

Лабораторная Работа 11

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

Дмитрий Владимирович Орлюк

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Выводы	13
5	Ответы на контрольные вопросы:	14

Список иллюстраций

3.1	Создал файлы	7
3.2	ТХТ	8
3.3	SH	9
3.4	Какие-то махинации	10
3.5	Прога на С	11
3.6	Скрипт для SH	11
3.7	Проверка работоспособности	12

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 Задание

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` — 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 ли 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`, написал командный файл, который анализирует командную строку с ключами: `-i` `inputfile` — прочитать данные из указанного файла; `-o` `outputfile` — 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`. (рис. [3.1])(рис. [3.2])(рис. [3.7])

```
dvorlyuk@dk3n60 ~ $ touch lab11.txt
dvorlyuk@dk3n60 ~ $ touch lab11.sh
dvorlyuk@dk3n60 ~ $ chmod +x lab11.sh
```

Рис. 3.1: Создал файлы

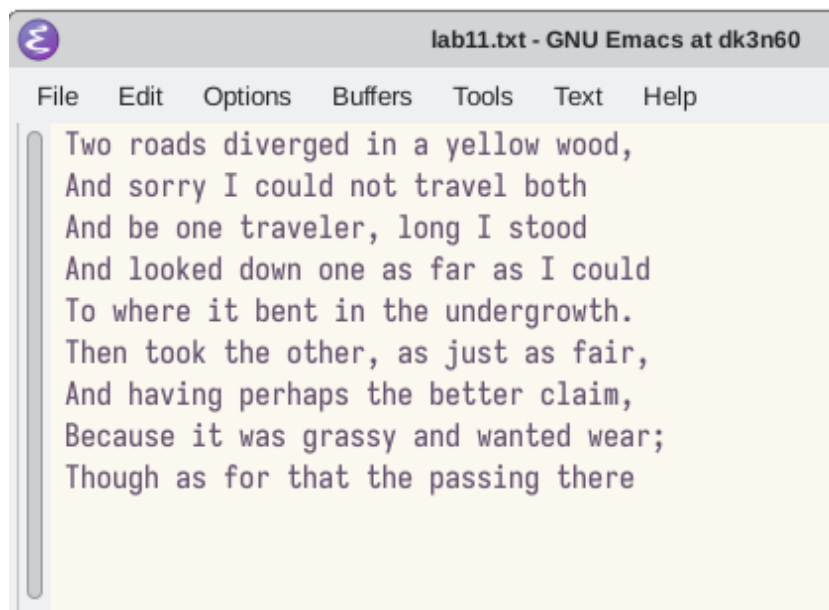
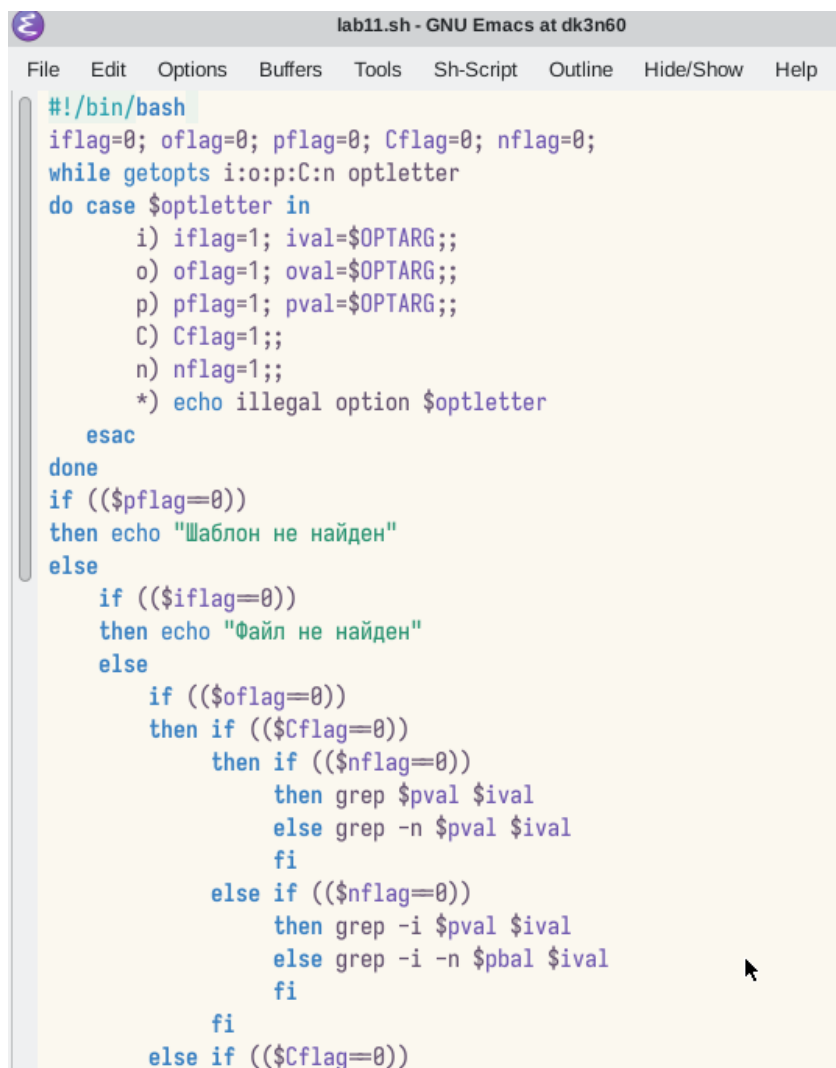


Рис. 3.2: TXT



```
lab11.sh - GNU Emacs at dk3n60
File Edit Options Buffers Tools Sh-Script Outline Hide/Show Help

#!/bin/bash
iflag=0; oflag=0; pflag=0; Cflag=0; nflag=0;
while getopts i:o:p:C:n optletter
do case $optletter in
    i) iflag=1; ival=$OPTARG;;
    o) oflag=1; oval=$OPTARG;;
    p) pflag=1; pval=$OPTARG;;
    C) Cflag=1;;
    n) nflag=1;;
    *) echo illegal option $optletter
    esac
done
if (($pflag==0))
then echo "Шаблон не найден"
else
    if (($iflag==0))
    then echo "Файл не найден"
    else
        if (($oflag==0))
        then if (($Cflag==0))
            then if (($nflag==0))
                then grep $pval $ival
                else grep -n $pval $ival
                fi
            else if (($nflag==0))
                then grep -i $pval $ival
                else grep -i -n $pval $ival
                fi
            fi
        else if (($Cflag==0))
```

Рис. 3.3: SH

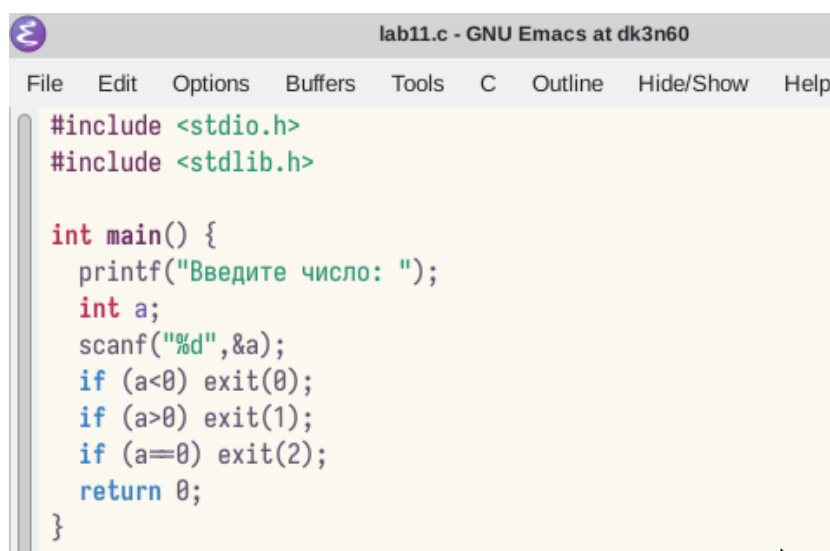
```

dvorlyuk@dk3n60 ~ $ bash lab11.sh -ilab11.txt -olab11-1.txt -pice
dvorlyuk@dk3n60 ~ $ cat ~/lab11.txt
dvorlyuk@dk3n60 ~ $ cat ~/lab11.txt
Two roads diverged in a yellow wood,
And sorry I could not travel both
And be one traveler, long I stood
And looked down one as far as I could
To where it bent in the undergrowth.
Then took the other, as just as fair,
And having perhaps the better claim,
Because it was grassy and wanted wear;
Though as for that the passing there
dvorlyuk@dk3n60 ~ $ ./lab11.sh -i ~/lab11.txt -o ~/lab11-1.txt -p it -C -n
dvorlyuk@dk3n60 ~ $ cat ~/lab11-1.txt
To where it bent in the undergrowth.
Because it was grassy and wanted wear;
dvorlyuk@dk3n60 ~ $ ./lab11.sh -i ~/lab11.txt -o ~/lab11-1.txt -p it -n
dvorlyuk@dk3n60 ~ $ cat ~/lab11-1.txt
5:To where it bent in the undergrowth.
8:Because it was grassy and wanted wear;
dvorlyuk@dk3n60 ~ $ ./lab11.sh -i ~/lab11.txt -C -n
Шаблон не найден
dvorlyuk@dk3n60 ~ $ ./lab11.sh -o ~/lab11-1.txt -p it -n
Файл не найден
dvorlyuk@dk3n60 ~ $ █

```

Рис. 3.4: Какие-то махинации

2. Написал 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. [3.5])(рис. [3.6])(рис. [??])



```
#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. 3.5: Прога на C



```
#!/bin/bash

gcc lab11.c -o lab11
./lab11
code=$?
case $code in
    0) echo "Число меньше 0";;
    1) echo "Число больше 0";;
    2) echo "Число равно 0";;
esac
```

Рис. 3.6: Скрипт для SH

```
dvorlyuk@dk3n60 ~ $ ./lab11c.sh
Введите число: 2
Число больше 0
dvorlyuk@dk3n60 ~ $ ./lab11c.sh
Введите число: -2
Число меньше 0
dvorlyuk@dk3n60 ~ $ 
```

Рис. 3.7: Проверка работоспособности

3. Написал 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. [??])(рис. [??])
4. Написал командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Модифицировал его так, чтобы запаковывались только те файлы, которые были изменены менее недели тому назад (использовал команду find).(рис. [??])(рис. [??])

4 Выводы

Изучил основы программирования в оболочке ОС UNIX, научился 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.