

Отчёт по лабораторной работе 11

Дисциплина: Операционные системы

Волчок Кристина Александровна

Содержание

| | | |
|----------|---------------------------------------|-----------|
| 1 | Цель работы | 4 |
| 2 | Задания | 5 |
| 3 | Выполнение лабораторной работы | 6 |
| 4 | Контрольные вопросы | 11 |
| 5 | Ответы на контрольные вопросы: | 12 |
| 6 | Вывод: | 15 |

Список иллюстраций

| | | |
|-----|---|----|
| 3.1 | Пишу первый скрипт | 6 |
| 3.2 | Проверяю в терминале | 7 |
| 3.3 | Пишу новый скрипт-на языке Си | 7 |
| 3.4 | Пишу еще один скрипт | 8 |
| 3.5 | Проверяю все в терминале | 8 |
| 3.6 | Пишу новый скрипт | 9 |
| 3.7 | Проверяю его в терминале | 9 |
| 3.8 | Пишу новый скрипт | 10 |
| 3.9 | Проверила его в терминале | 10 |

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

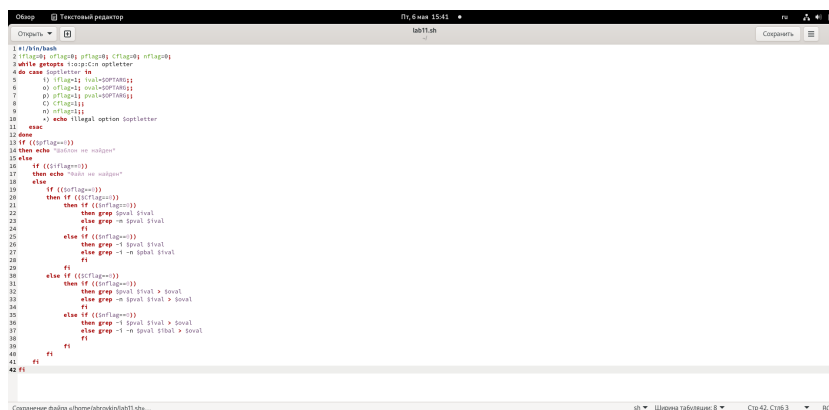
1. Используя команды `grep`, написать командный файл, который анализирует командную строку с ключами: `-i` — прочитать данные из указанного файла; `-o` — вывести данные в указанный файл; `-r` — указать шаблон для поиска; `-C` — различать большие и малые буквы; `-n` — выдавать номера строк. а затем ищет 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`, написала 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.

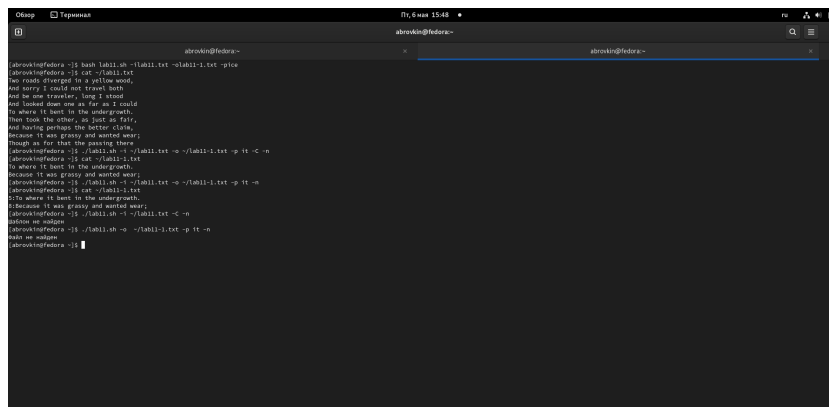
а затем ищет в указанном файле нужные строки, определяемые ключом `-r`. (рис. ??) (рис. 3.1) (рис. 3.2)

Вставил в файл любой текст из интернета

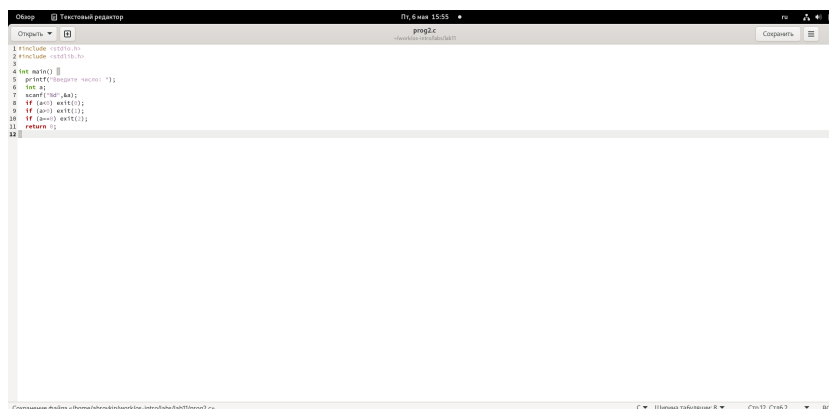


```
1 #!/bin/bash
2 if [ $# -eq 0 ]; then
3     echo "Usage: ./lab11.sh -i inputfile -o outputfile -r pattern -C case -n lines"
4     exit 1
5 fi
6 while getopts "i:o:r:C:n:" opt; do
7     case $opt in
8         i) inputfile=$OPTARG ;;
9         o) outputfile=$OPTARG ;;
10        r) pattern=$OPTARG ;;
11        C) case=$OPTARG ;;
12        n) lines=$OPTARG ;;
13        *) echo "Invalid option $OPTARG" ;;
14    esac
15 done
16 if [ $inputfile = "" ]; then
17     echo "Input file is required"
18     exit 1
19 fi
20 if [ $outputfile = "" ]; then
21     echo "Output file is required"
22     exit 1
23 fi
24 if [ $pattern = "" ]; then
25     echo "Pattern is required"
26     exit 1
27 fi
28 if [ $case = "" ]; then
29     echo "Case is required"
30     exit 1
31 fi
32 if [ $lines = "" ]; then
33     echo "Lines is required"
34     exit 1
35 fi
36 if [ $inputfile != "" ]; then
37     grep -r $pattern $inputfile -C $case -n $lines > $outputfile
38 else
39     echo "Input file is required"
40     exit 1
41 fi
```

Рис. 3.1: Пишу первый скрипт



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. 3.3)(рис. 3.4)(рис. 3.5)



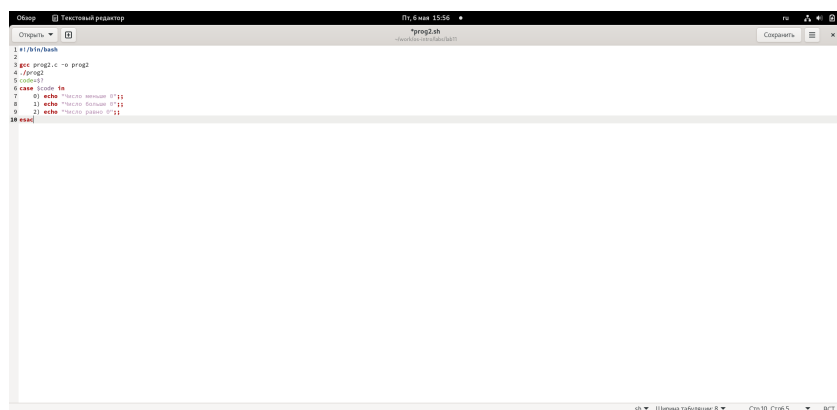


Рис. 3.4: Пишу еще один скрипт

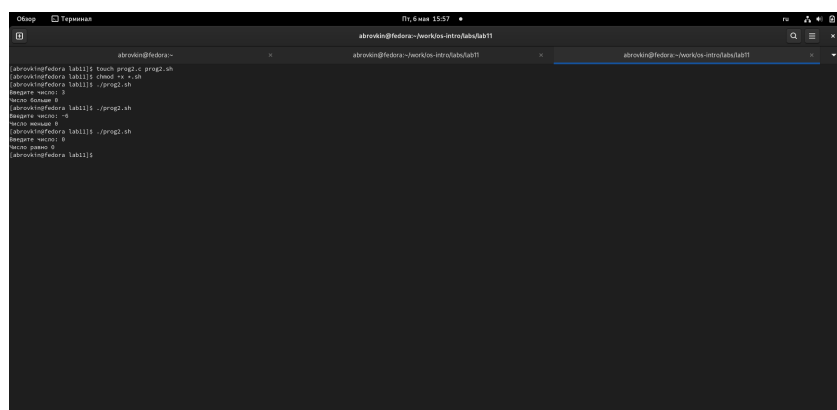


Рис. 3.5: Проверяю все в терминале

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. 3.6)(рис. 3.7)

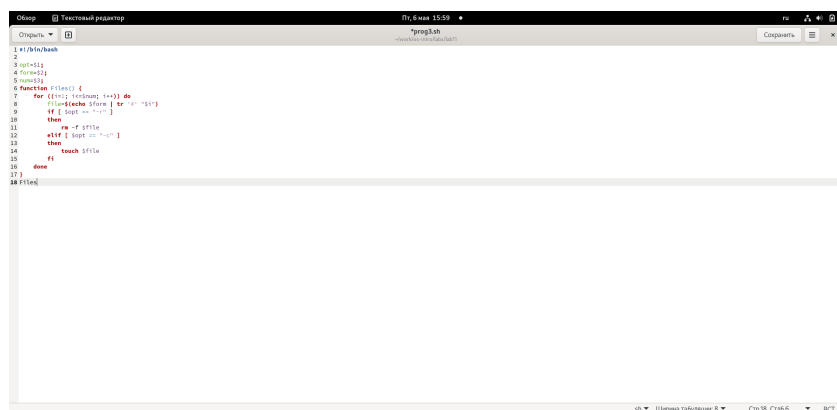


Рис. 3.6: Пишу новый скрипт

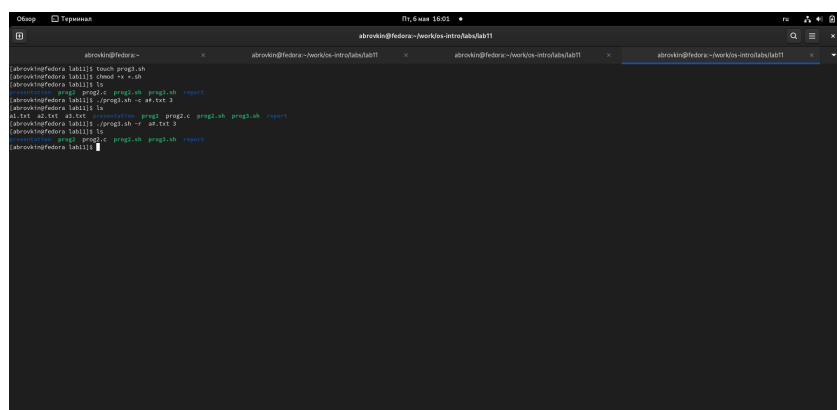


Рис. 3.7: Проверяю его в терминале

- 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. 3.8) (рис. 3.9)

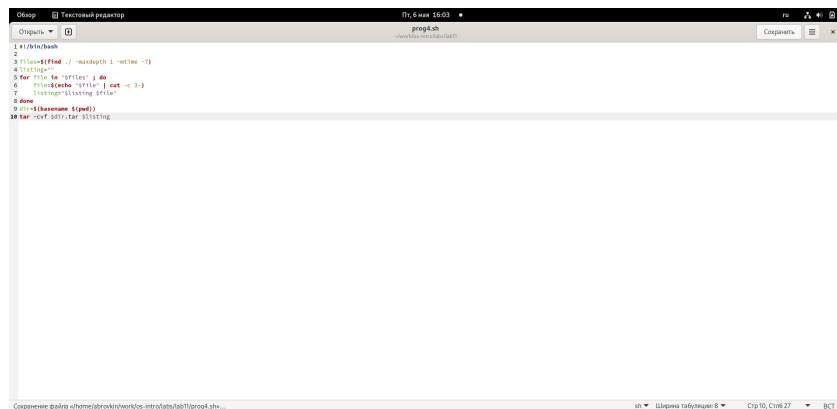


Рис. 3.8: Пишу новый скрипт

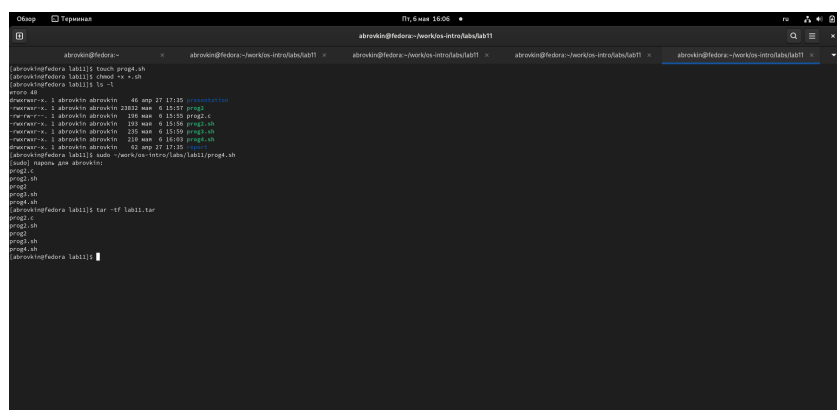


Рис. 3.9: Проверила его в терминале

4 Контрольные вопросы

1. Каково предназначение команды `getopts`?
2. Какое отношение метасимволы имеют к генерации имён файлов? 98 Лабораторная работа № 11. Программирование в командном процессоре ОС UNIX. ...
3. Какие операторы управления действиями вы знаете?
4. Какие операторы используются для прерывания цикла?
5. Для чего нужны команды `false` и `true`?
6. Что означает строка `if test -f mans/i.$s`, встреченная в командном файле?
7. Объясните различия между конструкциями `while` и `until`.

5 Ответы на контрольные вопросы:

- 1) Команда `getopts` осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных. Синтаксис команды следующий: `getopts option-string variable [arg ...]` Флаги – это опции командной строки, обычно помеченные знаком минус; Например, для команды `ls` флагом может являться `-F`. Строка опций `option-string` – это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за символом, обозначающим этот флаг, должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда `getopts` может распознать аргумент, то она возвращает истину. Принято включать `getopts` в цикл `while` и анализировать введенные данные с помощью оператора `case`. Функция `getopts` включает две специальные переменные среды – `OPTARG` и `OPTIND`. Если ожидается дополнительное значение, то `OPTARG` устанавливается в значение этого аргумента. Функция `getopts` также понимает переменные типа массив, следовательно, можно использовать её в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.
- 2) При перечислении имён файлов текущего каталога можно использовать следующие символы:
 - `-` – соответствует произвольной, в том числе и пустой строке;
 - `?` – соответствует любому одинарному символу;

- `[c1-c2]` – соответствует любому символу, лексикографически находящемуся между символами `c1` и `c2`. Например,
 - `echo *` – выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды `ls`;
 - `ls *.c` – выведет все файлы с последними двумя символами, совпадающими с `.c`.
 - `echo prog.?` – выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются `prog..`
 - `[a-z]*` – соответствует произвольному имени файла в текущем каталоге, начинающемуся с любой строчной буквы латинского алфавита.
- 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.
- 4) Два несложных способа позволяют вам прерывать циклы в оболочке `bash`. Команда `break` завершает выполнение цикла, а команда `continue` завершает

данную итерацию блока операторов. Команда `break` полезна для завершения цикла `while` в ситуациях, когда условие перестаёт быть правильным. Команда `continue` используется в ситуациях, когда больше нет необходимости выполнять блок операторов, но вы можете захотеть продолжить проверять данный блок на других условных выражениях.

- 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 код завершения (истина), выполняется последовательность команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `do`, после чего осуществляется безусловный переход на начало оператора цикла `while`. Выход из цикла будет осуществлён тогда, когда последняя выполненная команда из последовательности команд (операторов), которую задаёт список-команд в строке, содержащей служебное слово `while`, возвратит ненулевой код завершения (ложь). При замене в операторе цикла `while` служебного слова `while` на `until` условие, при выполнении которого осуществляется выход из цикла, меняется на противоположное. В остальном оператор цикла `while` и оператор цикла `until` идентичны.

6 Вывод:

В ходе выполнения лабораторной работы я изучила основы программирования в оболочке ОС UNIX, научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.