

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

Лабораторная работа №11

---

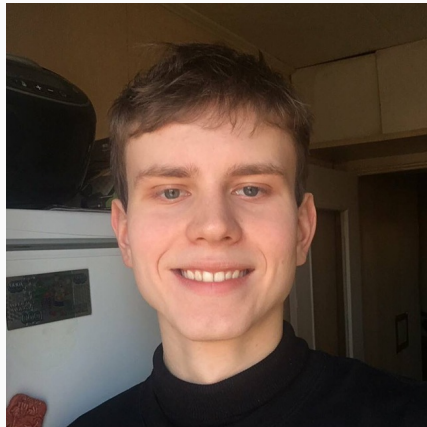
Татур С. А.

Российский университет дружбы народов, Москва, Россия

## Информация

---

- Татур Стефан Андреевич
- студент 1 курса, группа НММбд-03-22
- Российский университет дружбы народов



## Вводная часть

---

- Командный процессор ОС UNIX
- Командные файлы

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

- Ознакомиться с теоретическим материалом.
- Выполнить упражнения.
- Ответить на контрольные вопросы.

# Выполнение лабораторной работы №11

---



# Первая программа

```
#!/bin/bash
iflag=0 oflag=0 pflag=0 cflag=0 nflag=0
while getopts :iopcn optletter
do case $optletter in
    i) iflag=1; eval `date +%Y%m%d`;;
    o) oflag=1; eval `date +%Y%m%d`;;
    p) pflag=1; eval `date +%Y%m%d`;;
    c) cflag=1;;
    n) nflag=1;;
    *) echo "illegal option $optletter"
       exit
    esac
done
if ((iflag)); then echo "mission no vulgare"
else
    if ((oflag)); then echo "mission no vulgare"
    else
        if ((pflag)); then
            if ((cflag)); then
                then if ((nflag)); then
                    then grep $eval $eval > $eval
                    else grep -i $eval $eval
                fi
            else if ((nflag)); then
                then grep -i $eval $eval
                else grep -i $eval $eval
            fi
        else if ((cflag)); then
            then if ((nflag)); then
                then grep $eval $eval > $eval
                else grep -i $eval $eval > $eval
            fi
        else if ((nflag)); then
            then grep -i $eval $eval > $eval
            else grep -i $eval $eval > $eval
        fi
    fi
fi
fi
```

```
~]$ chmod +x lab11_1.sh
```

```
./lab11_1.sh -i 1.txt -o 2.txt -p Easy -C -n
cat 2.txt
```

## Вторая программа

```
Открыть ▾ [icon] lab11_2.c [icon] [icon] [icon] [icon]
#include <stdio.h>
#include <stdlib.h>
int main ()
{
    printf("Введите число\n");
    int a;
    scanf ("%d", &a);
    if (a<0) exit(0);
    if (a>0) exit(1);
    if (a==0) exit(2);
    return 0;
}
```

```
Открыть ▾ [icon] lab11_2.sh [icon] [icon] [icon] [icon]
lab11_2.c lab11_2.sh [icon]
#!/bin/bash
gcc lab11_2.c -o lab11_2
./lab11_2
code=$?
case $code in
    0) echo "Число меньше 0"
    1) echo "Число больше 0"
    2) echo "Число равно 0"
esac
```

```
~]$ chmod +x lab11_2.sh
```

# Третья программа

```
Открыть ▾ [icon] lab11_3.sh [icon] [icon] [icon] x
#!/bin/bash
opt=$1;
format=$2;
number=$3;
function Files()
{
    for (( i=1; i<=number; i++ )) do
        file=$(echo $format | tr 'd' 'i')
        if [ $opt == "-f" ]
        then
            rm -f $file
        elif [ $opt == "-c" ]
        then
            touch $file
        fi
    done
}
```

```
~]$ chmod +x lab11_3.sh
```

1.txt	lab11_3.sh	work	documents	configurations	development	lab11
lab09.sh	lab11_3.sh	apps	library	music	'Рабочий стол'	

# Четвёртая программа

```
Открыть ▾  lab11_4.sh   
#!/bin/bash
files=$(find ./ -maxdepth 1 -mtime -7)
listing=""
for file in "$files" ; do
    file=$(echo "$file" | cut -c 3-)
    listing="$listing $file"
done
dir=$(basename $(pwd))
tar -cvf $dir.tar $listing
```

```
~]$ chmod +x lab11_4.sh
```

```
work/
work/study/
work/study/2022-2023/
work/study/2022-2023/Архитектура компьютера/
work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/
work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/.git/
work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/.git/branches/
work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/.git/hooks/
work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/.git/hooks/applypatch.sample
work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/.git/hooks/commit-msg.sample
work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/.git/hooks/fsmonitor-steelman.sample
work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/.git/hooks/post-update.sample
work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/.git/hooks/pre-applypatch.sample
work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/.git/hooks/pre-commit.sample
work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/.git/hooks/pre-merge-commit.sample
work/study/2022-2023/Архитектура компьютера/study_2022-2023_arh-pc/.git/hooks/pre-push.s
```

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

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.

## Результаты

---

В ходе выполнения лабораторной работы были изучены основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.