

Лабораторная работа-11

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Световидова Полина НБИбд-04-22

Содержание

1	Цель работы	5
2	Задания	6
3	Выполнение лабораторной работы	7
4	Вывод:	12
5	Ответы на контрольные вопросы:	13

Список иллюстраций

3.1	Вставил в файл любой текст из интернета	7
3.2	Пишу первый скрипт	8
3.3	Проверяю в терминале	8
3.4	Пишу новый скрипт-на языке Си	9
3.5	Пишу еще один скрипт	9
3.6	Проверяю все в терминале	9
3.7	Пишу новый скрипт	10
3.8	Проверяю его в терминале	10
3.9	Пишу новый скрипт	11
3.10	Проверил его в терминале	11

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 Задания

1. Используя команды `grep`, написать командный файл, который анализирует командную строку с ключами: `-i` — прочитать данные из указанного файла; `-o` — 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 `-p`.
2. Написать на языке Си программу, которая вводит число и определяет, является оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию о коде 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 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

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

а затем ищет в указанном файле нужные строки, определяемые ключом `-р`. (рис. [3.1])(рис. [3.2])(рис. [3.3])

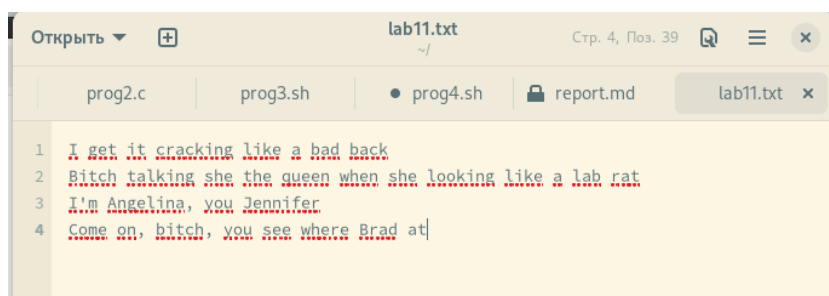
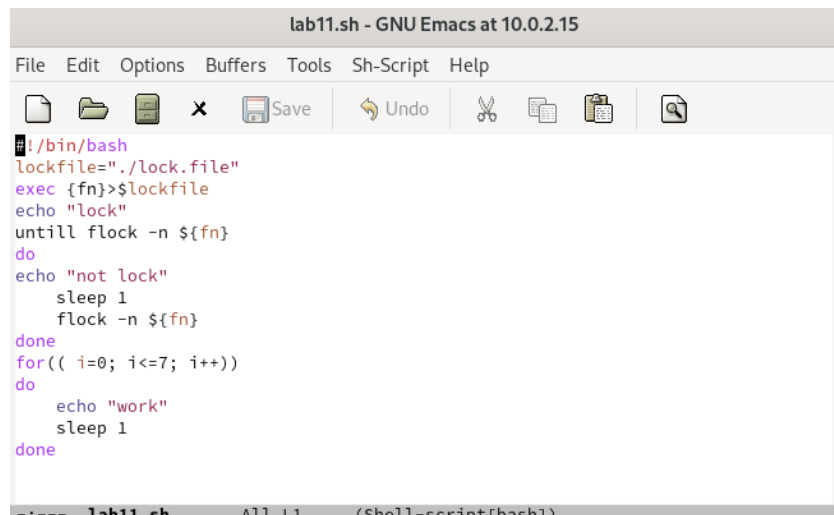
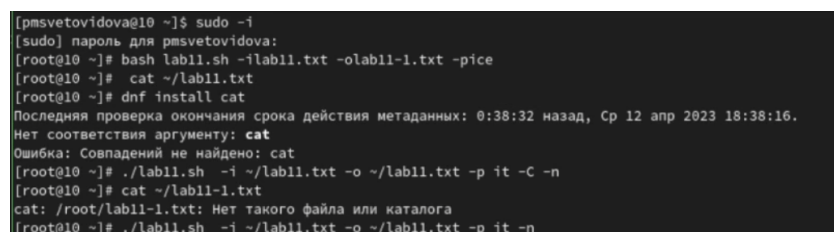


Рис. 3.1: Вставил в файл любой текст из интернета



```
#!/bin/bash
lockfile='./lock.file'
exec {fn}>$lockfile
echo "lock"
until flock -n ${fn}
do
echo "not lock"
sleep 1
flock -n ${fn}
done
for(( i=0; i<=7; i++))
do
echo "work"
sleep 1
done
```

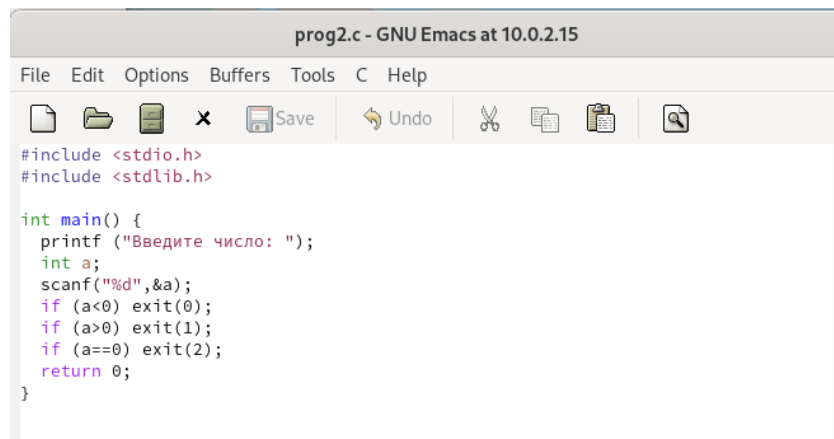
Рис. 3.2: Пишу первый скрипт



```
[pmsvetovidova@i0 ~]$ sudo -i
[sudo] пароль для pmsvetovidova:
[root@i0 ~]# bash lab11.sh -ilab11.txt -olab11-1.txt -pice
[root@i0 ~]# cat ~/lab11.txt
[root@i0 ~]# dnf install cat
Последняя проверка окончания срока действия метаданных: 0:38:32 назад, Ср 12 апр 2023 18:38:16.
Нет соответствия аргументу: cat
Ошибка: Совпадений не найдено: cat
[root@i0 ~]# ./lab11.sh -i ~/lab11.txt -o ~/lab11.txt -p it -C -n
[root@i0 ~]# cat ~/lab11-1.txt
cat: /root/lab11-1.txt: Нет такого файла или каталога
[root@i0 ~]# ./lab11.sh -i ~/lab11.txt -o ~/lab11.txt -p it -n
```

Рис. 3.3: Проверяю в терминале

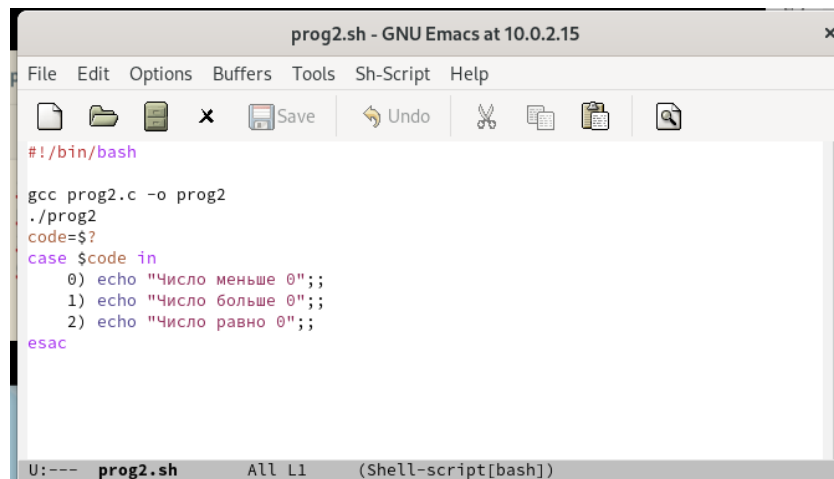
2. Написала на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции `exit(n)`, передавая информацию в о коде завершения в оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, какое число было введено.(рис. [3.4])(рис. [3.5])(рис. [3.6])



```
#include <stdio.h>
#include <stdlib.h>

int main() {
    printf("Введите число: ");
    int a;
    scanf("%d",&a);
    if (a<0) exit(0);
    if (a>0) exit(1);
    if (a==0) exit(2);
    return 0;
}
```

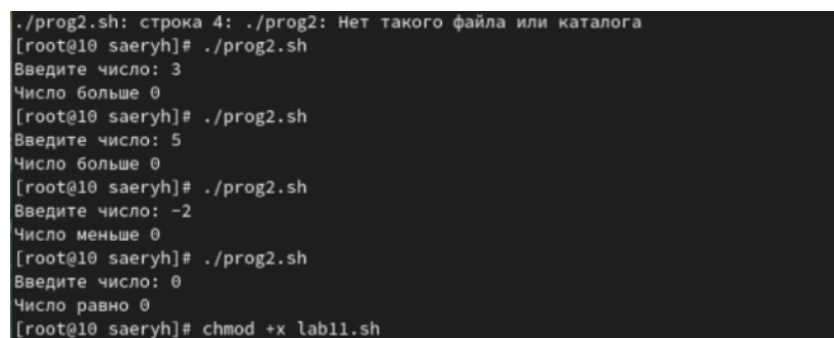
Рис. 3.4: Пишу новый скрипт-на языке Си



```
#!/bin/bash

gcc prog2.c -o prog2
./prog2
code=$?
case $code in
    0) echo "Число меньше 0";;
    1) echo "Число больше 0";;
    2) echo "Число равно 0";;
esac
```

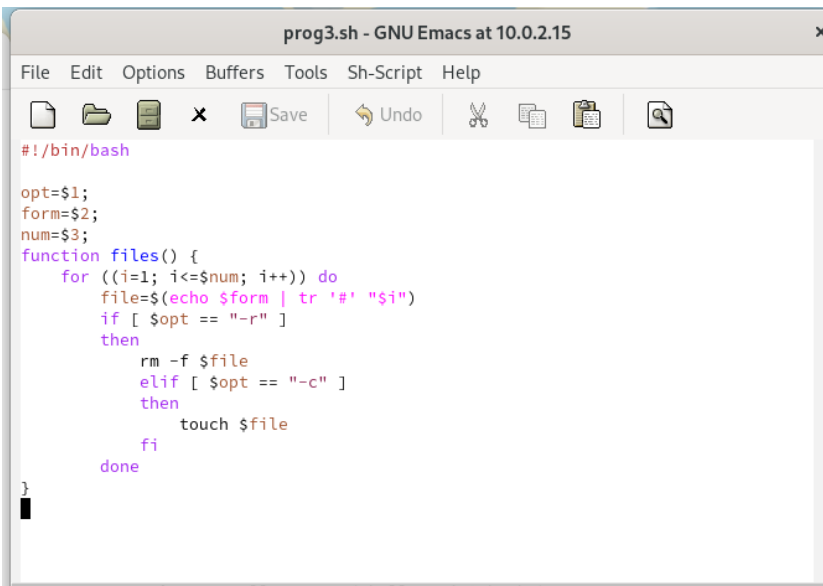
Рис. 3.5: Пишу еще один скрипт



```
./prog2.sh: строка 4: ./prog2: Нет такого файла или каталога
[root@10 saeryh]# ./prog2.sh
Введите число: 3
Число больше 0
[root@10 saeryh]# ./prog2.sh
Введите число: 5
Число больше 0
[root@10 saeryh]# ./prog2.sh
Введите число: -2
Число меньше 0
[root@10 saeryh]# ./prog2.sh
Введите число: 0
Число равно 0
[root@10 saeryh]# chmod +x lab11.sh
```

Рис. 3.6: Проверяю все в терминале

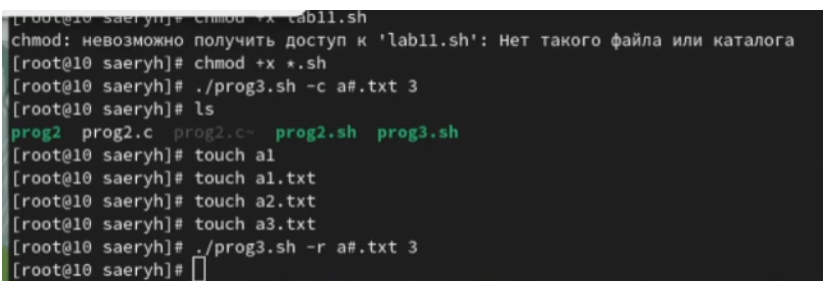
3. Написала командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (например 1.tmp, 2.tmp, 3.tmp, 4.tmp и т.д.). Число файлов, которые необходимо создать, передаётся в аргументы командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).(рис. [3.7])(рис. [3.8])



```
#!/bin/bash

opt=$1;
form=$2;
num=$3;
function files() {
    for ((i=1; i<=$num; i++)) do
        file=$(echo $form | tr '#' "$i")
        if [ $opt == "-r" ]
        then
            rm -f $file
        elif [ $opt == "-c" ]
        then
            touch $file
        fi
    done
}
```

Рис. 3.7: Пишу новый скрипт



```
[root@10 saeryh]# chmod +x lab11.sh
chmod: невозможно получить доступ к 'lab11.sh': Нет такого файла или каталога
[root@10 saeryh]# chmod +x *.sh
[root@10 saeryh]# ./prog3.sh -c a#.txt 3
[root@10 saeryh]# ls
prog2 prog2.c prog2.c~ prog2.sh prog3.sh
[root@10 saeryh]# touch a1
[root@10 saeryh]# touch a1.txt
[root@10 saeryh]# touch a2.txt
[root@10 saeryh]# touch a3.txt
[root@10 saeryh]# ./prog3.sh -r a#.txt 3
[root@10 saeryh]#
```

Рис. 3.8: Проверяю его в терминале

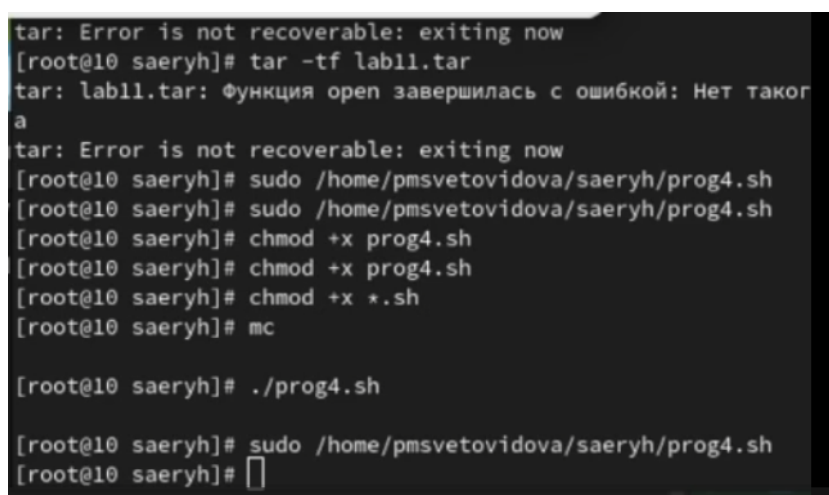
4. Написала командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировала его так, чтобы

запаковывались только те файлы, которые были изменены менее недели тому назад (использовал команду find).(рис. [3.9])(рис. [3.10])



```
Открыть ▾ + prog4.sh ~/saeryh Стр. 9, Пос. 6
prog2.sh prog2.c prog3.sh prog4.sh x
1 #!/bin/bash
2
3 files=$(find ./ -maxdepth 1 -mtime -7)
4 listing=""
5 for file in "$files" ; do
6     file=$(echo "$file" | cut -c 3-)
7     listing+="$listing $file"
8 done
9 dir=$|
```

Рис. 3.9: Пишу новый скрипт



```
tar: Error is not recoverable: exiting now
[root@10 saeryh]# tar -tf lab11.tar
tar: lab11.tar: Функция open завершилась с ошибкой: Нет такого файла
tar: Error is not recoverable: exiting now
[root@10 saeryh]# sudo /home/pmsvetovidova/saeryh/prog4.sh
[root@10 saeryh]# sudo /home/pmsvetovidova/saeryh/prog4.sh
[root@10 saeryh]# chmod +x prog4.sh
[root@10 saeryh]# chmod +x prog4.sh
[root@10 saeryh]# chmod +x *.sh
[root@10 saeryh]# mc

[root@10 saeryh]# ./prog4.sh

[root@10 saeryh]# sudo /home/pmsvetovidova/saeryh/prog4.sh
[root@10 saeryh]#
```

Рис. 3.10: Проверил его в терминале

4 Вывод:

Изучила основы программирования в оболочке ОС UNIX, научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

5 Ответы на контрольные вопросы:

1. Команда `getopts` является встроенной командой командной оболочки `bash`, предназначенной для разбора параметров сценариев. Она обрабатывает исключительно однобуквенные параметры как с аргументами, так и без них и этого вполне достаточно для передачи сценариям любых входных данных.
2. При генерации имен используют метасимволы:
 - произвольная (возможно пустая) последовательность символов; `?` один произвольный символ; `[...]` любой из символов, указанных в скобках перечислением и/или с указанием диапазона; `cat f*` выдаст все файлы каталога, начинающиеся с `"f"`; `cat f` выдаст все файлы, содержащие `"f"`; `cat program.?` выдаст файлы данного каталога с однобуквенными расширениями, скажем `"program.c"` и `"program.o"`, но не выдаст `"program.com"`; `cat [a-d]*` выдаст файлы, которые начинаются с `"a"`, `"b"`, `"c"`, `"d"`. Аналогичный эффект дадут и команды `"cat [abcd]"` и `"cat [bdac]"`.
3. Операторы `&&` и `||` являются управляющими операторами. Если в командной строке стоит `command1 && command2`, то `command2` выполняется в том, и только в том случае, если статус выхода из команды `command1` равен нулю, что говорит об успешном ее завершении. Аналогично, если командная строка имеет вид `command1 || command2`, то команда `command2` выполняется тогда, и только тогда, когда статус выхода из команды `command1` отличен от нуля.

4. Оператор `break` завершает выполнение ближайшего включающего цикла или условного оператора, в котором он отображается.
5. Команда `true` всегда возвращает ноль в качестве выходного статуса для индикации успеха. Команда `false` всегда возвращает не-ноль в качестве выходного статуса для индикации неудачи. Во всех управляющих конструкциях в качестве логического значения используется код возврата из программы, указанной в качестве условия. Код возврата 0 – истина, любое другое значение – ложь. Программа `true` – всегда завершается с кодом 0, `false` – всегда завершается с кодом 1.
6. Введенная строка означает условие существования файла `mans/i.$s`
7. Цикл `While` выполняется до тех пор, пока указанное в нем условие истинно. Когда указанное условие становится ложным - цикл завершается. Цикл `Until` выполняется до тех пор, пока указанное в нем условие ложно.