

Лабораторная работа-11

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

Бровкин Александр НБИбд-01-21

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` — 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: `-iinputfile` — прочитать данные из указанного файла; `-ooutputfile` — вывести данные в указанный файл; `-rшаблон` — 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`. (рис. 3.1)(рис. 3.2)(рис. 3.3)

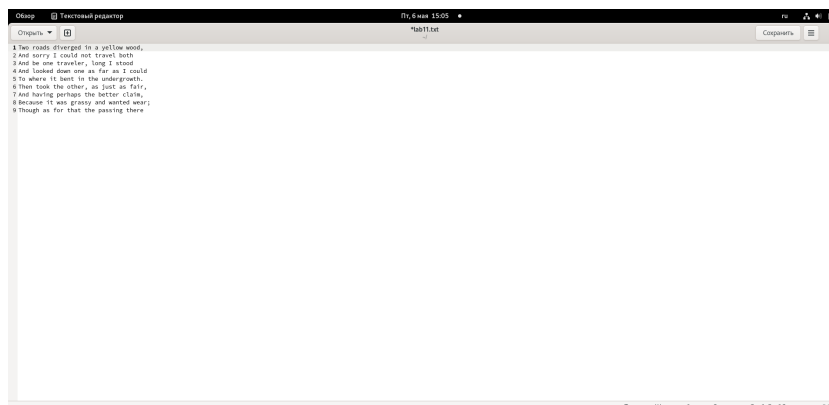


Рис. 3.1: Вставил в файл любой текст из интернета



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 оболочку. Командный файл должен вызывать эту программу и, проанализировав с помощью команды `$?`, выдать сообщение о том, как число было введено. (рис. 3.4)(рис. 3.5)(рис. 3.6)

командной строки. Этот же командный файл должен уметь удалять все созданные им файлы (если они существуют).(рис. 3.7)(рис. 3.8)

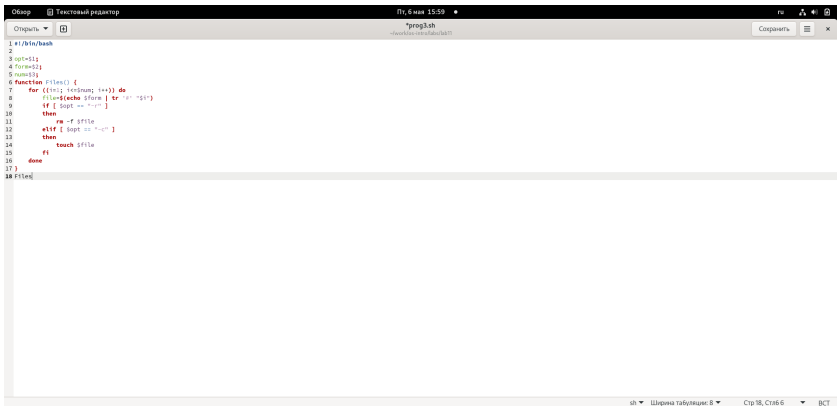


Рис. 3.7: Пишу новый скрипт

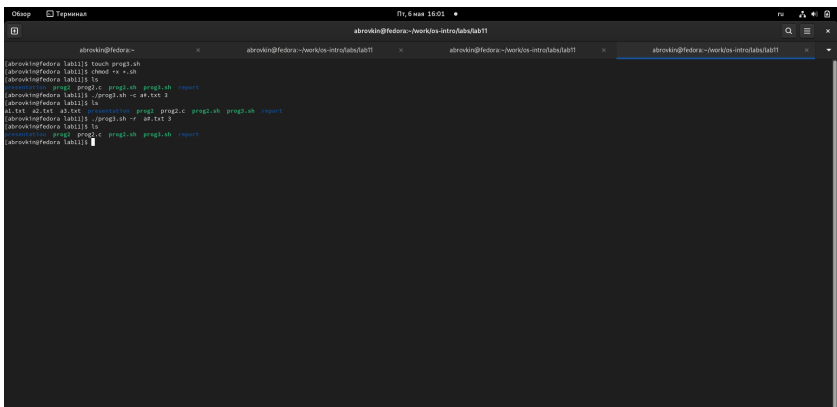
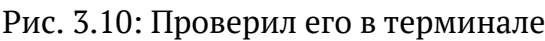


Рис. 3.8: Проверяю его в терминале

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. 3.9)(рис. 3.10)



4 Вывод:

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.