Отчет по лабораторной работе №11.

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Данила Андреевич Стариков

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

## 2.1 Задание 1.

Формулировка задания: Используя команды getopts grep, написать командный файл, который анализирует командную строку с ключами:

* -i — прочитать данные из указанного файла;
* -o — вывести данные в указанный файл;
* -p — 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.

Создан командный файл 1.sh (Листинг 1). Примеры выполнения скрипта представлены на Рисунке 1.

Листинг 1: Текст командного файла Задания №1.

#!/bin/bash  
while getopts i:o:p:Cn optletter  
do case $optletter in  
i) iflag=1; ival=$OPTARG;;  
o) oflag=1; oval=$OPTARG;;  
p) pflag=1; pval=$OPTARG;;  
C) Cflag=1;;  
n) nflag=1;;  
\*) echo Illegal option $optletter  
esac  
done  
OPTIONS+="-e $pval $ival"   
if [ ! $Cflag ]  
then OPTIONS+=" -i"  
fi  
if [ $nflag ]  
then OPTIONS+=" -n"  
fi  
grep ${OPTIONS} > $oval

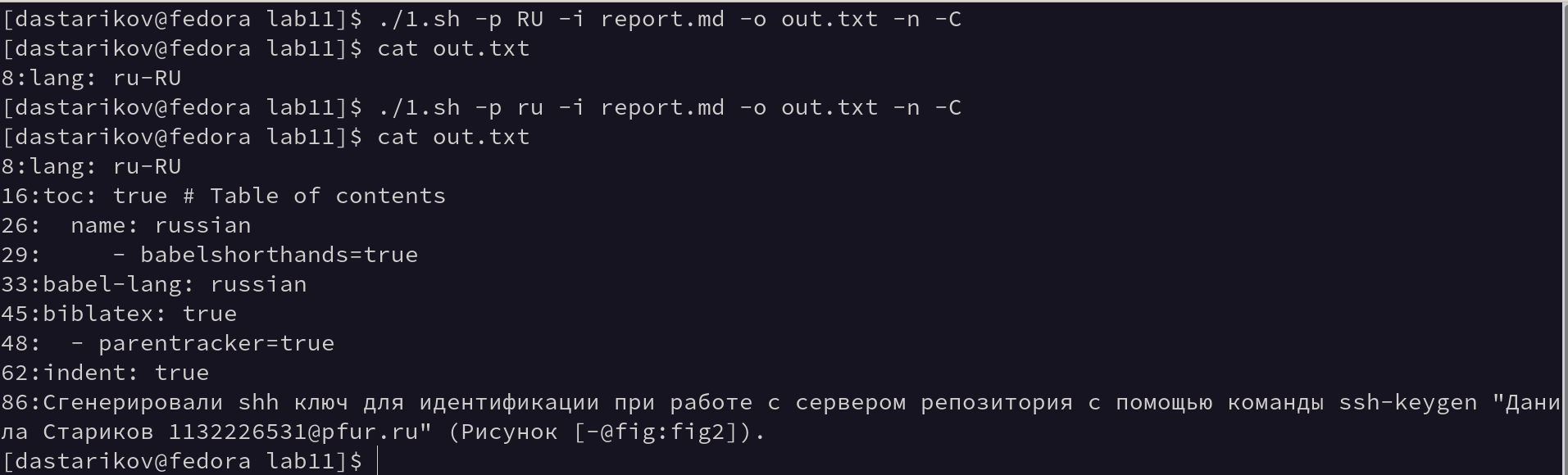


Рис. 1: Пример работы программы 1.sh.

## 2.2 Задание 2.

Формулировка задания: 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.

По заданию написаны 2 программы: mod.c на языке Си (Листинг 2) и командный файл 2.sh (Листинг 3). Примеры выполнения скрипта представлены на Рисунке 2.

Листинг 2: Текст файла mod.c Задания №1.

#include<stdio.h>  
#include<stdlib.h>  
  
int main(){  
 int n;  
 printf("Введите число: ");  
 scanf("%d", &n);  
 if(n > 0) {n = 1;}  
 else if (n<0) {n = -1;}  
 else {n=0;}  
 exit(n);  
 return 0;  
}

Листинг 3: Текст командного файла Задания №2.

#!/bin/bash  
./mod  
signal=$?  
case $signal in  
'1') echo число больше нуля;;  
'255') echo число меньше нуля;;  
'0') echo число равно нулю;  
esac

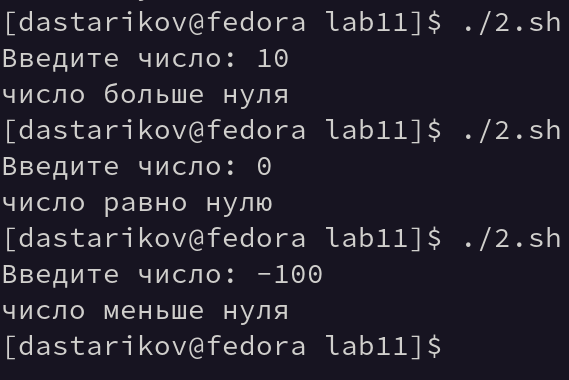


Рис. 2: Пример работы программы 2.sh.

## 2.3 Задание 3.

Формулировка задания: Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (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).

Создан командный файл 1.sh (Листинг 4). Примеры выполнения скрипта представлены на Рисунке 3.

Листинг 4: Текст командного файла Задания №3.

#!/bin/bash  
N=$1  
FILES=  
EXTENSION=.tmp  
DIR=temp  
for (( a=1; a <= N; a++ ))  
do  
 FILES+=$DIR/${a}${EXTENSION}" "  
done  
  
if (! find $DIR -type d 2>/dev/null)  
then   
 mkdir $DIR  
fi  
  
touch $FILES

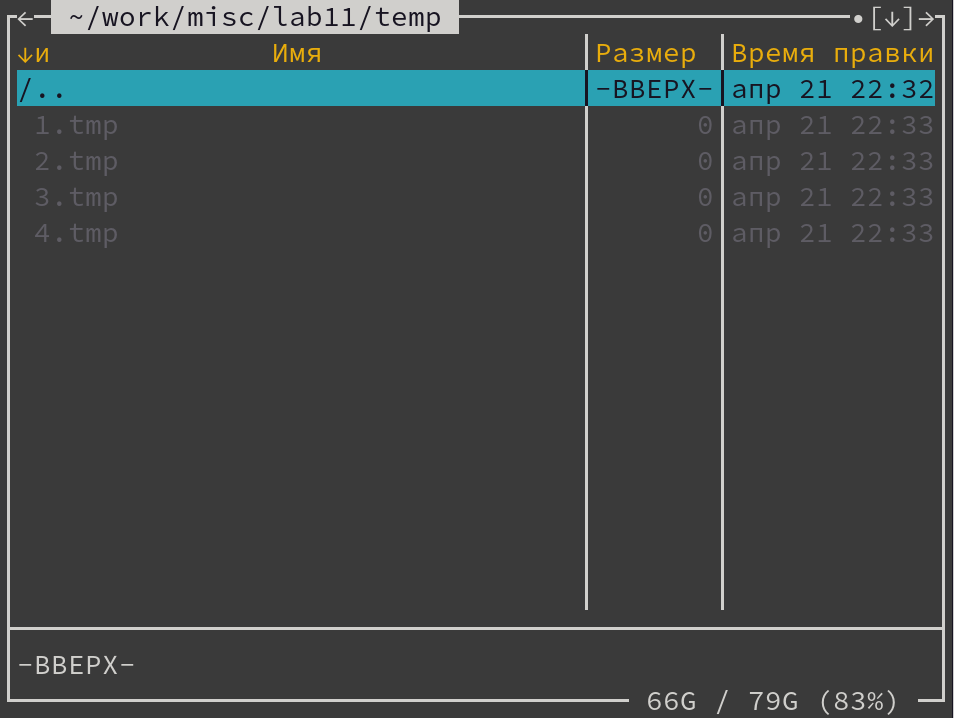


Рис. 3: Пример работы программы 3.sh.

## 2.4 Задание 4.

Формулировка задания: 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).

Создан командный файл 4.sh (Листинг 5). Примеры выполнения скрипта представлены на Рисунке 4.

Листинг 5: Текст командного файла Задания №4.

#!/bin/bash  
tar -c -f archive.tar $(find -type f -mtime -7)

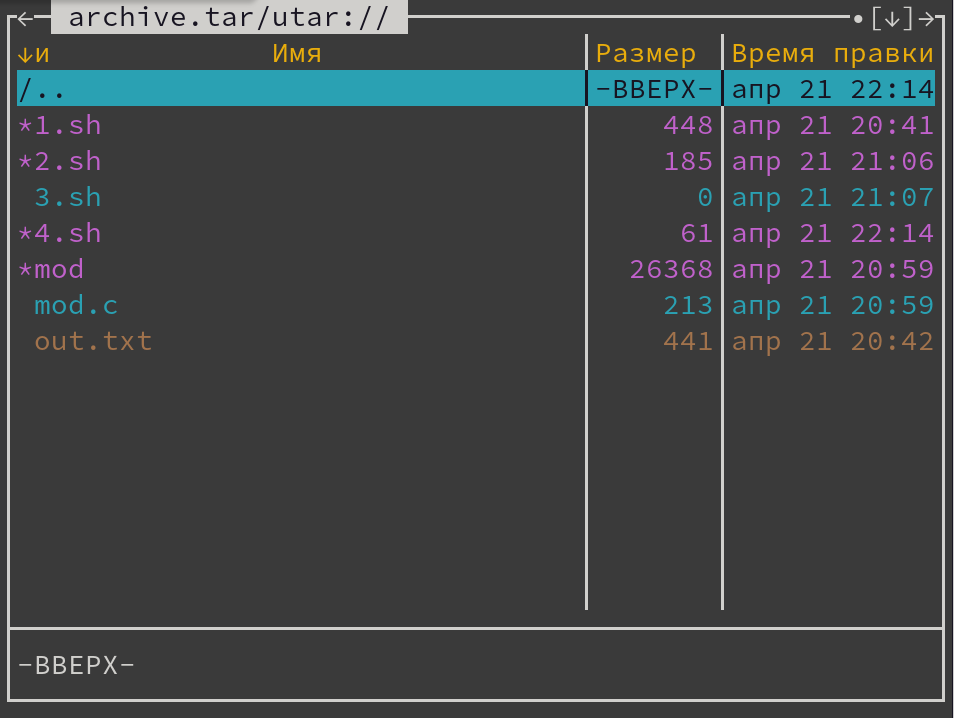


Рис. 4: Пример работы программы 4.sh.

# 3 Выводы

В рамках лабораторной работы начились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.