

# **Отчёт по лабораторной работе №13**

**Операционные системы**

Андреева Софья Владимировна

# Содержание

1	Цель работы	4
2	Выполнение работы	5
3	Контрольные вопросы	11
4	Выводы	14

## Список иллюстраций

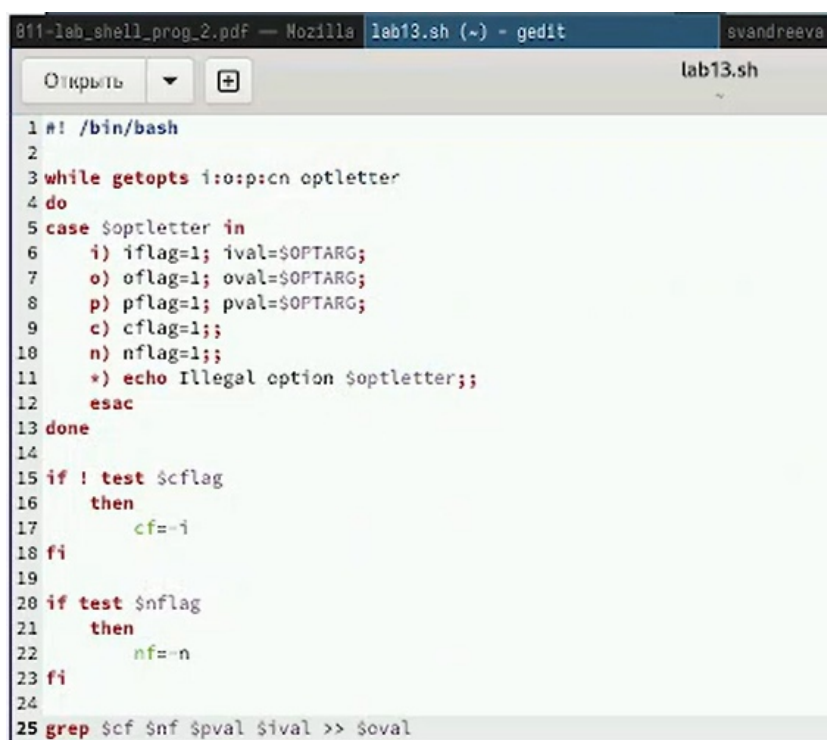
2.1	Скрипт . . . . .	5
2.2	Работа кода . . . . .	6
2.3	Скрипт кода С . . . . .	7
2.4	Скрипт . . . . .	7
2.5	Работа кода . . . . .	8
2.6	Скрипт . . . . .	8
2.7	Работа кода . . . . .	9
2.8	Скрипт . . . . .	10

# 1 Цель работы

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

## 2 Выполнение работы

Используя команды `getopts` `grep`, написала командный файл, который анализирует командную строку с ключами: `-i inputfile`; `-o outputfile`; `-p`; `-C`; `-n`, а затем ищет в указанном файле нужные строки, определяемые ключом `-p` (рис. fig. 2.1). Результат работы при заданной строке “От” (рис. fig. 2.2).



```
1 #!/bin/bash
2
3 while getopts i:o:p:C:n optletter
4 do
5 case $optletter in
6   i) iflag=1; ival=$OPTARG;
7   o) oflag=1; oval=$OPTARG;
8   p) pflag=1; pval=$OPTARG;
9   c) cflag=1;;
10  n) nflag=1;;
11  *) echo Illegal option $optletter;;
12  esac
13 done
14
15 if ! test $cflag
16 then
17   cf=-i
18 fi
19
20 if test $nflag
21 then
22   nf=-n
23 fi
24
25 grep $cf $nf $pval $ival >> $oval
```

Рис. 2.1: Скрипт

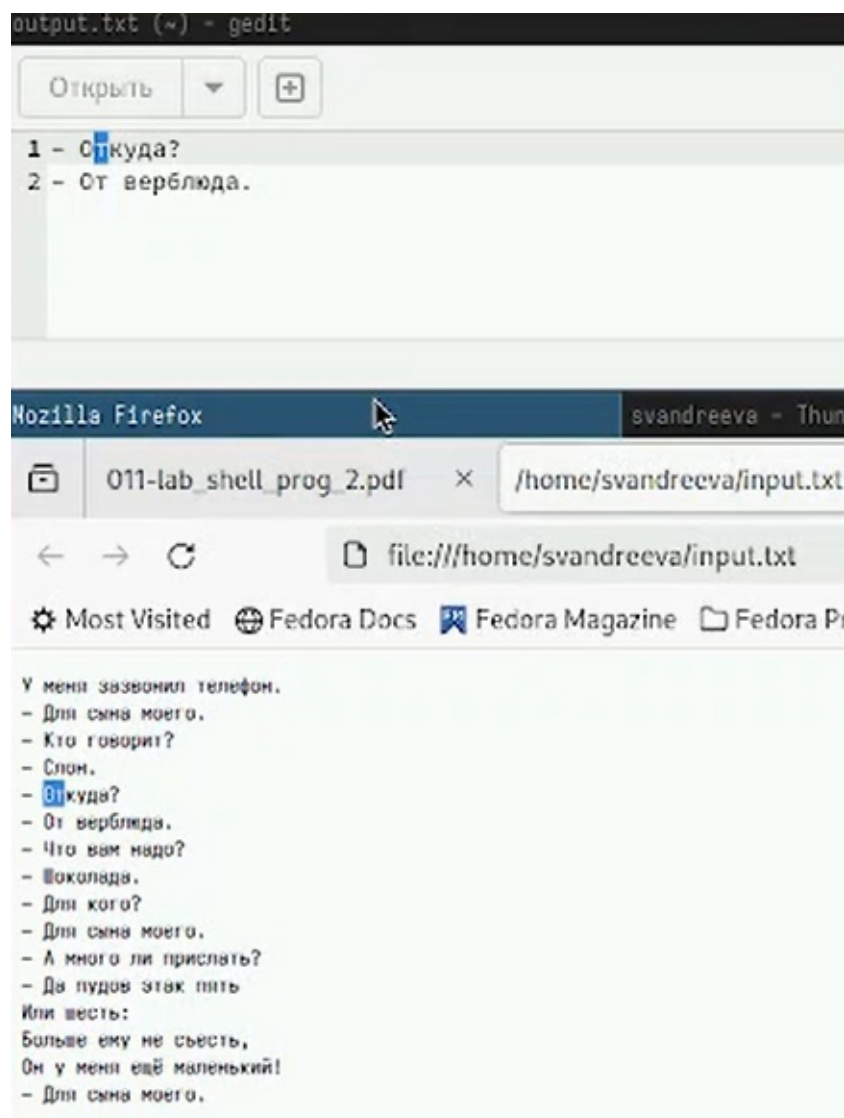


Рис. 2.2: Работа кода

Написала на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю (рис. fig. 2.3). 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 (рис. fig. 2.4), (рис. fig. 2.5).

```
1 #include <stdlib.h>
2 #include <stdio.h>
3
4 int main(){
5     int n;
6     printf('Enter the number');
7     scanf("%d",&n);
8     if (n>0){
9         exit(1);}
10    else if (n==0){
11        exit(0);}
12    else{
13        exit(2);}
14
15
```

Рис. 2.3: Скрипт кода С

```
1 #! /bin/bash
2
3 gcc -o cprog lab13.c
4 ./cprog
5 case $? in
6 0) echo "The number = 0";;
7 1) echo "The number > 0";;
8 2) echo "The number < 0";;
9 esac
```

Рис. 2.4: Скрипт

```

[svandreeva@fedora ~]$ bash lab13.sh
Enter the number13
The number > 0
[svandreeva@fedora ~]$ bash lab13.sh
Enter the number0
The number = 0
[svandreeva@fedora ~]$ bash lab13.sh
Enter the number-9
The number < 0

```

Рис. 2.5: Работа кода

Написала командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N ( 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) (рис. fig. 2.6) (рис. fig. 2.7).



```

011-lab_shell_prog_2.pdf — Mozilla | svandreeva -
Открыть ▼ +
1 #! /bin/bash
2
3 for((i=1; i<${1}; i++))
4 do
5 if test -f "$i".tmp
6 then rm "$i".tmp
7 else touch "$i".tmp
8 fi
9 done

```

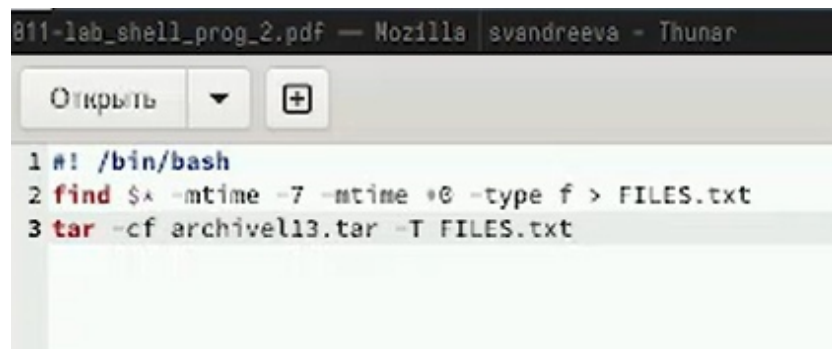
Рис. 2.6: Скрипт



```
[svandreeva@fedora ~]$ bash lab13_3.sh 3
[svandreeva@fedora ~]$ ls
1.tnp    bin          file         lab13_3.sh~
2.tnp    conf.txt     git-extended lab13.c
3.tnp    cprog        lab07.sh~    lab13.c~
backup  Downloads    lab13_3.sh   lab13.sh
[svandreeva@fedora ~]$ bash lab13_3.sh 4
[svandreeva@fedora ~]$ ls
4.tnp    cprog        lab07.sh~    lab13.c~
backup  Downloads    lab13_3.sh   lab13.sh
bin      file         lab13_3.sh~   lab13.sh~
conf.txt git-extended lab13.c        LICENSE
[svandreeva@fedora ~]$ bash lab13_3.sh 4
[svandreeva@fedora ~]$ ls
1.tnp    bin          file         lab13_3.sh~
2.tnp    conf.txt     git-extended lab13.c
3.tnp    cprog        lab07.sh~    lab13.c~
backup  Downloads    lab13_3.sh   lab13.sh
[svandreeva@fedora ~]$ bash lab13_3.sh 3
[svandreeva@fedora ~]$ ls
backup  Downloads    lab13_3.sh   lab13.sh
bin      file         lab13_3.sh~   lab13.sh~
conf.txt git-extended lab13.c        LICENSE
cprog    lab07.sh~    lab13.c~      main.cpp
```

Рис. 2.7: Работа кода

Написала командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировала его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад. (рис. fig. 2.8)



The image shows a screenshot of a Thunar file manager window. The title bar at the top reads "811-lab\_shell\_prog\_2.pdf — Mozilla | svandreeva - Thunar". Below the title bar is a toolbar with a button labeled "Открыть" (Open), a dropdown arrow, and a plus icon. The main area of the window displays a shell script with three lines of code, each preceded by a line number:

```
1 #! /bin/bash
2 find $* -mtime -7 -mtime *0 -type f > FILES.txt
3 tar -cf archivell3.tar -T FILES.txt
```

Рис. 2.8: Скрипт

### 3 Контрольные вопросы

- Каково предназначение команды `getopts`? Осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg ... ]` Флаги – это опции командной строки, обычно помеченные знаком минус; Например, `-F` является флагом для команды `ls -F`. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций `option-string` — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`.
- Какое отношение метасимволы имеют к генерации имён файлов? При перечислении имён файлов текущего каталога можно использовать следующие символы: `-` соответствует произвольной, в том числе и пустой строке; `?` – соответствует любому одинарному символу; `[c1-c2]` – соответствует любому символу, лексикографически находящемуся между символами `c1` и `c2`. Например, `echo *` – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`; `ls .c` – выведет все файлы с последними двумя символами, совпадающими с `.c`. `echo prog.?` – вы-

ведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются prog.. [a-z] – соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.

- Какие операторы управления действиями вы знаете? Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости от результатов проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет возможность использовать такие управляющие конструкции, как `for`, `case`, `if` и `while`. С точки зрения командного процессора эти управляющие конструкции являются обычными командами и могут использоваться как при создании командных файлов, так и при работе в интерактивном режиме. Команды, реализующие подобные конструкции, по сути, являются операторами языка программирования `bash`. Поэтому при описании языка программирования `bash` термин оператор будет использоваться наравне с термином команда. Команды ОС UNIX возвращают код завершения, значение которого может быть использовано для принятия решения о дальнейших действиях. Команда `test`, например, создана специально для использования в командных файлах. Единственная функция этой команды заключается в выработке кода завершения.
- Какие операторы используются для прерывания цикла? Два несложных способа позволяют вам прерывать циклы в оболочке `bash`. Команда `break` завершает выполнение цикла, а команда `continue` завершает данную итерацию блока операторов. Команда `break` полезна для завершения цикла `while` в ситуациях, когда условие перестаёт быть правильным. Команда `continue` используется в ситуациях, когда больше нет необходимости выполнять блок операторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях.

- Для чего нужны команды `false` и `true`? Следующие две команды ОС UNIX используются только совместно с управляющими конструкциями языка программирования `bash`: это команда `true`, которая всегда возвращает код завершения, равный нулю (т.е. истина), и команда `false`, которая всегда возвращает код завершения, не равный нулю (т. е. ложь).
- Что означает строка `if test -f man/i.$s`, встреченная в командном файле? Строка `if test -f mans/i.s, mans/i.s` и является ли этот файл обычным файлом. Если данный файл является каталогом, то команда вернет нулевое значение (ложь).
- Объясните различия между конструкциями `while` и `until`. Выполнение оператора цикла `while` сводится к тому, что сначала выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, а затем, если последняя выполненная команда из этой последовательности команд возвращает нулевой код завершения (истина), выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `do`, после чего осуществляется безусловный переход на начало оператора цикла `while`. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда из последовательности команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, возвратит ненулевой код завершения (ложь). При замене в операторе цикла `while` служебного слова `while` на `until` условие, при выполнении которого осуществляется выход из цикла, меняется на противоположное. В остальном оператор цикла `while` и оператор цикла `until` идентичны

## 4 Выводы

В процессе выполнения данной лабораторной работы я изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.