

Отчёт по лабораторной работе №11

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

Федюшина Ярослава Андреевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	8
4	Выводы	11
5	Контрольные вопросы	12

Список иллюстраций

3.1	создание файла	8
3.2	написание кода	9
3.3	проверка кода	10

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:
 - `-i`inputfile — прочитать данные из указанного файла;
 - `-o`outputfile — вывести данные в указанный файл;
 - `-r`шаблон — указать шаблон для поиска;
 - `-C` — различать большие и малые буквы;
 - `-n` — выдавать номера строк, а затем ищет в указанном файле нужные строки, определяемые ключом `-r`.
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 код 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 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).
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

Используя команды `getopts` `grep`, написала командный файл, который анализирует командную строку с ключами: `-i`inputfile — прочитать данные из указанного файла; `-o`outputfile — вывести данные в указанный файл; `-r`шаблон — указать шаблон для поиска; `-C`—различать большие и малые буквы; `-n`—выдавать номера строк.

3.1

Для данной работы создала файл `11lab.sh`

3.2

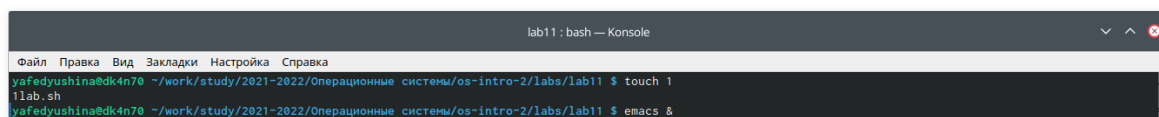
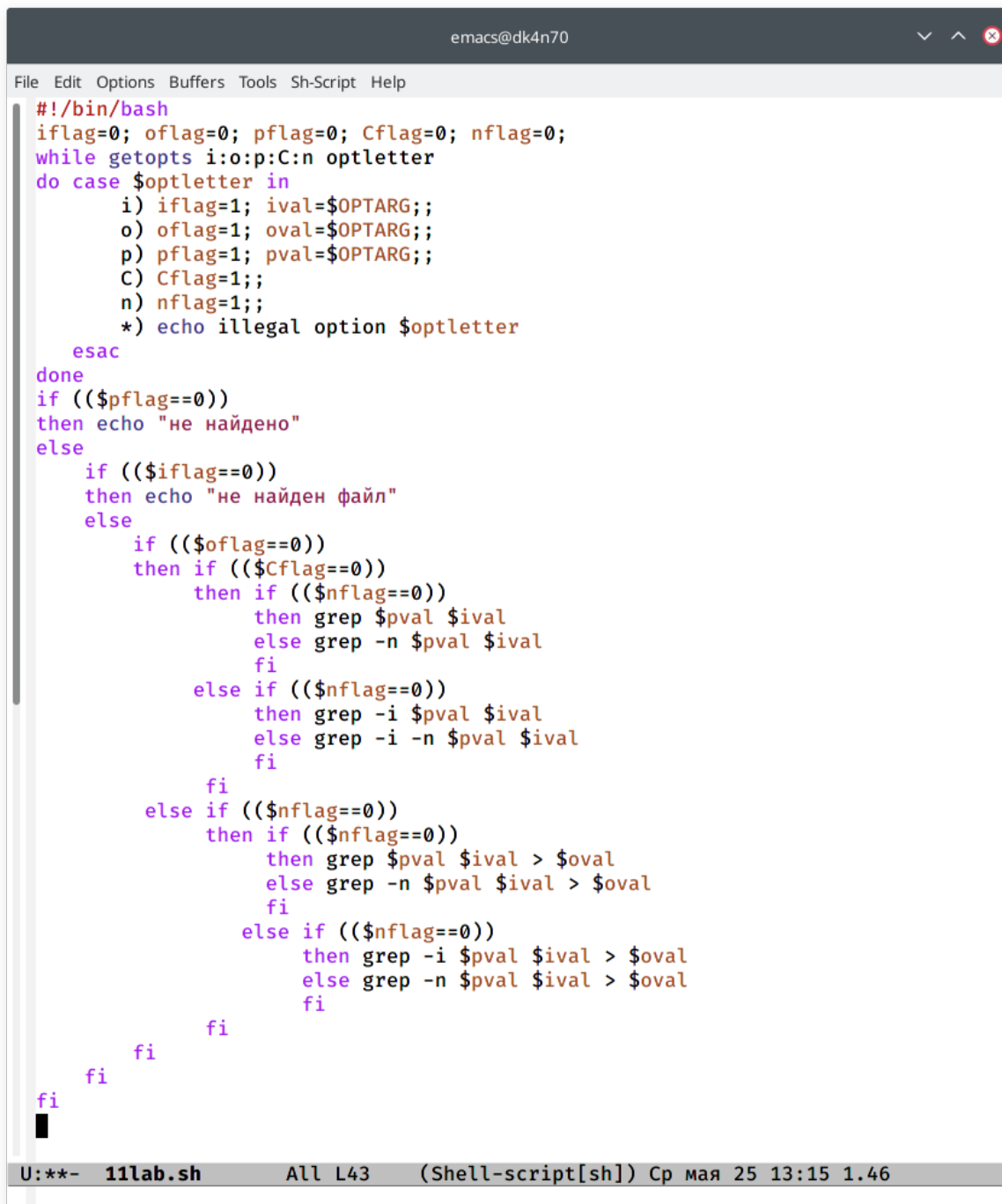


Рис. 3.1: создание файла

3.3

написала нужный код

3.4



```
emacs@dk4n70
File Edit Options Buffers Tools Sh-Script Help
#!/bin/bash
iflag=0; oflag=0; pflag=0; Cflag=0; nflag=0;
while getopts i:o:p:C:n 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 $optletter
  esac
done
if (($pflag==0))
then echo "не найдено"
else
  if (($iflag==0))
  then echo "не найден файл"
  else
    if (($oflag==0))
    then if (($Cflag==0))
        then if (($nflag==0))
            then grep $pval $ival
            else grep -n $pval $ival
            fi
        else if (($nflag==0))
            then grep -i $pval $ival
            else grep -i -n $pval $ival
            fi
        fi
    else if (($nflag==0))
        then if (($nflag==0))
            then grep $pval $ival > $oval
            else grep -n $pval $ival > $oval
            fi
        else if (($nflag==0))
            then grep -i $pval $ival > $oval
            else grep -n $pval $ival > $oval
            fi
        fi
    fi
  fi
fi
fi
```

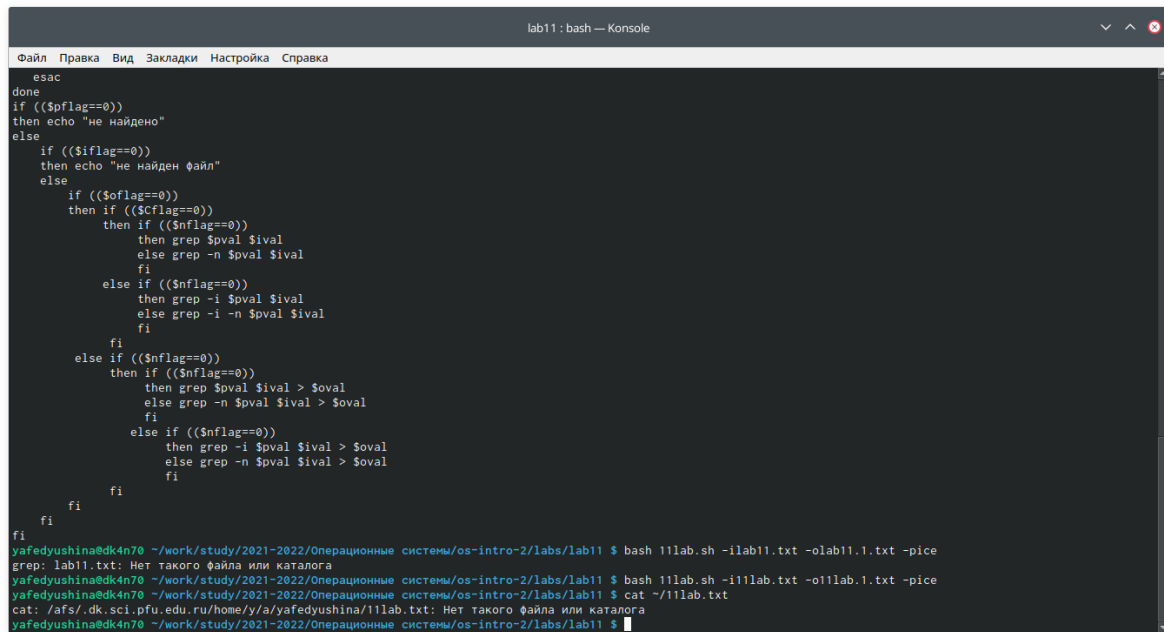
U:***- 11lab.sh All L43 (Shell-script[sh]) Ср мая 25 13:15 1.46

Рис. 3.2: написание кода

3.5

Проверяю работу данного кода, используя разные опции, предварительно добавив право на исполнение файла

3.6



```
lab11: bash — Konsole
Файл  Правка  Вид  Закладки  Настройка  Справка
esac
done
if (($flag==0))
then echo "не найдено"
else
  if (($iflag==0))
  then echo "не найден файл"
  else
    if (($oflag==0))
    then if (($cflag==0))
        then if (($nflag==0))
            then grep $pval $ival
            else grep -n $pval $ival
            fi
        else if (($nflag==0))
            then grep -i $pval $ival
            else grep -i -n $pval $ival
            fi
        fi
    else if (($nflag==0))
        then if (($nflag==0))
            then grep $pval $ival > $oval
            else grep -n $pval $ival > $oval
            fi
        else if (($nflag==0))
            then grep -i $pval $ival > $oval
            else grep -n $pval $ival > $oval
            fi
        fi
    fi
  fi
fi
fi
yafedyushina@dk4n70 ~/work/study/2021-2022/Операционные системы/os-intro-2/labs/lab11 $ bash t1lab.sh -ilab11.txt -olab11.1.txt -pice
grep: lab11.txt: Нет такого файла или каталога
yafedyushina@dk4n70 ~/work/study/2021-2022/Операционные системы/os-intro-2/labs/lab11 $ bash t1lab.sh -i11lab.txt -o11lab.1.txt -pice
yafedyushina@dk4n70 ~/work/study/2021-2022/Операционные системы/os-intro-2/labs/lab11 $ cat ~/11lab.txt
cat: /afs/.dk.sci.pfu.edu.ru/home/y/a/yafedyushina/11lab.txt: Нет такого файла или каталога
yafedyushina@dk4n70 ~/work/study/2021-2022/Операционные системы/os-intro-2/labs/lab11 $
```

Рис. 3.3: проверка кода

3.7

4 Выводы

В ходе выполнения лабораторной работы №11 я изучила основы программирования в оболочке ОС Linux, а так же научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов

5 Контрольные вопросы

1. Команда `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, для команды `ls` флагом может являться `-F`. Строка опций `option-string` – это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за символом, обозначающим этот флаг, должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, то она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента. Функция `getopts` также понимает переменные типа массив, следовательно, можно использовать её в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.
2. 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.? – выведет 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.
3. 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.
4. 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.

5. 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). Примеры бесконечных циклов: `while true do echo hello andy done` и `until false do echo hello mike done`
6. 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).
7. 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.