# Лабораторная работа № 2

## Программирование на Bash в среде Linux

**Цель:** Освоение базовых и продвинутых возможностей Bash, работа с файловой системой, алгоритмизация, управление процессами.

---

## Задание 1: Шахматная доска в консоли

**Требования:**

1. Пользователь вводит размер доски (например, 8 → 8x8).
2. Доска должна отображаться в консоли с чередованием цветов (например, белый/чёрный, синий/зелёный).
3. Поддержка как чётных, так и нечётных размеров.
4. Использовать ANSI-коды для цветов.

---

## Задание 2: Аналог `du` (подсчёт дискового пространства)

**Требования:**

1. Скрипт принимает путь к папке (например, `./script.sh /home/user`).
2. Рекурсивно подсчитывает объём файлов и поддиректорий.
3. Выводит результат в человеко-читаемом формате (например, "2.5G", "100M").
4. Запрещено использовать `du`, `ls` — только `find`, `stat`, арифметические операции.

**Пример вывода:**

```
/home/user/docs: 150M
/home/user/images: 2.1G
```

**Подсказка:**
Используйте `find $dir -type f -printf "%s\n"` для получения размеров файлов.

---

## Задание 3: Сортировка файлов по расширениям в соответствующие подпапки

**Требования:**

1. Скрипт принимает путь к папке.
2. Создает поддиректории с именами, соответствующими расширениям файлов (например, `txt`, `pdf`).
3. Перемещает файлы в соответствующие папки.
4. Файлы без расширения помещаются в папку `no_extension`.

**Пример:**
Для файлов `a.txt`, `b.jpg`, `README` → создаются папки `txt`, `jpg`, `no_extension`.

---

## Задание 4: Резервное копирование с ротацией

**Требования:**

1. Скрипт создает архив (tar.gz) указанной директории с датой в имени.
2. Автоматически удаляет архивы старше 7 дней.
3. Проверяет успешность создания архива.

**Пример команды:**
`./backup.sh /var/www /backups`

## Задание 5: Анализ частоты слов в текстовых файлах

**Требования:**

1. Скрипт принимает три аргумента: путь к директории, расширение файлов (например, `txt`), и число `N` (топ-N слов).
2. Рекурсивно обрабатывает все файлы с указанным расширением в директории и её поддиректориях.
3. Подсчитывает количество вхождений каждого слова (без учёта регистра, игнорируя знаки препинания).
4. Выводит топ-N самых частых слов в формате `Слово: Количество`.

**Пример вызова:**

```
./wordstats.sh /var/log txt 10
```

**Пример вывода:**

```
error: 245
warning: 198
server: 156
connection: 120
...
```

**Подсказка:**

- Используйте `grep -ohE "\w+"` для извлечения слов.
- Примените `tr '[:upper:]' '[:lower:]'` и `tr -d '[:punct:]'` для нормализации.
- Используйте ассоциативные массивы Bash для подсчёта частот.
- Сортируйте результат с помощью `sort` и `head`.

**Дополнительно:**

- Реализуйте игнорирование стоп-слов (например, предлогов из файла-исключений).
- Добавьте проверку на пустые файлы или отсутствие подходящих данных.

---

# Рекомендации:

1. Используйте `#!/bin/bash` в начале каждого скрипта.
2. Добавьте проверки на ошибки (например, существование директорий, права доступа).
3. Оформите код в виде функций для повышения читаемости.

**Критерии оценки:**

- Корректность выполнения задач.
- Обработка ошибок и граничных случаев.
- Оптимизация кода (минимизация вызовов внешних утилит).
- Читаемость и структурированность скриптов.

**Форма отчёта:**

- Исходный код скриптов.
- Примеры выполнения с разными входными данными.
- Описание проблем, с которыми столкнулись, и их решение.