Лабораторная работа №13

Отчет

Зубов Иван Александрович

Содержание

Список иллюстраций

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

1. Используя команды getopts grep, написать командный файл, который анализирует командную строку с ключами: – -iinputfile — прочитать данные из указанного файла; – -ooutputfile — вывести данные в указанный файл; – -ршаблон — указать шаблон для поиска; – -C — различать большие и малые буквы; – -n — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом -p.
2. Написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции exit(n), передавая информацию в о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено.
3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до (например 1.tmp, 2.tmp, 3.tmp,4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).
4. Написать командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find).

3 Выполнение лабораторной работы

Создаем файлы в каталоге lab13 и используя команды getopts grep, пишем командный файл, который анализирует командную строку с ключами: – -iinputfile — прочитать данные из указанного файла; – -ooutputfile — вывести данные в указанный файл; – -pшаблон — указать шаблон для поиска; – -C — различать большие и малые буквы; – -n — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом -p

```bash
#!/bin/bash

# Инициализация переменных
input_file=""
output_file=""
pattern=""
case_sensitive=0
line_numbers=0

# Разбор аргументов командной строки
while getopts "i:o:p:Cn" opt; do
  case $opt in
    i) input_file="$OPTARG" ;;
    o) output_file="$OPTARG" ;;
    p) pattern="$OPTARG" ;;
    C) case_sensitive=1 ;;
    n) line_numbers=1 ;;
    *) echo "Неверный параметр: -$OPTARG" >&2
       echo "Использование: $0 [-i inputfile] [-o outputfile] [-p шаблон] [-C] [-n]" >&2
       exit 1 ;;
  esac
done

# Проверка обязательных параметров
if [ -z "$pattern" ]; then
  echo "Ошибка: не указан шаблон для поиска (параметр -p)" >&2
  exit 1
fi

if [ -z "$input_file" ]; then
  echo "Ошибка: не указан входной файл (параметр -i)" >&2
  exit 1
fi

# Проверка существования входного файла
if [ ! -f "$input_file" ]; then
  echo "Ошибка: файл '$input_file' не существует или недоступен" >&2
  exit 1
fi

# Формирование команды grep
```

Рис. 3.1: Пишем код

```
bash: /usr/libexec/mc/mc.sh: Нет такого файла или каталога
ianzubov@iazubov:~$ cd lab13
ianzubov@iazubov:~/lab13$ touch 1.sh
ianzubov@iazubov:~/lab13$ touch text
ianzubov@iazubov:~/lab13$ chmod *x 1.sh
chmod: неверный режим: «*x»
По команде «chmod --help» можно получить дополнительную информацию.
ianzubov@iazubov:~/lab13$ chmod +x 1.sh
ianzubov@iazubov:~/lab13$ ./1.sh
Ошибка: не указан шаблон для поиска (параметр -p)
ianzubov@iazubov:~/lab13$ ./1.sh -p "Солнце" -i text
Солнце
ianzubov@iazubov:~/lab13$ ./1.sh -p "Здесь" -i text
Здесь
Здесь нечего ловить нам
ianzubov@iazubov:~/lab13$ ./1.sh -p "Даже" -i text
Даже
ianzubov@iazubov:~/lab13$
```

Рис. 3.2: Проверяем работу кода

Создаем файл 2.c и 2.sh. Напишем на языке Си программу, которая вводит число

и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции exit(n), передавая информацию в о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено.

```c
#include <stdio.h>
#include <stdlib.h>

int main() {
    int number;

    printf("Введите число: ");
    scanf("%d", &number);

    if (number > 0) {
        printf("Число больше нуля\n");
        exit(1);
    } else if (number < 0) {
        printf("Число меньше нуля\n");
        exit(2);
    } else {
        printf("Число равно нулю\n");
        exit(0);
    }

    return 0;
}
```

Рис. 3.3: Пишем первый код

```bash
#!/bin/bash
./2
case $? in
    0)
        echo "Число равно нулю";;
    1)
        echo "Число больше нуля";;
    2)
        echo "Число меньше нуля";;
esac
```

Рис. 3.4: Пишем второй код

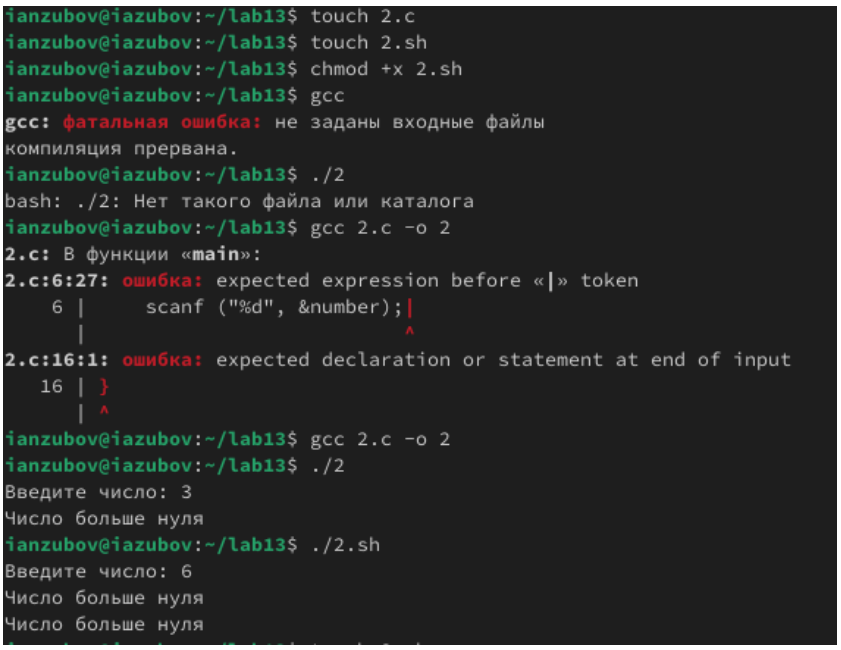

Рис. 3.5: Проверяем работу кода

Создаем файл 3.sh. Напишем командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до (например 1.tmp, 2.tmp, 3.tmp,4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).

Открыть • 3.sh ~/lab13

nd | report.md | presentation.md | 1.sh | text | 2.c

```
#!/bin/bash

# Функция создания файлов
create_files() {
    local count=$1
    for ((i=1; i<=count; i++)); do
        touch "${i}.tmp"
    done
    echo "Создано $count файлов: от 1.tmp до ${count}.tmp"
}

# Функция удаления файлов
delete_files() {
    local count=$1
    local deleted=0
    for ((i=1; i<=count; i++)); do
        if [ -f "${i}.tmp" ]; then
            rm -f "${i}.tmp"
            ((deleted++))
        fi
    done
    echo "Удалено $deleted файлов из возможных $count"
}

# Основной код скрипта
if [ $# -ne 2 ]; then
    echo "Использование: $0 [create|delete] N"
    echo "  create N - создать N файлов"
    echo "  delete N - удалить N файлов"
    exit 1
fi

action=$1
count=$2

# Проверка что N - положительное число
if ! [[ "$count" =~ ^[1-9][0-9]*$ ]]; then
    echo "Ошибка: N должно быть положительным целым числом"
    exit 1
fi
```

Рис. 3.6: Пишем код

```
ianzubov@iazubov:~/lab13$ touch 3.sh
ianzubov@iazubov:~/lab13$ chmod +x 3.sh
ianzubov@iazubov:~/lab13$ ./3.sh 5
Использование: ./3.sh [create|delete] N
  create N - создать N файлов
  delete N - удалить N файлов
ianzubov@iazubov:~/lab13$ ./3.sh create 5
Создано 5 файлов: от 1.tmp до 5.tmp
ianzubov@iazubov:~/lab13$ ./3.sh delete 5
Удалено 5 файлов из возможных 5
ianzubov@iazubov:~/lab13$
```

Рис. 3.7: Проверяем работу кода

Создаем файл 4.sh. Напишем командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировать его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовать команду find).

```bash
#!/bin/bash

directory=$1
output_archive="archive.tar.gz"  # Исправлено: добавлен знак = и исправлено имя переменной
threshold_days=7

if [ -z "$directory" ]; then
    echo "Укажите директорию в качестве аргумента"
    exit 1  # Исправлено: убрано "-" перед 1
fi

if [ ! -d "$directory" ]; then  # Исправлено: добавлено "!" для проверки отрицания
    echo "Указанная директория не существует"
    exit 1
fi

# Используем find для поиска файлов, измененных менее чем threshold_days назад,
# и передаем их в tar
find "$directory" -type f -mtime -$threshold_days -print0 | tar --null -czf "$output_archive" --files-from -
echo "Архивация завершена. Архив создан: $output_archive"
```

Рис. 3.8: Пишем код

```
Удалено 5 файлов из возможных 5
ianzubov@iazubov:~/lab13$ touch 4.sh
ianzubov@iazubov:~/lab13$ chmod +x 4.sh
ianzubov@iazubov:~/lab13$ ./4.sh .
Архивация завершена. Архив создан: archive.tar.gz
ianzubov@iazubov:~/lab13$
```

Рис. 3.9: Проверяем работу кода

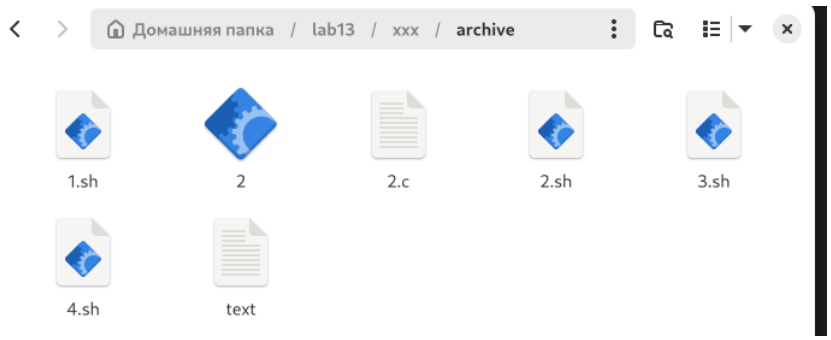

Рис. 3.10: Смотрим как архивировались файлы

4 Выводы

Я научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.