

Санкт-Петербургский Национальный Исследовательский Университет  
Информационных Технологий, Механики и Оптики

Кафедра Систем Управления и Информатики

**Лабораторная работа №3**  
*Вариант №1*

Выполнили: Антипов В.А.  
Труфанова А.А.  
Черняев М.К.

Проверили: Глебова Е.С.  
Мусаев А.А.

Санкт-Петербург  
2018

## 1) Написание скриптов. Операторы ветвления

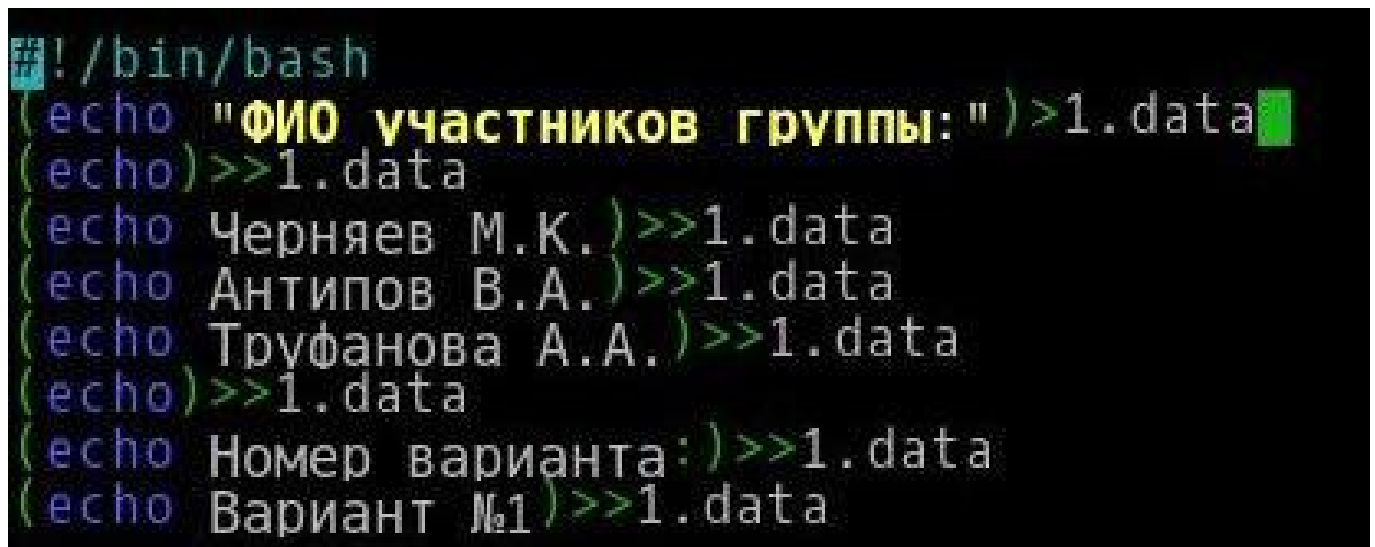
Написание bash-скрипт, который будет выводить Фамилии И.О. и Вариант группы тремя способами:

### 1) Напрямую выводя требуемый текст

На Рисунке 1.1 изображено окно текстового редактора nano.

Небольшое пояснение к коду:

`#!/bin/bash` указывает путь к интерпретатору для исполнения скрипта. С помощью оператора потокового вывода `>>1.data` результат работы скрипта сохраняем в файл *1.data*. Результат работы скрипта и вывод содержимого файла *1.data* можно увидеть на Рисунке 1.2.



```
#!/bin/bash
(echo "ФИО участников группы:")>>1.data
(echo)>>1.data
(echo Черняев М.К.)>>1.data
(echo Антипов В.А.)>>1.data
(echo Труфанова А.А.)>>1.data
(echo)>>1.data
(echo Номер варианта:>>1.data
(echo Вариант №1)>>1.data
```

Рисунок 1.1 - Скрипт написанный в nano.



```
makar@makar:~/lab2$ ls
1.1 1.data lab3_2.sh lab3_3.sh lab3.sh lab4.sh
makar@makar:~/lab2$ bash lab3.sh
makar@makar:~/lab2$ cat 1.data
ФИО участников группы:

Черняев М.К.
Антипов В.А.
Труфанова А.А.

Номер варианта:
Вариант №1
makar@makar:~/lab2$
```

Рисунок 1.2 - Работа скрипта и вывод содержимого файла *1.data*

### 2) Используя переменную

На Рисунке 1.3 изображено окно текстового редактора nano. Результат работы скрипта и вывод содержимого файла *2.data* можно увидеть на Рисунке 1.4.

```

#!/bin/bash
a="ФИО участников группы:"
b="Антипов В.А."
c="Труфанова А.А"
d="Черняев М.к."
(echo $a)>>2.data
(echo)>>2.data
(echo $b)>>2.data
(echo $c)>>2.data
(echo $d)>>2.data
f="Номер варианта:"
g="Вариант№1"
(echo $f)>>2.data
(echo)>>2.data
(echo $g)>>2.data

```

Рисунок 1.3 - Скрипт написанный в папо.

```

makar@makar:~/lab2$ bash lab3_2.sh
makar@makar:~/lab2$ cat 2.data
ФИО участников группы:

Антипов В.А.
Труфанова А.А
Черняев М.к.
Номер варианта:

Вариант№1
makar@makar:~/lab2$ █

```

Рисунок 1.4 - Работа скрипта и вывод содержимого файла *2.data*

### 3) Используя функцию (с локальной переменной)

На Рисунке 1.5 изображено окно текстового редактора папо. Результат работы скрипта и вывод содержимого файла *3.data* можно увидеть на Рисунке 1.6.

```
#!/bin/bash
function my (){
local a="ФИО участников группы:"
local b="Труфанова А.А."
local c="Черняев М.К"
local d="Антипов А.А."
(echo $a)>>3.data
(echo)>>3.data
(echo $b)>>3.data
(echo $c)>>3.data
(echo $d)>>3.data
local e="Номер варианта:"
local f="Вариант№1"
(echo)>>3.data
(echo $e)>>3.data
(echo)>>3.data
(echo $f)>>3.data
}
```

Рисунок 1.5 - Скрипт написанный в папо.

```
makar@makar:~/lab2$ bash lab3_3.sh
makar@makar:~/lab2$ cat 3.data
ФИО участников группы:

Труфанова А.А.
Черняев М.К
Антипов А.А.

Номер варианта:

Вариант№1
makar@makar:~/lab2$ █
```

Рисунок 1.6 - Работа скрипта и вывод содержимого файла *3.data*

## Анализ (достоинства и недостатки) каждого метода из Задания 2

Первый метод является самым легким и примитивным, так как для его реализации нужно знать только самые основы написания скриптов на `bash`. Недостатком является то, что ввод текста каждый раз приходится делать вручную, что не удобно при повторяющемся тексте.

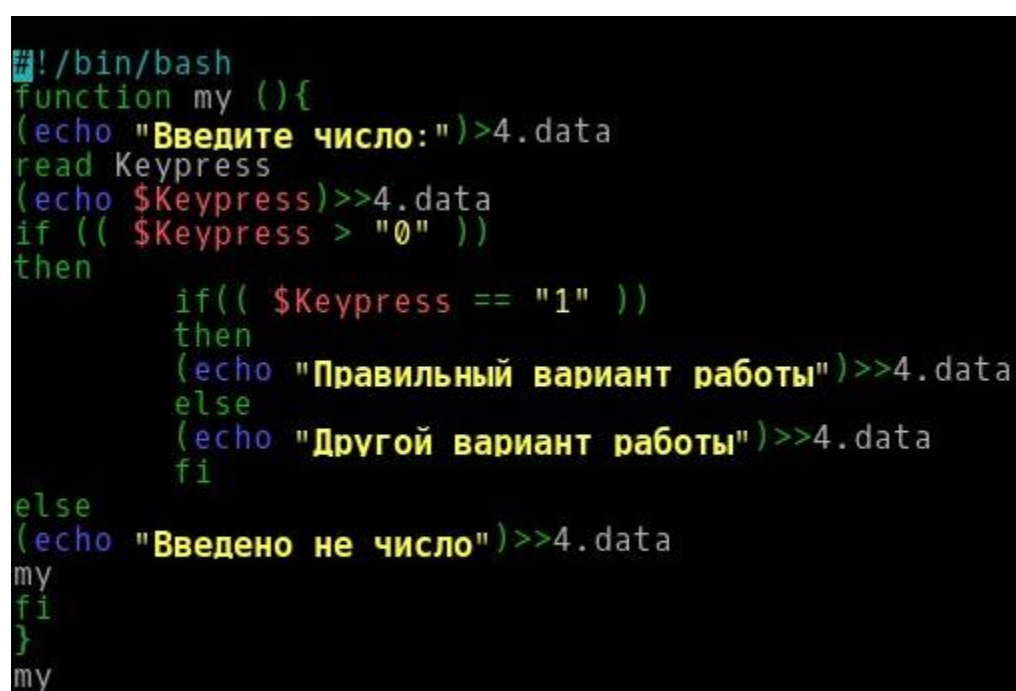
Второй метод является более сложным в применении так как нужно знать основы написания скриптов. Данный метод является более универсальным, так как дублирующийся части текста теперь не нужно вводить снова, можно просто вызвать нужную переменную.

Последний метод самый сложный в реализации, так как для него нужны обширные познания. Этот метод является самым удобным и универсальным, потому что после написания функции, для вывода текста достаточно вызвать такое количество раз, сколько раз нужно вывести данный текст.

Написание скрипта, который будет запрашивать ввести число (учесть возможность ввода не числового значения как ошибки с оповещением и повторным запросом).

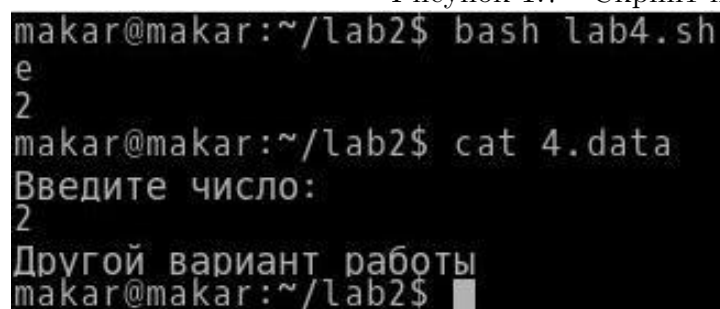
Небольшое пояснение к коду:

Для реализации повторного запроса задаем данный алгоритм ветвления в функции, и при необходимости повторного запроса создаем рекурсию-вызов функцией саму себя, пока не будет достигнут удовлетворяющий нас вариант. На Рисунке 1.7 изображено окно текстового редактора nano. Результат работы скрипта и вывод содержимого файла 4.data можно увидеть на Рисунке 1.8.



```
#!/bin/bash
function my (){
(echo "Введите число:")>4.data
read Keypress
(echo $Keypress)>>4.data
if (( $Keypress > "0" ))
then
    if(( $Keypress == "1" ))
    then
        (echo "Правильный вариант работы")>>4.data
    else
        (echo "Другой вариант работы")>>4.data
    fi
else
    (echo "Введено не число")>>4.data
my
fi
}
my
```

Рисунок 1.7 - Скрипт написанный в nano.



```
makar@makar:~/lab2$ bash lab4.sh
e
2
makar@makar:~/lab2$ cat 4.data
Введите число:
2
Другой вариант работы
makar@makar:~/lab2$ █
```

Рисунок 1.8 - Работа скрипта и вывод содержимого файла 3.data

## 2)Задание «Написание скриптов. Циклы»

1. Изучить и предоставить краткий теоретический материал по использованию аргументов и циклов.

Оболочка `bash` поддерживает циклы `for`, `while`, а также условия `if` и `case`.

Аргумент цикла `for` может принимать последовательно заданные значения, список файлов, директории, значения из файла (с использованием команд прочтения), также это могут быть не только простые(однословные) значения, но также и целые фразы. Если на вход подаётся файл, то разделением выступают знаки табуляции, пробелы и переносы строк, чтобы выбрать например по знаку `:`, следует записать следующую команду `IFS=:`.

Цикл `while` принимает на сравнение значение переменной или выполнение команды. И в `for` и в `while` можно использовать команды `break` и `continue`. Есть возможность использовать вложенные циклы. Также после выполнения цикла переменная хранит последнее значение, и с ней можно работать дальше.

В условии `if` подается команда (выполнение команды = 0 = переход по `then`; невыполнение=1=переход по `else`) или сравнение. Сравнение осуществляется либо численными значениями, либо строками (по кодировке ASCII) условия сравнения строк заключаются в квадратные скобки, чисел в круглые.

2. Напишите скрипт на `bash`, который будет определять, в какую возрастную группу попадают пользователи.

При запуске скрипт должен вывести сообщение "Enter your name:" и ждать от пользователя ввода имени (команда `read`). Затем должно быть выведено имя (Например: «The name you entered is <Имя>. Hi <Имя>!»). После чего у пользователя должен быть выбор, продолжать работу или нет. Когда имя введено и сделан выбор, то скрипт должен написать "Enter your age:" и ждать ввода возраста. Когда возраст введен, скрипт пишет на экран «Имя>, your group is <группа> где <группа> определяется на основе возраста по следующим правилам:

младше либо равно 16: "child

от 17 до 25 (включительно): "youth

старше 25: "adult".

После этого скрипт опять выводит сообщение "Enter your name:" и всё начинается по новой. Если в какой-то момент работы скрипта будет введено пустое имя или возраст 0 (недопустимые символы, буквы итп.), то скрипт должен написать на экран "bye" и закончить свою работу.

Содержимое скрипта можно увидеть на рисунке 2.1  
Был выбран следующий алгоритм работы:

- При запуске скрипт выводит строку с просьбой ввести своё имя (`echo "Enter your name:"`), после чего считывает введенное значение в переменную(`read name`), проверив строку на пустое значение. (`if [ "$name" = "" ]`, в данной конструкции сравнивается значение переменной `'name'` с пустым значением, после чего осуществляется разветвление в `if`. Если пользователь ввел пустое значение (условие выполняется)(`then`), то в файл (Рисунок 2.3)

записывается сообщение 'echo "Bye :c»>"age.txt" и просходит выход из скрипта(exit 0), иначе(else) (пользователь ввел верное значение) скрипт продолжает работу) и в файл (P) записывается сообщение с указанием имени и приветствием(echo "The name you entered is \$name. Hi \$name!»>"age.txt").

- Следующей строкой ползвателю задаётся вопрос о желание продолжить выполнение команды (echo "Do you want continue this program? (Yes/No)"), введенное значение (read answer) также проверяется на пустую строку (if [ -z \$answer ]), после чего обрабатывается в стуктре case: если ползователь согласен (введенное значение определяется, как "да")(Y|y|yes|Yes)), то в файл (Рисунок 2.3) записывается сообщение и скрипт продолжает выполнение (echo "You're amazing c:»>"age.txt"), если ответ ползователя оценивается как отрицательный (No; no; N; n), записывается прощание и производится выход из скрипта ( echo "Bye :c»>"age.txt"; exit 0) .
- Далее отображается просьба ввести возраст (echo "Enter your age:"), после чего считывается введенное значение (read age), в файл (Рисунок 2.3) записывается введенный возраст (echo "\$age»>"age.txt") и выполнется проверка на недопустимые символы, посредством сравнение переменной с 0 (if ((0<"\$age))), из-за чего скрипт воспринимает только целочисленную переменную, как выполнение условия, в противном случае (не целое число), скрипт заносив в файл (Рисунок 2.3) 'Bye :(' и завершает выполнение. Если условие выполнилось, и скрипт продолжает работу, выполняется выбор группы посредством сравнений введенного значения с разграничивающими значениями групп с помощью вложенный условий (if ((" \$age«=16)); then; group=child; elif ((17<=" \$age"&& " \$age«=25)); then; group=youth; else; group=adult; fi).
- Поскольку весь скрипт заключет в бесконечный цикл (while (true); do ... done), он не прекращает свою работу, пока ползоватень не откажется от продолжения работы скрипта, либо не введет недопустимое значение в качестве ответа.
- Запуск, выполнение и выход из скрипта рисунок 2.2

```
#!/bin/bash

while (true)
do

echo "Enter your name:"
read name
if [ "$name" == "" ]
then
    echo "Bye :c">>"age.txt"
    exit 0
else
echo "The name you entered is $name. Hi $name!" >> "age.txt"
fi

echo "Do you want continue this program? (Yes/No)"
read answer
if [ -z $answer ]
then
    echo "Bye :c">>"age.txt"
    exit 0
else
case "$answer" in
Y|y|yes|Yes) echo "You're amazing c:" >>"age.txt"
;;
N|n|no|No) echo "Bye :c">>"age.txt"
exit 0
;;
esac
fi

echo "Enter your age:"
read age
if ((0<"$age"))
then
    if ((" $age" <= 16))
    then
        group=child

    elif ((17<= "$age" && "$age" <= 25))
    then
        group=youth

    else
        group=adult
    fi
    echo "$name, your group is $group.">>"age.txt"
else
    echo "Bye :c">>"age.txt"
    exit 0
fi

done
```

Рисунок 2.1 скрипт для определения возрастной группы



```

keks@keks101:~/Documents/IT/Lab3$ touch age.txt
keks@keks101:~/Documents/IT/Lab3$ bash 2.sh
Enter your name:
Keks
Do you want continue this program? (Yes/No)
Yes
Enter your age:
3
Enter your name:
Shkiper
Do you want continue this program? (Yes/No)
yes
Enter your age:
21
Enter your name:
Kuki
Do you want continue this program? (Yes/No)
y
Enter your age:
83
Enter your name:
Jek
Do you want continue this program? (Yes/No)
Y
Enter your age:
*(
2.sh: line 35: ((: 0<*( : syntax error: operand expected (error token is "*(")
keks@keks101:~/Documents/IT/Lab3$ █

```

Рисунок 2.2 Запуск, ход выполнения и завершение выполнения скрипта на терминале

```

The name you entered is Keks. Hi Keks!
You're amazing c:
3
Keks, your group is child.
The name you entered is Shkiper. Hi Shkiper!
You're amazing c:
21
Shkiper, your group is youth.
The name you entered is Kuki. Hi Kuki!
You're amazing c:
83
Kuki, your group is adult.
The name you entered is Jek. Hi Jek!
You're amazing c:
*(
Bye :c
The name you entered is John. Hi John!
Bye :c

```

Рисунок 2.3 текстовый файл age.txt с результатом работы скрипта

### 3) Написание скриптов на `bash`. Математические вычисления.

#### 1) Скрипт для отображение чисел Фибоначи от 0 до N для заданного N, при $N \in \mathbb{R}$

На рисунке 3.2 изображено окно текстового редактора *sublime*.

Небольшое пояснение к коду:

`#!/bin/bash` указывает путь к интерпретатору для исполнения скрипта.

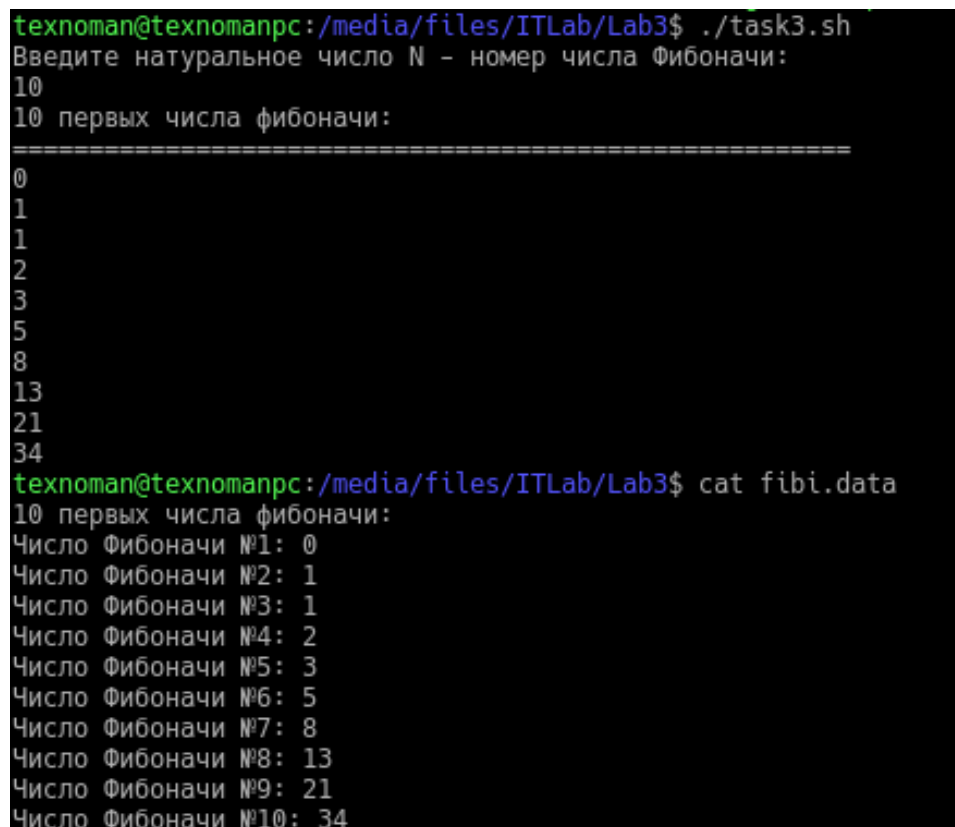
`printf` - команда перекочевавшая из языка *C*, более продвинутый аналог *echo* для вывода сообщения в стандартное устройство вывода. В качестве параметров принимает 'формат вывода' 'Тест для вывода'

Номер, до которого нужно вычислять последовательность чисел Фибоначи получаем из потока ввода с командной строки. Он хранится в переменной `$N`. После записи строки в переменную происходит проверка, на то чтобы число было целым или натуральным, так как последовательность Фибоначи определена только для целых чисел. Если все хорошо, переменная передается в качестве параметра функции `fun()`

В программе спользуется цикл `for` для контроля номера числа Фибоначи. В качестве начальных известных значений чисел Фибоначи взяты первые три числа.

С помощью оператора потокового вывода `>` результат работы скрипта сохраняем в файл `fibi.data`

Результат работы скрипта и вывод содержимого файла `fibi.data` можно увидеть на Рисунке 3.1.



```
texnoman@texnomanpc:/media/files/ITLab/Lab3$ ./task3.sh
Введите натуральное число N - номер числа Фибоначи:
10
10 первых числа фибоначи:
=====
0
1
1
2
3
5
8
13
21
34
texnoman@texnomanpc:/media/files/ITLab/Lab3$ cat fibi.data
10 первых числа фибоначи:
Число Фибоначи №1: 0
Число Фибоначи №2: 1
Число Фибоначи №3: 1
Число Фибоначи №4: 2
Число Фибоначи №5: 3
Число Фибоначи №6: 5
Число Фибоначи №7: 8
Число Фибоначи №8: 13
Число Фибоначи №9: 21
Число Фибоначи №10: 34
```

Рисунок 3.1 - результат работы скрипта по поиску чисел Фибоначи,

вывод содержимого файла, хранящего результат работы скрипта.

```
1  #!/bin/bash
2
3  fun(){
4      echo "$1 первых числа фибоначи:"
5      echo "=====
6      (echo "$1 первых числа фибоначи:")>"fibi.data"
7      for((i=0; i<3 && i<N; i++)) do
8          echo ${arrfibo[i]}
9          (printf '%b' 'Число Фибоначи №' $((i+1)) ': ' ${arrfibo[i]})>>"fibi.data"
10         (printf '%b' '\n')>>"fibi.data"
11     done
12     for((i=4; i<=$1; i++)) do
13         fib=$((fiblast+fibfirst))
14         fibfirst=$((fiblast))
15         fiblast=$((fib))
16         echo "$fiblast"
17         (echo "Число Фибоначи №$i: $fiblast")>>"fibi.data"
18     done
19 }
20
21 fiblast=1
22 fibfirst=1
23 fib=0
24 arrfibo=(0 1 1)
25 echo "Введите натуральное число N - номер числа Фибоначи:"
26 read N
27 if [[ $N == *.* ]]; then
28     echo "Необходимо натуральное число! Выход"
29     exit 0
30 elif [[ $N == -* ]]; then
31     echo "Необходимо только положительное число!"
32 elif (( $N > '0' )); then
33     fun $N
34 else
35     echo "Недопустимые символы, необходимо число!"
36     exit 0
37 fi
```

Рисунок 3.2 - скрипт для вычисления последовательности Фибоначи от 0 до N

## 2) Скрипт вычисляющий факториал N для заданного N, при $N \in \mathbb{R}$

На рисунке 3 изображено окно того же текстового редактора *sublime*, показывающего код программы.

### Пояснение к коду:

`#!/bin/bash` все также указывает путь к интерпретатору для исполнения скрипта (bash).

Команда *let* производит арифметические операции над числами и переменными, в данном случае умножение на число и одновременное присвоение ему нового значения (`*=`). Оператор разыменования переменной `$` для *let* не нужен.

Данный скрипт считает факториал для заданного числа N (считанного с консоли). В начале в программе проверяется является ли введенная строка числом, натуральное ли это число. Затем если все вышеупомянутые условия соблюдены оно передается в качестве параметра функции *fun()*, где и происходит итеративное вычисление факториала.

С помощью оператора потокового вывода `>` (перезаписывает файл) результат работы скрипта сохраняем в файл *factor.data*. Результат работы скрипта и вывод содержимого файла *factor.data*

можно увидеть на Рисунке 4.

```
2  #!/bin/bash
3
4  fun(){
5      N=$1
6      fact=1
7      for((i=2;i<=$N; i++)) do
8          let "fact*=i"
9      done
10     echo "Факториал от $N = $fact"
11     (echo "Факториал от $N = $fact")>"factor.data"
12
13 }
14
15 echo "Введите натуральное число N для вычисления факториала:"
16 read N
17 if [[ $N == *.* ]]
18 then
19     echo "Необходимо натуральное число! Выход"
20     exit 0
21 elif [[ $N == -* ]]
22 then
23     echo "Необходимо только неотрицательное число! Выход"
24 elif (( $N >= '0' )); then
25     fun $N
26 else
27     echo "Недопустимые символы, необходимо число! Выход"
28     exit 0
29 fi
```

Рисунок 3.3 - скрипт для вычисления факториала для числа N.

```
texnoman@texnomanpc:/media/files/ITLab/Lab3$ ./task3_2.sh
Введите натуральное число N для вычисления факториала:
12
Факториал от 12 = 479001600
texnoman@texnomanpc:/media/files/ITLab/Lab3$ cat factor.data
Факториал от 12 = 479001600
texnoman@texnomanpc:/media/files/ITLab/Lab3$ ./task3_2.sh
Введите натуральное число N для вычисления факториала:
0
Факториал от 0 = 1
texnoman@texnomanpc:/media/files/ITLab/Lab3$ cat factor.data
Факториал от 0 = 1
texnoman@texnomanpc:/media/files/ITLab/Lab3$ ./task3_2.sh
Введите натуральное число N для вычисления факториала:
20
Факториал от 20 = 2432902008176640000
texnoman@texnomanpc:/media/files/ITLab/Lab3$ cat factor.data
Факториал от 20 = 2432902008176640000
```

Рисунок 3.4 - результат работы скрипта по поиску факториала числа N, вывод содержимого файла, хранящего результат работы скрипта.

## Вывод

В качестве вывода можно выделить некоторые особенности написания скриптов на `bush`: так например, вокруг знаков присвоения недопустимо ставить пробелы, а при написании условия в `if` напротив перед скобками следует ставить пробелы, также были изучены способности заносить в циклы и условия не только математические операции, но и выполнение/невыполнение различных команд. Конечно были изучены структура и синтаксис циклов, условий, функций и некоторых операций для написания скриптов, о чем изложено в работе выше.

Вкупе с приведенными ранее моментами можно сказать о том, что нами были изучены и освоены основы написания скриптов в оболочке `bush`. А также были решены небольшие практические задачи, подробнее о которых можно узнать из пунктов 1-3.



Спасибо за внимание