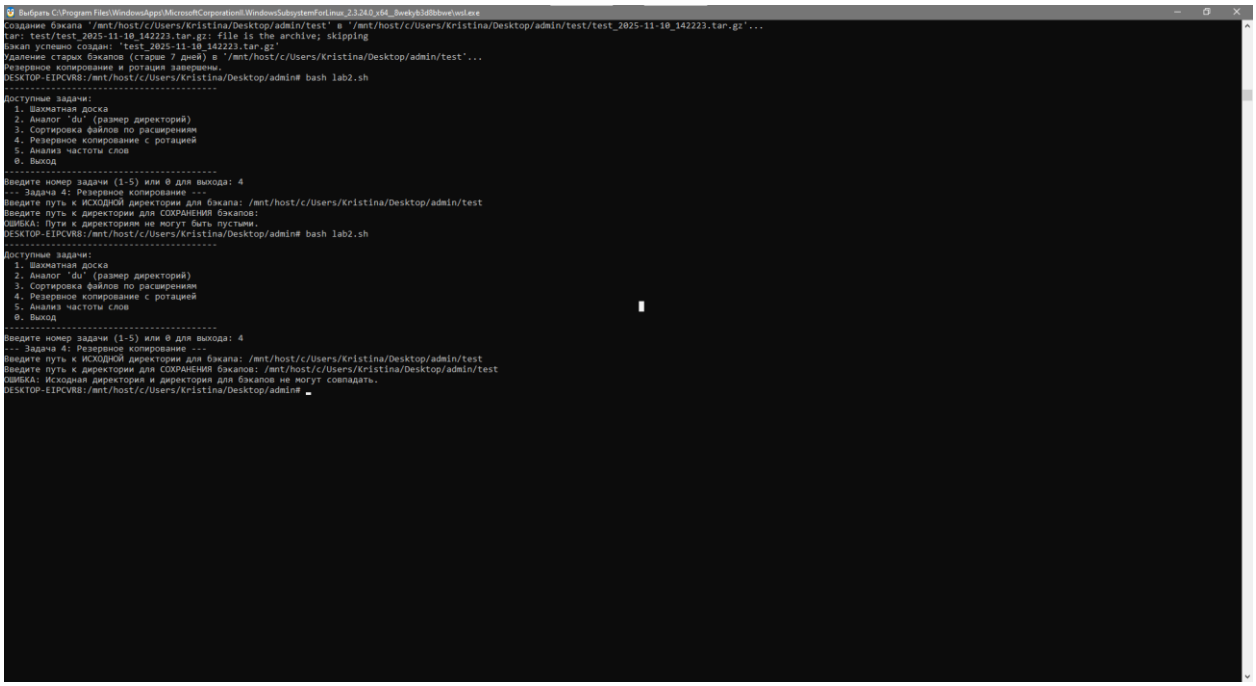
Скриншот 8: консольный вывод успешного создания архива.



Скриншот 9: содержимое папки с готовым архивом.



Скриншот 10: пример ошибки при совпадении путей (архивация самой себя).



Задание 5. Анализ частоты слов в текстовых файлах

Требования:

- Принимает три аргумента: директория, расширение файлов, количество N.
- Подсчитывает частоту слов без учёта регистра.
- Игнорирует знаки препинания.
- Выводит топ-N самых частых слов.

Описание реализации:

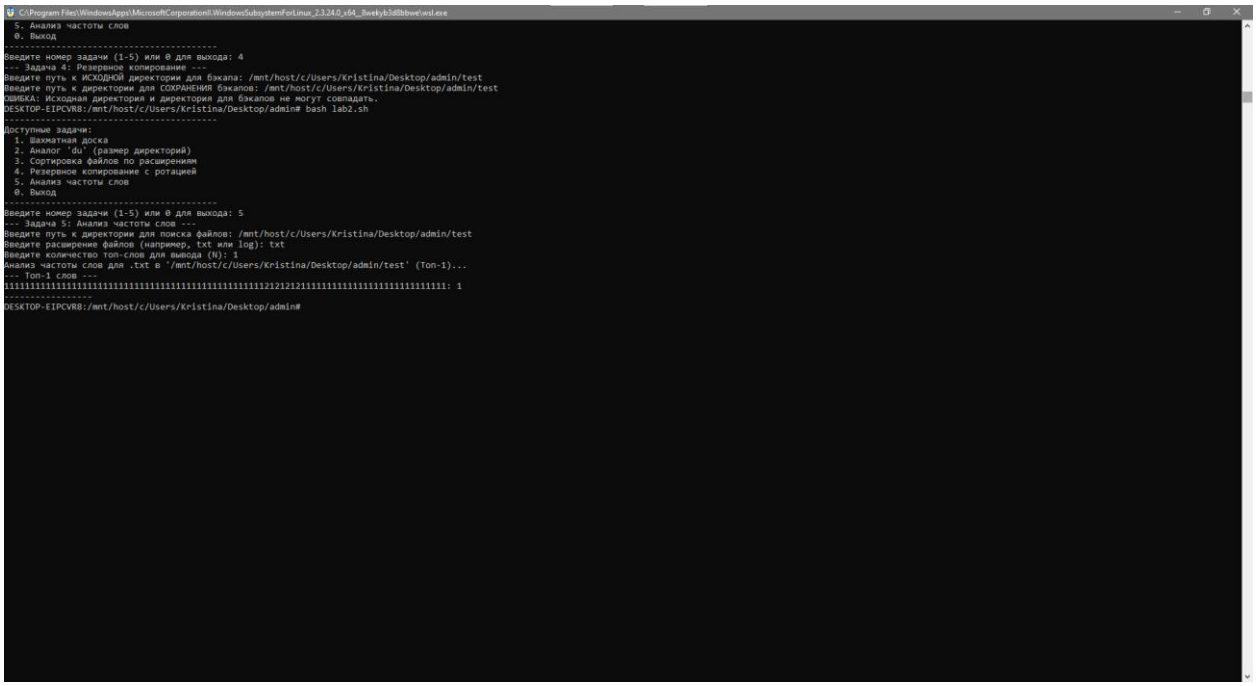
- Слова извлекаются через grep -ohE '[[:alnum:]]+'.
- Используется tr для перевода текста в нижний регистр.
- Подсчёт частот — через sort | uniq -c | sort -nr | head -n N.
- Добавлена проверка на отсутствие подходящих файлов.

Фрагмент кода:

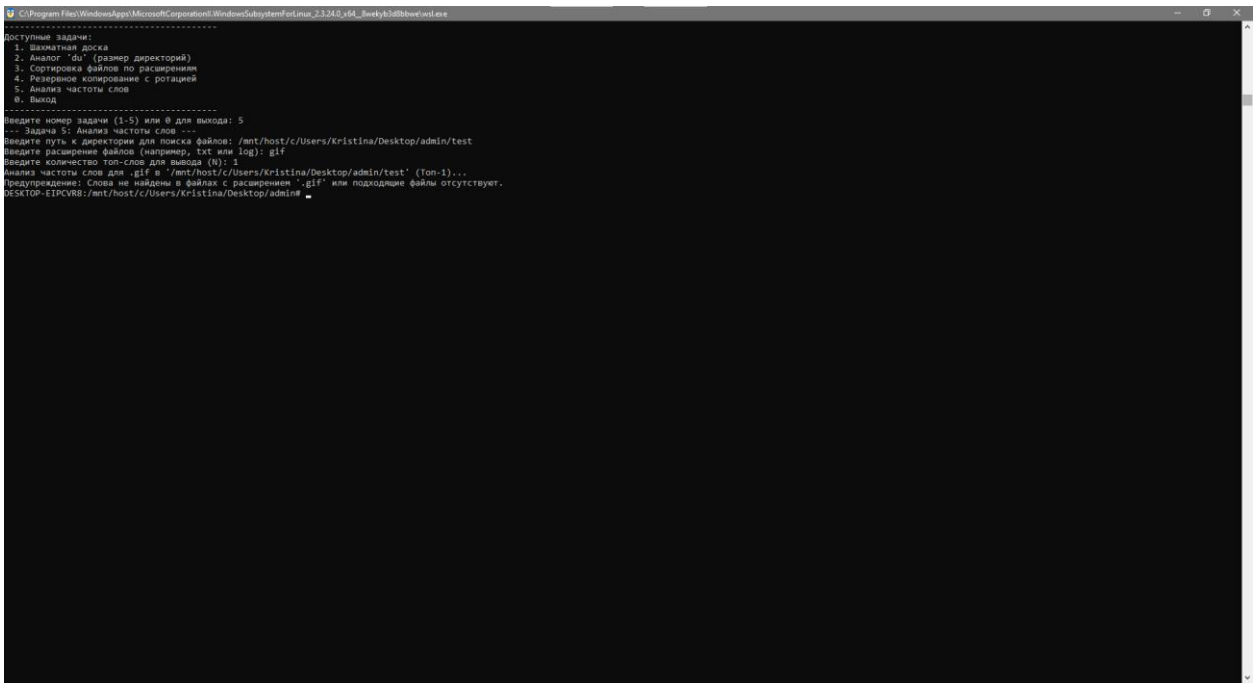
```
analyze_word_frequency() {
  local search_dir="$1"
  local extension="$2"
  local top_n="$3"
  search_dir="${search_dir%/}"
  if [ ! -d "$search_dir" ]; then
    error_exit "Директория '$search_dir' не найдена."
  fi
  if ! [[ "$top_n" =~ ^[1-9][0-9]*$ ]]; then
    error_exit "Внутренняя ошибка: <top_n> должно быть числом."
  fi
  if [ ! -r "$search_dir" ] || [ ! -x "$search_dir" ]; then
    error_exit "Нет прав на чтение или вход в директорию '$search_dir'."
  fi

  echo "Анализ частоты слов для .$extension в '$search_dir' (Топ-$top_n)..."
  local result
  result=$(find "$search_dir" -type f -name "*.$extension" -print0 2>/dev/null | \
          xargs -0 cat -- 2>/dev/null | \
          grep -ohE '\w+' | \
          tr '[:upper:]' '[:lower:]' | \
          sort | \
          uniq -c | \
          sort -nr | \
```

```
        head -n "$top_n" | \
        awk '{print $2 ": " $1}')


 if [ -z "$result" ]; then
   echo "Предупреждение: Слова не найдены в файлах с расширением '.$extension' или подходящие файлы отсутствуют."
 else
   echo "--- Топ-$top_n слов ---"
   echo "$result"
   echo "-----------------"
 fi
}
```

Скриншот 11: запуск задачи 5 и ввод параметров (директория, расширение, N).



Скриншот 12: пример ошибки при выполнении.


C:\Program Files\WindowsApps\MicrosoftCorporationII.WindowsSubsystemForLinux_2.3.24.0_x64__8wekyb3d8bbwe\wsl.exe
----
Доступные задачи:
1. Шахматная доска
2. Аналог 'du' (размер директорий)
3. Сортировка файлов по расширениям
4. Резервное копирование с ротацией
5. Анализ частоты слов
0. Выход

Введите номер задачи (1-5) или 0 для выхода: 5
--- Задача 5: Анализ частоты слов ---
Введите путь к директории для поиска файлов: /mnt/host/c/Users/Kristina/Desktop/admin/test
Введите расширение файлов (например, txt или log): gif
Введите количество топ-слов для вывода (N): 1
Анализ частоты слов для .gif в '/mnt/host/c/Users/Kristina/Desktop/admin/test' (Топ-1)...
Предупреждение: Слова не найдены в файлах с расширением '.gif' или подходящие файлы отсутствуют.
DESKTOP-EIPCVR8:/mnt/host/c/Users/Kristina/Desktop/admin#