

# Лабораторная работа No 11.

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

---

Брасалес Сарасбати

22.04.2023

Российский университет дружбы народов, Москва, Россия

## Информация

---

- Сарасбати Брасалес
- Российский университет дружбы народов
- [sarasbati2904@gmail.com](mailto:sarasbati2904@gmail.com)

## Вводная часть

---

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

1. Используя команды `getopts` `grep`, написать командный файл, который анализирует командную строку с ключами: `-iinputfile` — прочитать данные из указанного файла; `-ooutputfile` — 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`).

## Выполнение работы

---



## Первый командный файл

```
Open ▾  *script1.sh  
~/lab11  
1 #!/bin/bash  
2  
3 while getopts i:o:p:cn 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 $oflag  
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  
26
```

```
[sbrasales@fedora lab11]$ gedit scscript1.sh
[sbrasales@fedora lab11]$ bash scscript1.sh -p lia -i input.txt -o output.txt -cn
[sbrasales@fedora lab11]$ gedit output.txt
[sbrasales@fedora lab11]$ gedit output.txt
[sbrasales@fedora lab11]$ gedit input.txt
[sbrasales@fedora lab11]$ bash scscript1.sh -p lia -i input.txt -o output.txt -cn
[sbrasales@fedora lab11]$ gedit output.txt
```

## Второй командный файл

Open 

program1.c  
~/lab11

```
1 #include <stdlib.h>
2 #include <stdio.h>
3
4 `int main () {' |
5     int n
6     printf ("Input number: ");
7     scanf ("%d", &n);
8     if (n>0) {
9         exit(1);
10    } else if (n==0) {
11        exit(1);
12    } else {
13        exit(2);
14    }
15 }
16
```

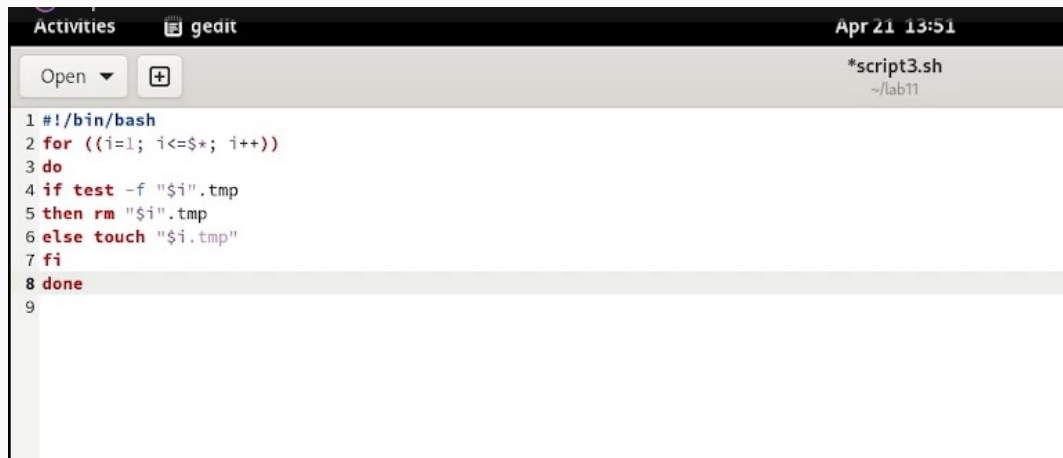
## Второй командный файл



The image shows a terminal window with a light gray title bar. On the left, there is an 'Open' button with a dropdown arrow and a '+' icon. On the right, the file name '\*script11.sh' is displayed above the path '~/lab11'. The terminal content consists of a shell script with line numbers 1 through 9. The script starts with a shebang, compiles a C program, and then uses a case statement to print messages based on the input number. The script ends with 'esac'.

```
1 #!/bin/bash
2
3 gcc -o cprog program1.c
4 ./cprog
5 case $? in
6 0) echo "Number equal to 0";;
7 2) echo "Number less than 0";;
8 3) echo "NUmber greater than 0";;
9 esac
```

## Третий командный фал



The screenshot shows a gedit text editor window. The title bar at the top includes 'Activities', the gedit icon, and the filename 'script3.sh'. The window's toolbar contains an 'Open' button with a dropdown arrow and a '+' icon. The main editing area displays a shell script with the following content:

```
1 #!/bin/bash
2 for ((i=1; i<=$*; i++))
3 do
4 if test -f "$i".tmp
5 then rm "$i".tmp
6 else touch "$i.tmp"
7 fi
8 done
9
```

```
[sbrasales@fedora lab11]$ bash script3.sh 3  
[sbrasales@fedora lab11]$ ls  
1.tmp 3.tmp      output.txt  program1.c  script2.sh  
2.tmp input.txt  program1.c  script1.sh  script3.sh  
[sbrasales@fedora lab11]$
```

## Четвертый командный файл



The image shows a code editor window with a title bar that includes an 'Open' button with a dropdown arrow and a '+' icon. The file name 'script4.sh' and its path '~/lab11' are displayed on the right. The editor contains a shell script with four lines: a shebang, a blank line, a 'find' command, and a 'tar' command. The third and fourth lines are highlighted with a light gray background.

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

```
[sbrasales@fedora lab11]$ touch script4.sh
[sbrasales@fedora lab11]$ gedit script4.sh
[sbrasales@fedora lab11]$ bash script4.sh /home/sbrasales/work
tar: Removing leading '/' from member names
tar: Removing leading '/' from hard link targets
[sbrasales@fedora lab11]$ ls
archive.tar  input.txt  program1.c  scrip1.sh  script3.sh
FILES.txt    output.txt program1.c  script2.sh  script4.sh
[sbrasales@fedora lab11]$
```



## Вывод

---

Мы изучили основы программирования в оболочке ОС UNIX и научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

- Pandoc: преобразователь текстовых файлов
- Сайт: <https://pandoc.org/>
- Репозиторий: <https://github.com/jgm/pandoc>

- Использование LaTeX
- Пакет для презентации: beamer
- Тема оформления: **metropolis**

```
slide_level: 2  
aspectratio: 169  
section-titles: true  
theme: metropolis
```

- Используется фреймворк `reveal.js`
- Используется тема `beige`

- Тема задаётся в файле `Makefile`

```
REVEALJS_THEME = beige
```

```
...
```

- На слайд выносится та информация, которая без зрительной опоры воспринимается хуже

- На слайд выносится та информация, которая без зрительной опоры воспринимается хуже
- Слайды должны дополнять или обобщать содержание выступления или его частей, а не дублировать его



- На слайд выносится та информация, которая без зрительной опоры воспринимается хуже
- Слайды должны дополнять или обобщать содержание выступления или его частей, а не дублировать его
- Информация на слайдах должна быть изложена кратко, чётко и хорошо структурирована

- На слайд выносится та информация, которая без зрительной опоры воспринимается хуже
- Слайды должны дополнять или обобщать содержание выступления или его частей, а не дублировать его
- Информация на слайдах должна быть изложена кратко, чётко и хорошо структурирована
- Слайд не должен быть перегружен графическими изображениями и текстом

- На слайд выносится та информация, которая без зрительной опоры воспринимается хуже
- Слайды должны дополнять или обобщать содержание выступления или его частей, а не дублировать его
- Информация на слайдах должна быть изложена кратко, чётко и хорошо структурирована
- Слайд не должен быть перегружен графическими изображениями и текстом
- Не злоупотребляйте анимацией и переходами

- Лучше представить в виде схемы

- Лучше представить в виде схемы
- Менее оптимально представить в виде рисунка, графика, таблицы

- Лучше представить в виде схемы
- Менее оптимально представить в виде рисунка, графика, таблицы
- Текст используется, если все предыдущие способы отображения информации не подошли