# Лабораторная работа №13

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

Буллер Татьяна Александровна

# Содержание

# Список иллюстраций

# 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 лабораторной работы

## 2.1 Обработка ключей

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

Принимать некоторое значение должны 3 ключа: -i, -o, -p. Чтобы указать на это, после их обозначения в команде getopts ставим двоеточие. Далее рассмотрим каждый случай через конструкцию case: – если получили ключ -i - в переменную fin введем значение аргумента после ключа (файл анализа). – если получили ключ -o - в переменную fout введем значение аргумента после ключа (файл вывода). – если получили ключ -p - в переменную reg введем значение аргумента после ключа (регулярное выражение для поиска grep).

По умолчанию grep различает большие и маленькие буквы. Для того, чтобы он этого не делал, используем опцию -i. Так как опция -C должна задавать программе различать регистры, то ставим, что по умолчанию она их НЕ различает, а при получении опции обнуляем переменную. – -n — опция, выдающая номера строк. Точно так же она используется и в grep, поэтому при получении этого ключа просто сохраняем его в переменную и в дальнейшем вставляем в команду.

В случае, если мы получили файл вывода, то нужно перенаправить результат

выполнения туда. В противном случае переменная fout останется пустой, вывод будет произведен в консоль.

```
#!/bin/bash
c="-i"
while getopts 'o:i:p:Cn' OPTION; do
  case "$OPTION" in
    o)
      fout="$OPTARG" ;;

    i)
      fin="$OPTARG" ;;

    p)
      reg="$OPTARG" ;;

    C) c="" ;;

    n) n="-n";;

    ?)
      echo "unknown option $OPTIND"
      exit 1
      ;;
  esac
done
if [ -v fout ]
then grep $n $c $reg $fin > $fout
else grep $n $c $reg $fin
fi

```

Рис. 2.1: Скрипт обработки ключей

После исполнения скрипта проверим с его помощью текстовый файл одной из предыдущих лабораторных работ: видим, что создался файл вывода с указанным названием и вывод в нем соответствует заданному регулярному выражению.

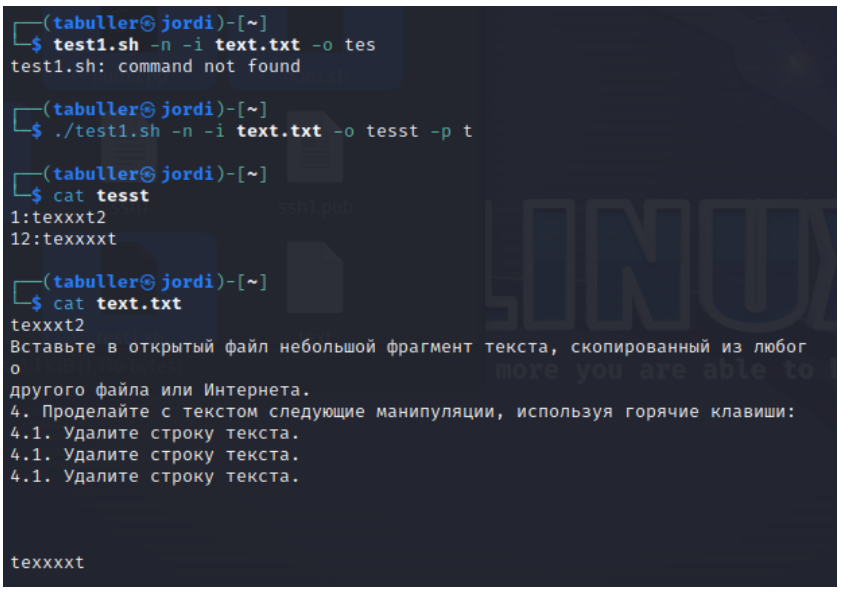
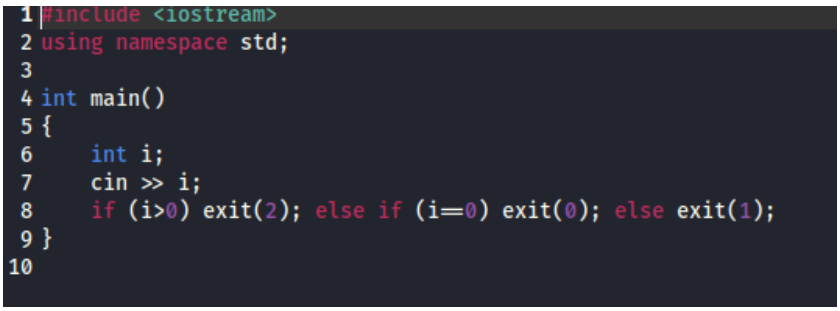



Рис. 2.2: Результат выполнения скрипта

## 2.2 Обработка кодов завершения

Требуется написать на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Командный файл должен вызывать эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено.

Для выполнения задания напишем простую программу на языке Си: если число больше 0 - код завершения 2, равно 0 - код завершения 0, меньше - код завершения 1. На вход принимаем целое число. Необходимо учитывать, что в случае, если в аргумент попадет строка, она будет либо приравняна к 0, либо в аргумент пойдут самые первые численные символы.



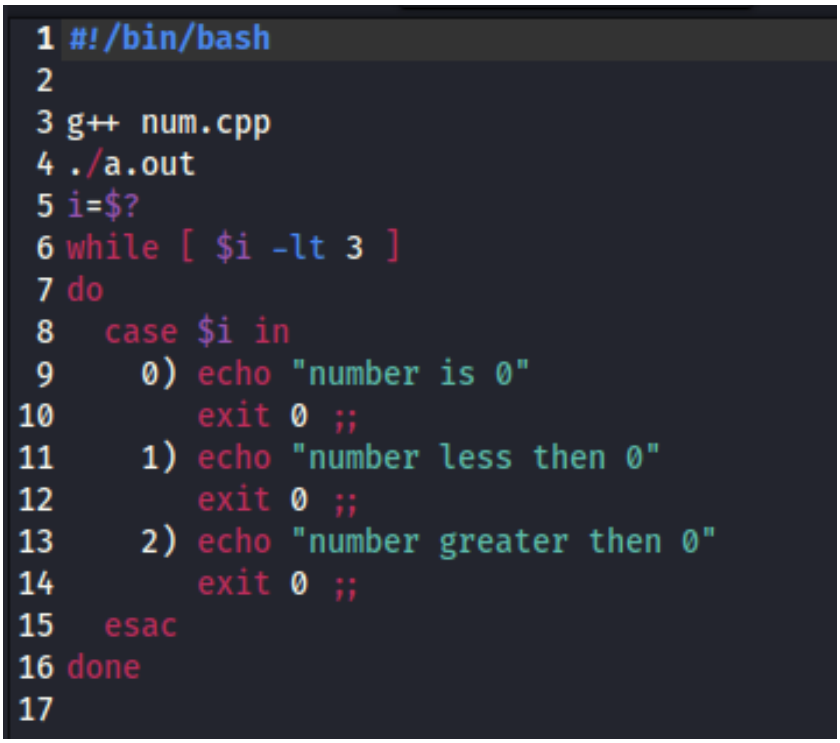
```cpp
#include <iostream>
using namespace std;

int main()
{
    int i;
    cin >> i;
    if (i>0) exit(2); else if (i==0) exit(0); else exit(1);
}

```

Рис. 2.3: Программа определения числа

Далее работаем с командой $?, определяющей код завершения программы. В самом скрипте вызовем программу, после чего приравняем переменную i к значению кода завершения и переберем варианты:



```bash
#!/bin/bash

g++ num.cpp
./a.out
i=$?
while [ $i -lt 3 ]
do
  case $i in
    0) echo "number is 0"
       exit 0 ;;
    1) echo "number less then 0"
       exit 0 ;;
    2) echo "number greater then 0"
       exit 0 ;;
  esac
done

```

Рис. 2.4: Скрипт определения числа

Для проверки работы скрипта используем несколько вариантов аргумента: больше нуля, 0, меньше нуля и две строки: начинающуюся с числа и начинающуюся с буквы. В двух последних случаях видим, что скрипт выполняется для 0 и для 9 соответственно.

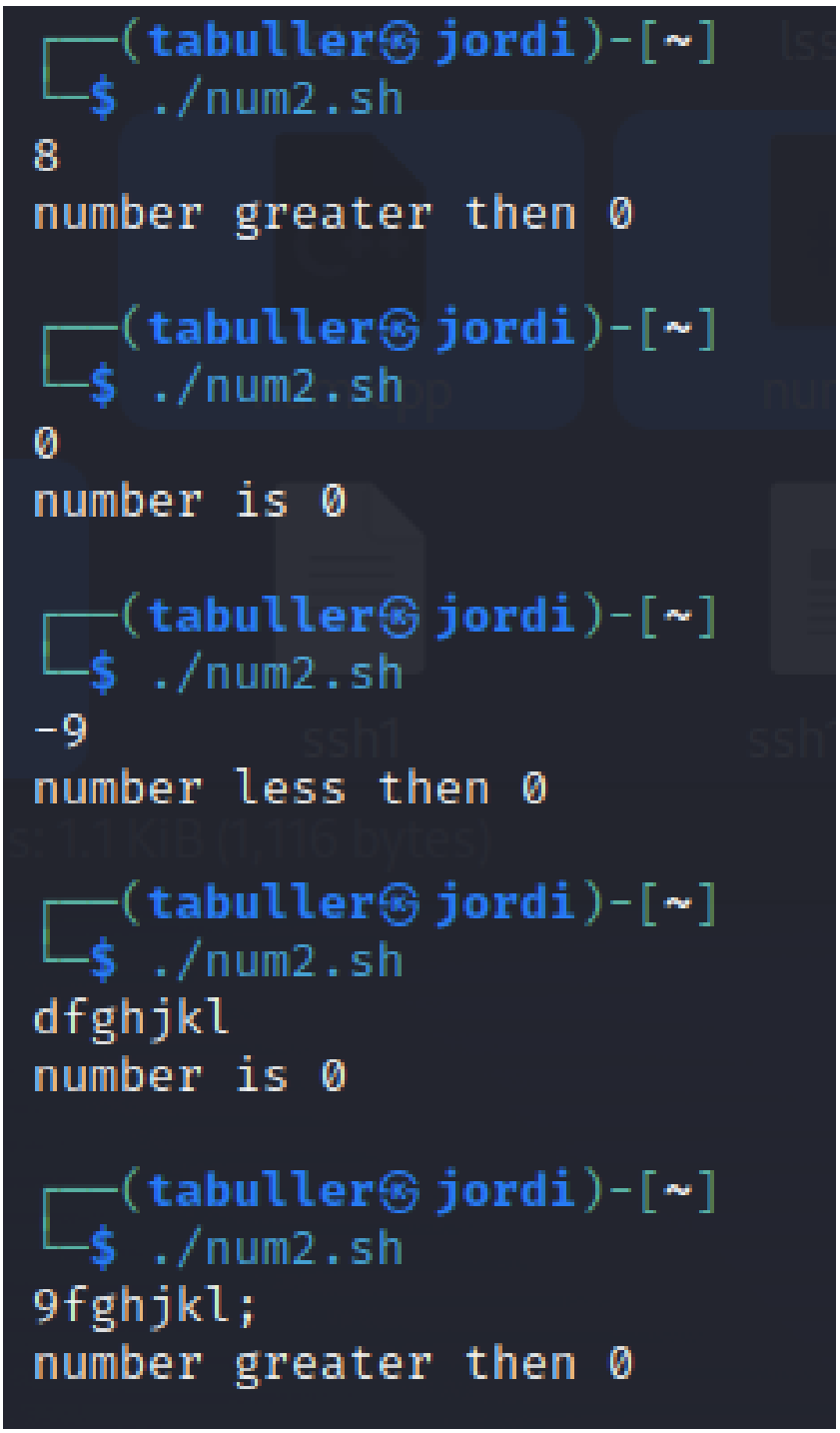
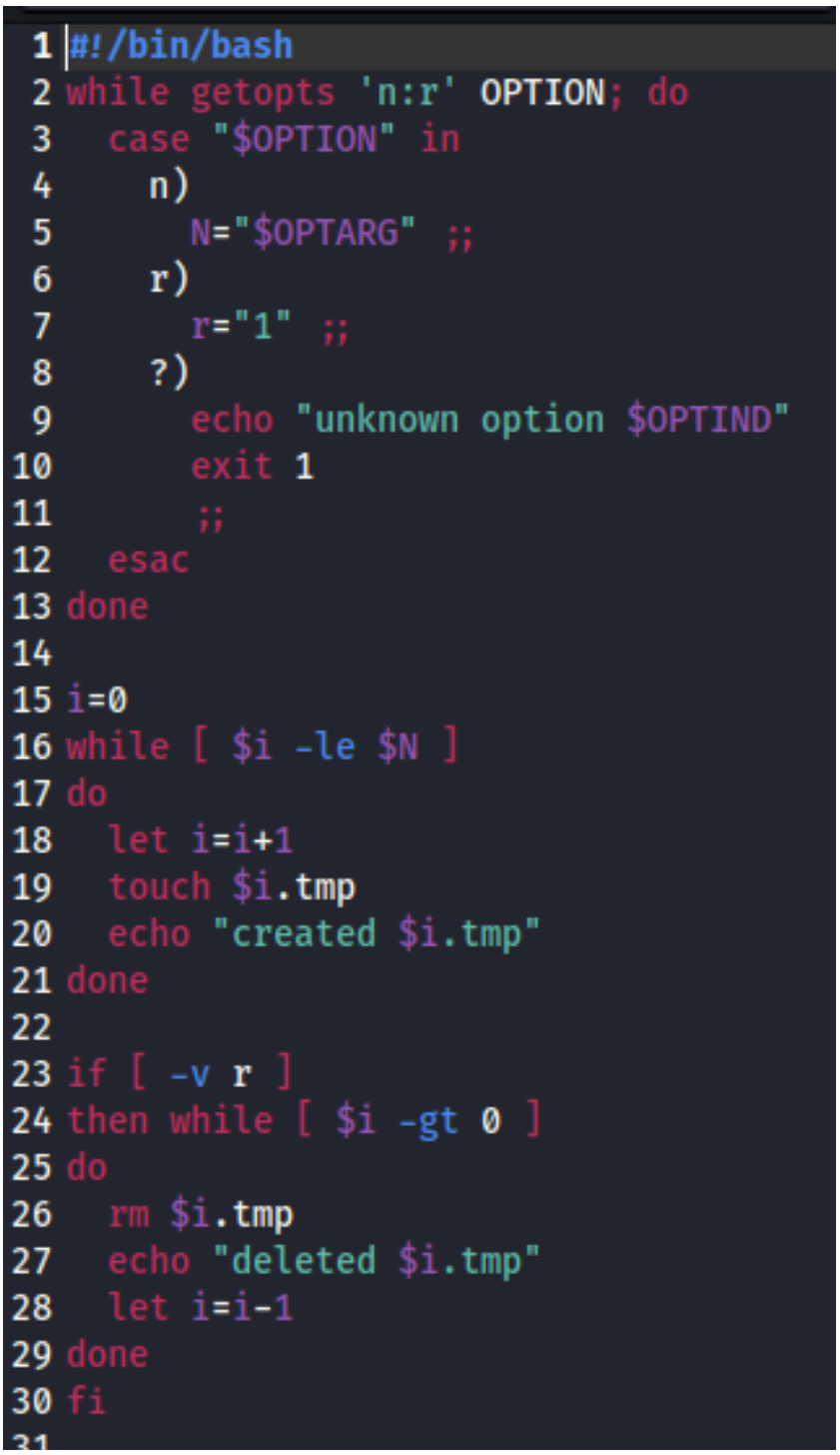



Рис. 2.5: Результат работы скрипта

## 2.3 Создание и удаление некоторого числа файлов.

Зададим команде два флага: -n и -r. При получении флага -n будем ожидать дальнейший аргумент в виде числа файлов к созданию, второй же флаг будет активировать команду на удаление созданных файлов. Механизм для обоих реализуем с помощью цикла while, но в первом случае создаем, а во втором, если флаг на удаление присутствует - удаляем файлы.



```
#!/bin/bash
while getopts 'n:r' OPTION; do
  case "$OPTION" in
    n)
      N="$OPTARG" ;;
    r)
      r="1" ;;
    ?)
      echo "unknown option $OPTIND"
      exit 1
      ;;
  esac
done

i=0
while [ $i -le $N ]
do
  let i=i+1
  touch $i.tmp
  echo "created $i.tmp"
done

if [ -v r ]
then while [ $i -gt 0 ]
do
  rm $i.tmp
  echo "deleted $i.tmp"
  let i=i-1
done
fi
```

Рис. 2.6: Скрипт создания нескольких файлов

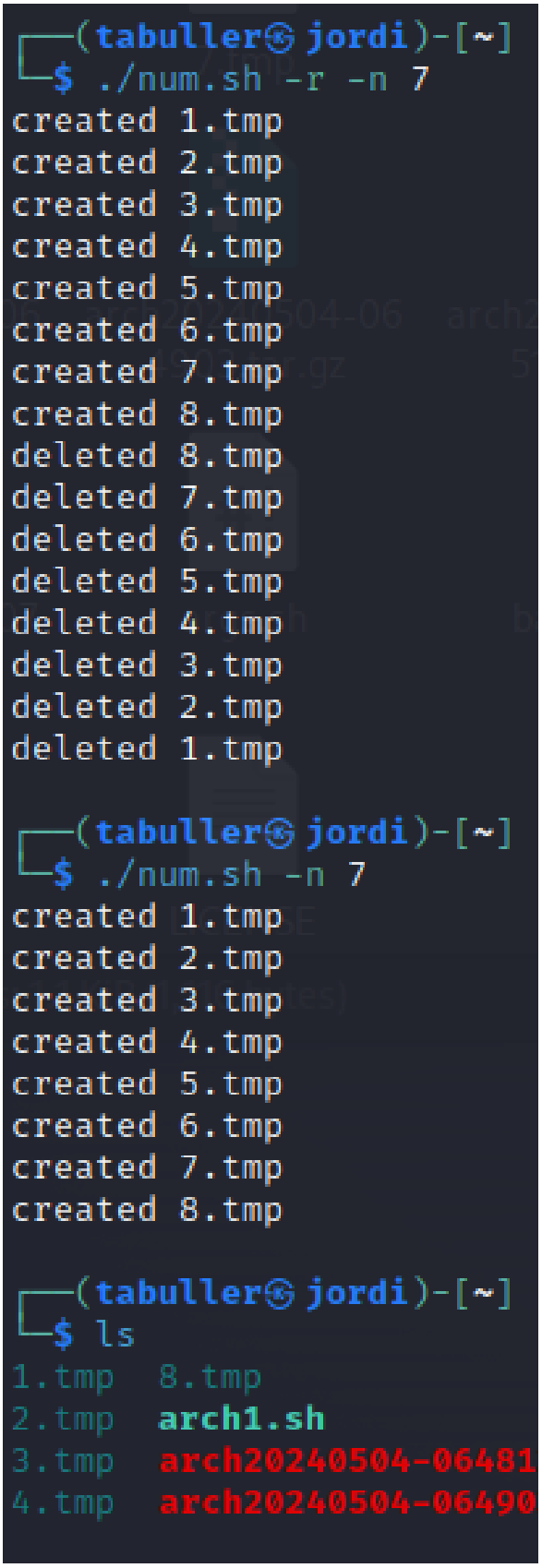



Рис. 2.7: Результат выполнения скрипта

## 2.4 Создание и удаление некоторого числа файлов.

Последний скрипт - архиватор. Для начала выполним команду find, которая найдет все файлы (-type f) в заданной директории (-maxdepth 1), созданные менее недели назад (-mtime -7). Результат выполнения сохраняем в текстовый файл, после чего создаем архив по списку с помощью флага -T.

```
1 #! /bin/bash
2 find $1 -maxdepth 1 -type f -mtime -7 > list.txt
3 tar -czvf arch$(date +%Y%m%d-%H%M%S).tar.gz -T list.txt
4 exit 0
5
```

Рис. 2.8: Скрипт архиватора

В результате видим вывод файлов, запакованных в архив, и получаем сам архив соответственно:

```
┌──(tabuller㉿jordi)-[~]
└─$ ls
a.out      backup1.sh  Downloads  lss.sh    num.sh     ssh1.pub   text.txt
arch1.sh   count.sh    LICENSE    Music     Pictures   Templates  Videos
args.sh    Desktop     list.txt   num2.sh   Public     test1.sh   work
backup1    Documents   lss1.sh    num.cpp   ssh1       text

┌──(tabuller㉿jordi)-[~]
└─$ ./arch1.sh
./.Xauthority.gz
./num.cpp
./arch1.sh
./list.txt
./num2.sh
./test1.sh
./.xsession-errors.gz
./num.sh
./a.out
./.zsh_history
```

Рис. 2.9: Результат выполнения скрипта

# 3 Выводы

Изучены основы программирования в оболочке ОС UNIX/Linux. Написаны более сложные командные файлы с использованием логических управляющих конструкций и циклов.