

Лабораторная работа №13

Презентация

Устинова В. В.

10 мая 2025

Российский университет дружбы народов, Москва, Россия

Информация

- Устинова Виктория Вадимовна
- студент НПИбд-01-24
- Российский университет дружбы народов

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

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами
2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю.
3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до `?`
4. Написать командный файл, который с помощью команды `tar` запаковывает в архив все файлы в указанной директории.

Выполняем первое задание, ключи: `-n` — выдавать номера строк, `iinputfile` — прочитать данные из указанного файла



```
35     exit 1
36 ;;
37 esac
38 done
39
40 # Проверяем наличие обязательных параметров
41 if [ -z "$pattern" ]; then
42     echo "Не указан шаблон для поиска" >&2
43     exit 1
44 fi
45
46 if [ -z "$inputfile" ]; then
47     echo "Не указан входной файл" >&2
48     exit 1
49 fi
50
51 # Формируем опции для grep
52 if [ "$case_sensitive" = true ]; then
53     grep_options+=" -i" # -i делает поиск нечувствительным к регистру
54 fi
55
56 if [ "$line_numbers" = true ]; then
57     grep_options+=" -n" # -n выводит номера строк
58 fi
59
60 # Выполняем поиск и выводим результаты в файл или на экран
61 if [ -n "$outputfile" ]; then
62     grep $grep_options "$pattern" "$inputfile" > "$outputfile"
63 else
64     grep $grep_options "$pattern" "$inputfile"
65 fi
66
67 exit 0
```

Текс файла

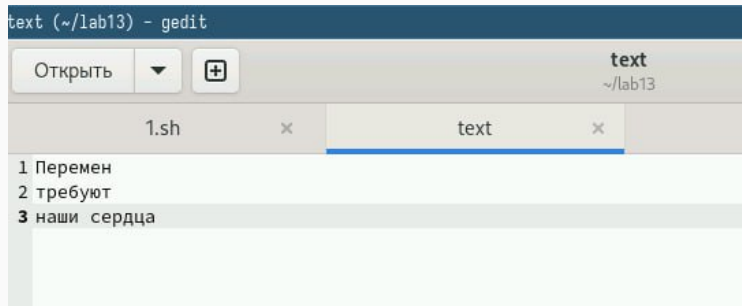


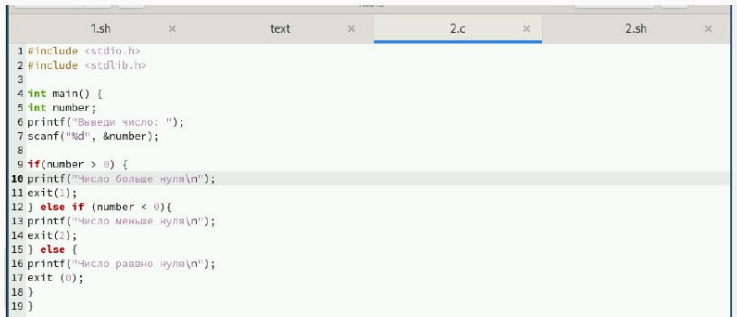
Рис. 2: Файл txt

Используем определенные ключи

```
[vvustinova@vvustinova lab13]$ ./1.sh -p "сердца" -i text
наши сердца
[vvustinova@vvustinova lab13]$ ./1.sh -p "сердца" -i text -n
3:наши сердца
```

Рис. 3: Правильный вывод

Написать программу, которая определяет число $>$ $<$ нуля



```
1.sh x text x 2.c x 2.sh x
1 #include <stdio.h>
2 #include <stdlib.h>
3
4 int main() {
5     int number;
6     printf("Введи число: ");
7     scanf("%d", &number);
8
9     if(number > 0) {
10    printf("Число больше нуля\n");
11    exit(1);
12 } else if (number < 0){
13    printf("Число меньше нуля\n");
14    exit(2);
15 } else {
16    printf("Число равно нулю\n");
17    exit (0);
18 }
19 }
```

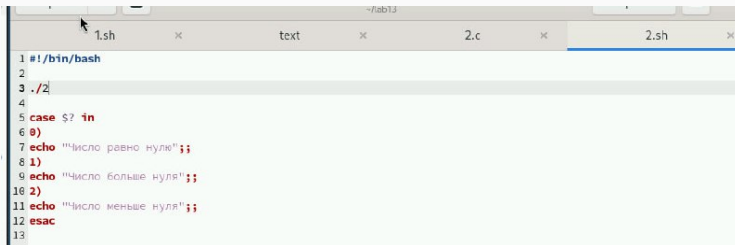
Рис. 4: Пишем программу в 2.c

Открываем программу

```
[vvustinova@vvustinova lab13]$ gcc 2.c -o 2  
[vvustinova@vvustinova lab13]$ ./2  
Выведи число: 13  
Число больше нуля
```

Рис. 5: Смотрим вывод

Команд- ный файл должен вызывать эту программу и, проанализировав с помощью команды \$?, выдать сообщение о том, какое число было введено



```
1 #!/bin/bash
2
3 ./2
4
5 case $? in
6 0)
7 echo "число равно нулю";;
8 1)
9 echo "число больше нуля";;
10 2)
11 echo "число меньше нуля";;
12 esac
13
```

Рис. 6: Используем расширение 2.sh

Открываем программу

работы;
эти от

```
[vvustinova@vvustinova lab13]$ ./2.sh
Выведи число: 4
Число больше нуля
Число больше нуля
[vvustinova@vvustinova lab13]$ ./2.sh
Выведи число: -9
Число меньше нуля
Число меньше нуля
[vvustinova@vvustinova lab13]$ ./2.sh
Выведи число: 0
Число равно нулю
Число равно нулю
```

Рис. 7: Смотрим вывод

Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы

```
--
12 # Функция для удаления файлов
13 delete_files() {
14     local count=$1
15     for ((i=1; i<=count; i++)); do
16         if [ -e "$i.tmp" ]; then
17             rm "$i.tmp"
18             echo "удален файл $i.tmp"
19         fi
20     done
21 }
22
23 # Проверяем, передан ли аргумент (количество файлов)
24 if [ $# -eq 0 ]; then
25     echo "Не указано количество файлов для создания"
26     exit 1
27 fi
28
29 # Определяем действие на основе первого аргумента
30 action=$1
31
32 # В зависимости от действия, вызываем соответствующую функцию
33 case "$action" in
34     create)
35         create_files "$2"
36     ;;
37     ..
38 ..
```

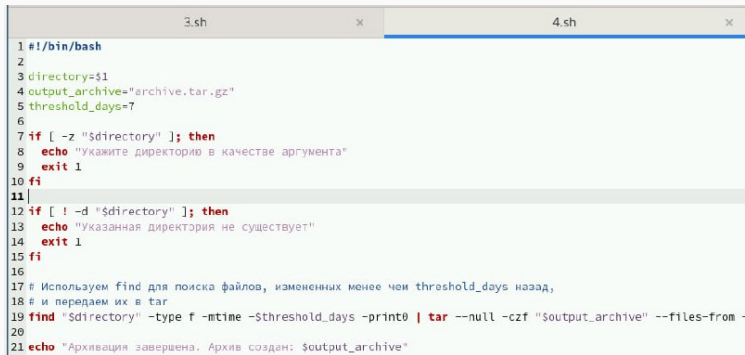
Рис. 8: Пишем программу

Открываем ее

```
[vvustinova@vvustinova lab13]$ ./3.sh create 3  
Создан файл 1.tmp  
Создан файл 2.tmp  
Создан файл 3.tmp  
[vvustinova@vvustinova lab13]$ ./3.sh delete 3  
Удален файл 1.tmp  
Удален файл 2.tmp  
Удален файл 3.tmp
```

Рис. 9: Показывает создание и удаление

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



```
3.sh x 4.sh x
1 #!/bin/bash
2
3 directory=$1
4 output_archive="archive.tar.gz"
5 threshold_days=7
6
7 if [ -z "$directory" ]; then
8     echo "укажите директорию в качестве аргумента"
9     exit 1
10 fi
11
12 if [ ! -d "$directory" ]; then
13     echo "указанная директория не существует"
14     exit 1
15 fi
16
17 # Используем find для поиска файлов, измененных менее чем threshold_days назад,
18 # и передаем их в tar
19 find "$directory" -type f -mtime -$threshold_days -print0 | tar --null -czf "$output_archive" --files-from -
20
21 echo "Архивация завершена. Архив создан: $output_archive"
```

Рис. 10: Расширение sh, пишем командный файл

Заходим в новую папку



Рис. 11: Распаковываем

Мы успешно изучили основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.