

Отчет по лабораторной работе № 13

**Программирование в командном процессоре ОС Unix. Ветвления и
циклы**

Скобеева Алиса Алексеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	12
5	Ответы на контрольные вопросы	13

Список иллюстраций

3.1	Пишем скрипт	7
3.2	Пишем текст в файле	7
3.3	Все сработало корректно	8
3.4	Скрипт программы	8
3.5	Все работает корректно	9
3.6	Пишем скрипт	9
3.7	Программа работает корректно	10
3.8	Пишем скрипт	10
3.9	Последовательно вводим команды	11
3.10	Программа работает корректно	11

Список таблиц

1 Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научится писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

2 Задание

Написать несколько программ используя ветвления и циклы.

3 Выполнение лабораторной работы

Пишем командный файл, который анализирует командную строку с ключами, а затем ищет в указанном файле строки, определяемые ключом -r.

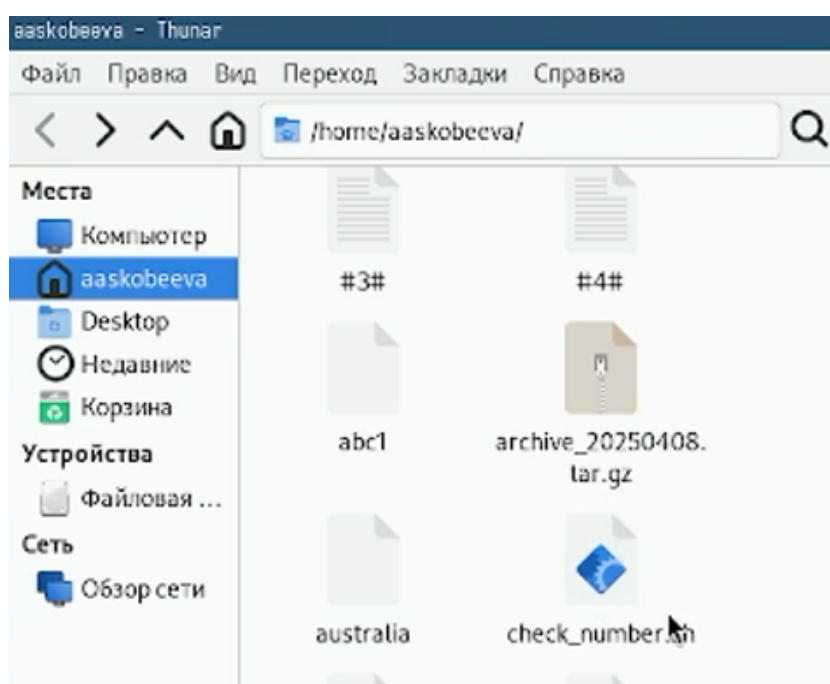


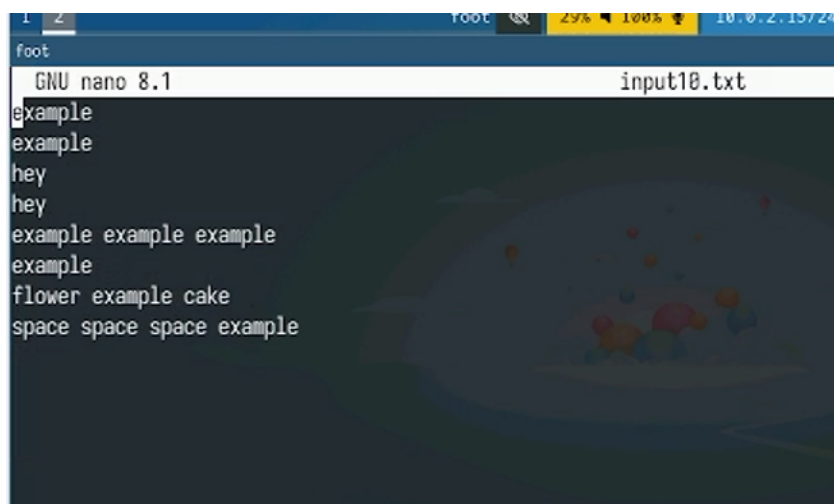
Рис. 3.1: Пишем скрипт

Создаем файл, в котором пишем некоторый текст

```
[aaskobeeva@fedora ~]$ ./search.sh -i input10.txt -r "example" -o output.txt -C -n
[aaskobeeva@fedora ~]$ nano output.txt
[aaskobeeva@fedora ~]$ nano input10.txt
```

Рис. 3.2: Пишем текст в файле

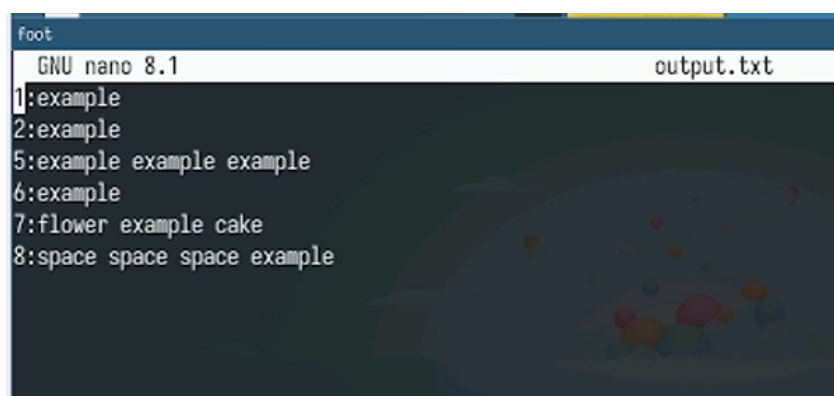
Запускаем командный файл, и проверяем результат работы в файле output.txt



```
foot
GNU nano 8.1 input10.txt
example
example
hey
hey
example example example
example
flower example cake
space space space example
```

Рис. 3.3: Все сработало корректно

Пишем следующую программу, которая определяет, что число больше, меньше или равно нулю



```
foot
GNU nano 8.1 output.txt
1:example
2:example
5:example example example
6:example
7:flower example cake
8:space space space example
```

Рис. 3.4: Скрипт программы

Проверяем корректность работы программы


```
GNU nano 8.1 check_number.sh
#!/bin/bash

read -p "Введите число: " num

# Проверка, является ли ввод числом. Ограничение: допускаются только целые числа
if ! [[ "$num" =~ ^-[0-9]+$ ]]; then
    echo "Ошибка: Неверный ввод. Введите целое число."
    exit 2
fi

# Сравнение числа и вывод сообщения
if (( num > 0 )); then
    echo "Число больше нуля."
    exit 0
elif (( num < 0 )); then
    echo "Число меньше нуля."
    exit 1
else
    echo "Число равно нулю."
    exit 3
fi

exit 0 # На всякий случай, если ни один из условий не сработает (чего быть не должно)
```

Рис. 3.5: Все работает корректно

Пишем командный файл, создающий указанное число файлов, а также умеющий их удалять

```
[aaskobeeva@fedora ~]$ ./check_number.sh
Введите число: 4
Число больше нуля.
[aaskobeeva@fedora ~]$ ./check_number.sh
Введите число: 0
Число равно нулю.
```

Рис. 3.6: Пишем скрипт

Проверяем корректность работы программы

```
GNU nano 8.1                                filemake1.sh
exit 1
fi

# Создание файлов
N="$1"
for i in $(seq 1 $N); do
    touch "$i.tmp"
done

echo "Создано $N файлов."

# Функция для удаления файлов
cleanup() {
    echo "Удаление временных файлов..."
    for i in $(seq 1 $N); do
        rm -f "$i.tmp"
    done
    echo "Временные файлы удалены."
}

# Вызов cleanup при выходе (Ctrl+C, exit и т.д.)
trap cleanup EXIT

exit 0
```

Рис. 3.7: Программа работает корректно

Пишем командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории

```
[aaskobeeva@fedora ~]$ ./filemake1.sh 5
Создано 5 файлов.
Удаление временных файлов...
Временные файлы удалены.
[aaskobeeva@fedora ~]$ ./filemake1.sh 14
Создано 14 файлов.
Удаление временных файлов...
Временные файлы удалены.
```

Рис. 3.8: Пишем скрипт

Запускаем файл

```
foot
GNU nano 8.1                                lasttask.sh
#!/bin/bash

# Скрипт для архивирования файлов, измененных менее недели назад

# Проверка наличия аргумента (каталога)
if [ -z "$1" ]; then
    echo "Использование: $0 <каталог>"
    exit 1
fi

directory="$1"

# Проверка, что каталог существует
if [ ! -d "$directory" ]; then
    echo "Ошибка: Каталог '$directory' не существует."
    exit 1
fi

# Определение имени архива
archive_name="archive_$(date +%Y%m%d).tar.gz"

# Поиск файлов, измененных менее 7 дней назад
find "$directory" -type f -mtime -7 -print0 | tar -czvf "$archive_name" --null -T -

echo "Архив '$archive_name' создан."
```

Рис. 3.9: Последовательно вводим команды

Проверяем корректность работы программы

```
[aaskobeeva@fedora ~]$ ./lasttask.sh /home/aaskobeeva/lab71
tar: Удаляется начальный '/' из имен объектов
/home/aaskobeeva/lab71/lol.txt
tar: Удаляются начальные '/' из целей жестких ссылок
/home/aaskobeeva/lab71/dura.txt
Архив 'archive_20250408.tar.gz' создан.
[aaskobeeva@fedora ~]$
```

Рис. 3.10: Программа работает корректно

4 Выводы

Мы успешно написали 4 командных файла и проверили корректность их работы.

5 Ответы на контрольные вопросы

1. `getopts`: Разбор параметров командной строки с использованием стандартного синтаксиса (`-a`, `-b value`).
2. Метасимволы и имена файлов: Метасимволы (`*`, `?`, `[]`) расширяются в имена файлов, соответствующие шаблону (globbing).
3. Операторы управления действиями: `&&` (и), `||` (или), `;` (последовательное выполнение).
4. Прерывание цикла: `break` (выход из цикла), `continue` (переход к следующей итерации).
5. `false/true`: Команды, всегда возвращающие `false` (1) или `true` (0) соответственно. Удобны для упрощения логических выражений.
6. `if test -f mans/i.$s`: Проверка, существует ли обычный файл с именем `man[значение переменной s]/[значение переменной i].[значение переменной s]`.
7. `while/until`:
 - `while condition`: Цикл выполняется, пока `condition` истинно.
 - `until condition`: Цикл выполняется, пока `condition` ложно