Лабораторная работа № 11.

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

ОЗЬЯС Стев Икнэль Дани

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

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 getopts grep, написать командный файл, который анализирует командную строку с ключами:
    - -iinputfile — прочитать данные из указанного файла;
    - -ooutputfile — вывести данные в указанный файл;
    - -pшаблон — указать шаблон для поиска;
    - — 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 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.
3. Написать командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до ⊠ (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).

3 Теоретическое введение

- Командный файл (скрипт или сценарий) - это текстовый файл, состоящий из команд интерпретатора. При запуске этого файла последовательно выполняются все команды, содержащиеся в нем.
- Для создания командного файла:
  - Запустите текстовый редактор.
  - Последовательно запишите команды, располагая каждую команду на отдельной строке.
  - Сохраните этот файл, сделайте его исполняемым, применив команду:
    1. chmod +x имя_файла.
  - Запустите созданный файл и проверьте правильность выполнения команд.
  - В случае нахождения ошибки, в текстовом редакторе внесите изменения в командный файл, сохраните его и проверьте еще раз.
- Более подробно об Unix см. в [1–6].

4 Выполнение лабораторной работы

4.1 Задание 1

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;
   - -pшаблон — указать шаблон для поиска;
   - — различать большие и малые буквы;
   - -n — выдавать номера строк. а затем ищет в указанном файле нужные строки, определяемые ключом -p. (рис. 4.1)

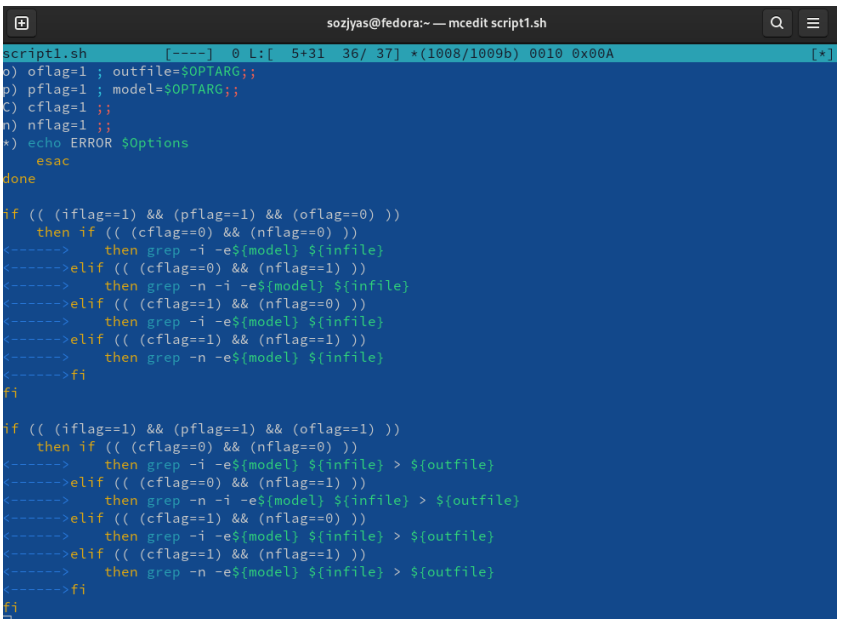

Рис. 4.1: Командный файл №1

- Выполнил его. (рис. 4.2)

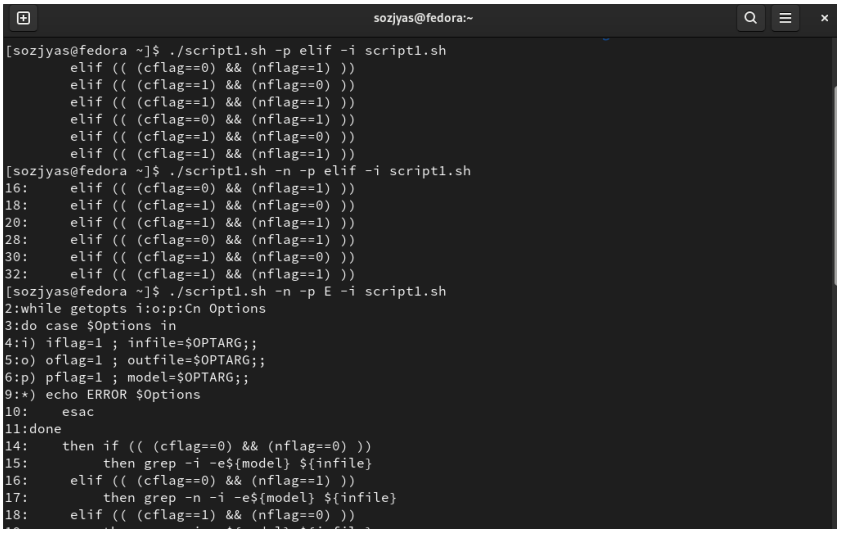

Рис. 4.2: Выполнение командного файла №1

- Часть 2 (рис. 4.3)

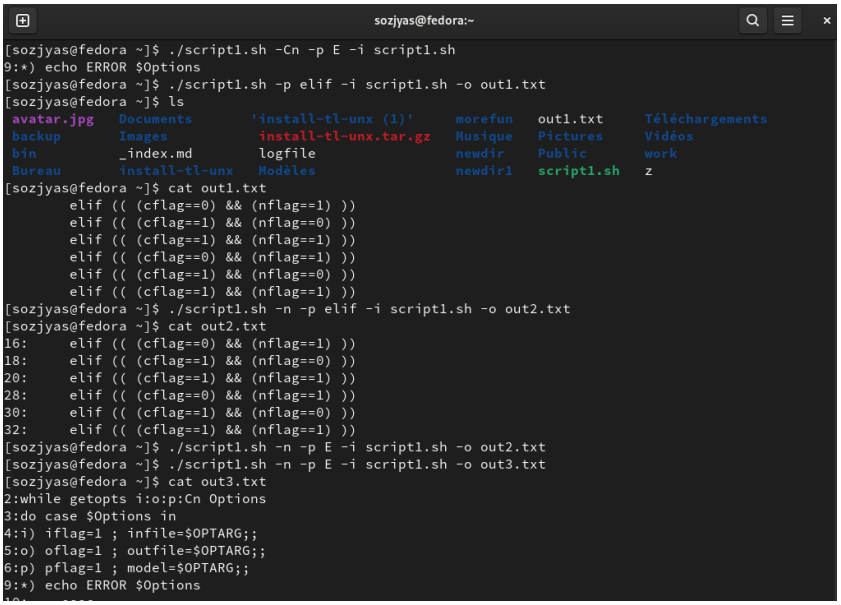

Рис. 4.3: Выполнение командного файла №1

4.2 Задание 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. (рис. 4.4)

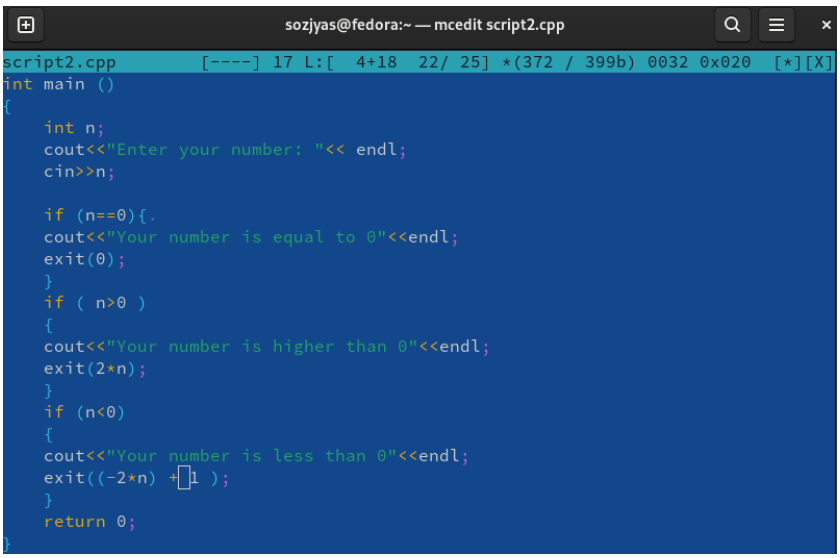

Рис. 4.4: Программа на языке Си

- Написал командный файл, который вызывает эту программу и, проанализировав с помощью команды $?, выдает сообщение о том, какое число было введено. (рис. 4.5)

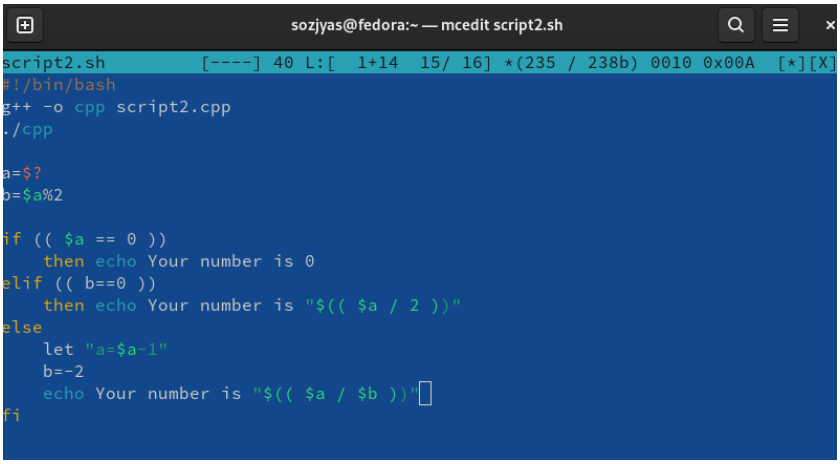

Рис. 4.5: Командный файл №2

- Выполнил командный файл (рис. 4.6)

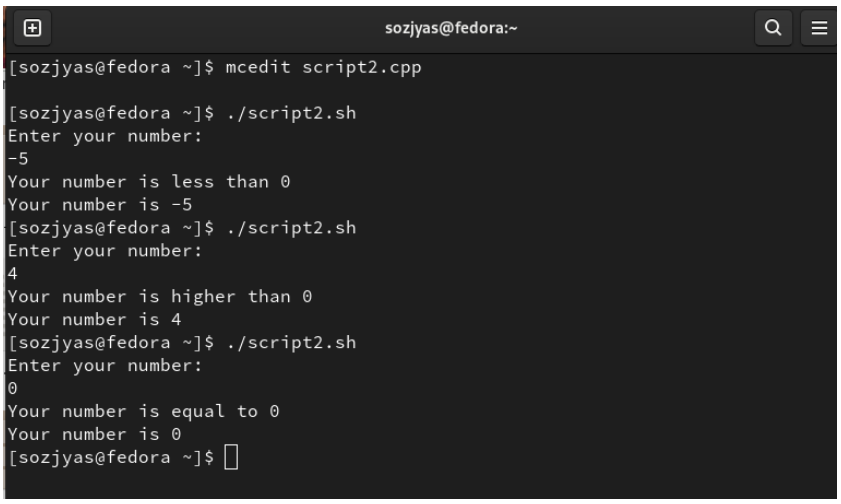

Рис. 4.6: Выполнение командного файла №2

4.3 Задание 3

- Написал командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до ☒ (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 умеет удалять все созданные им файлы (если они существуют). (рис. 4.7)

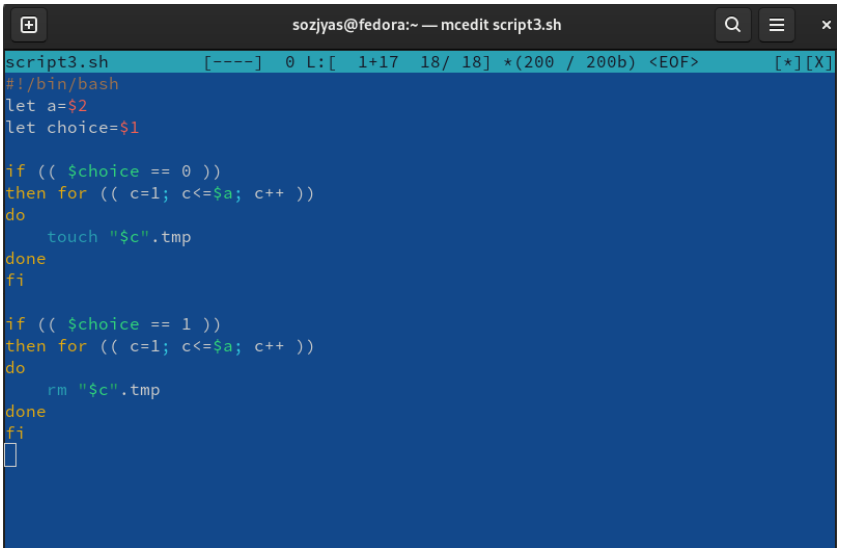

Рис. 4.7: Командный файл №3

- Выполнил командный файл (рис. 4.8)

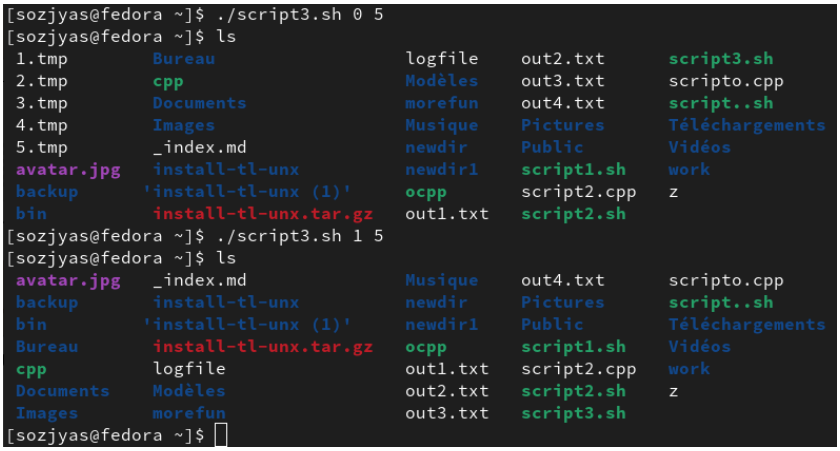

Рис. 4.8: Выполнение командного файла №2

4.4 Задание 4

- Написал 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). (рис. 4.9)

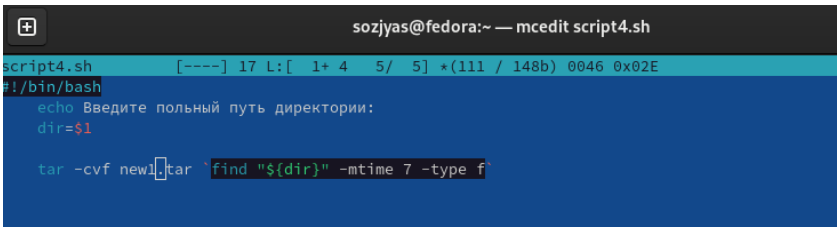

Рис. 4.9: Командный файл №4

- Выполнил командный файл (рис. 4.10)

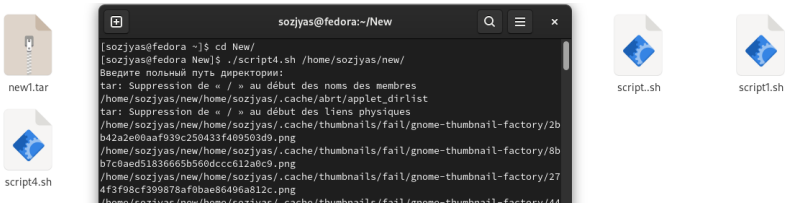

Рис. 4.10: Выполнение командного файла №2

5 Выводы

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

6 Контрольные вопросы

1. Весьма необходимой при программировании является команда getopts, которая 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, -F является флагом для команды ls -F. Иногда эти флаги имеют аргументы, связанные с ними. Программы интерпретируют эти флаги, соответствующим образом изменяя свое поведение. Строка опций option-string — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за этой буквой должно следовать двоеточие. Соответствующей переменной присваивается буква данной опции. Если команда getopts может распознать аргумент, она возвращает истину. Принято включать getopts в цикл while и анализировать введенные данные с помощью оператора case. Предположим, необходимо распознать командную строку следующего формата: testprog -ifile_in.txt -ofile_out.doc -L -t -r Вот как выглядит использование оператора getopts в этом случае: while getopts o:i:LtOPTLETer do case $optletterino) oflag = 1; oval =$OPTARG;; i) iflag=1; ival=$OPTARG;; L) Lflag=1;; t) tflag=1;; r) rflag=1;; *) echo Illegal option $optletter esac done Функция getopts включает две специальные переменные среды – OPTARG и OPTIND. Если ожидается дополнительное значение, то OPTARG устанавливается в значение этого аргумента (будет равна file_in.txt для опции i и file_out.doc для опции o) . OPTIND является

числовым индексом на упомянутый аргумент. Функция getopts также понимает переменные типа массив, следовательно, можно использовать ее в функции не только для синтаксического анализа аргументов функций, но и для анализа введенных пользователем данных.

2. При перечислении имен файлов текущего каталога можно использовать следующие символы: — соответствует произвольной, в том числе и пустой строке; ? — соответствует любому одному символу; [c1-c1] — соответствует любому символу, лексикографически на ходящемуся между символами c1 и c2. echo * — выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды ls; ls *.c — выведет все файлы с последними двумя символами, равными .c. echo prog.? — выдаст 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 Вам 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 дела являются операторами языка программирования bash. Поэтому при описании языка программирования bash термин оператор будет использоваться наравне с термином команда.

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 перестает быть правильным. Пример бесконечного цикла while, с прерыванием в момент, когда файл перестает существовать: while true do if [! -f $file] then break fi sleep 10 done

5. 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.

6. Введенная строка означает условие существования файла man$s$/i.$s

7. Если речь идет о 2-х параллельных действиях, то это while. когда мы показываем, что сначала делается 1-е действие. потом оно заканчивается при наступлении 2-го действия, применяем until

Список литературы

1. GNU Bash Manual [Электронный ресурс]. Free Software Foundation, 2016. URL: https://www.gnu.org/software/bash/manual/.
2. Newham C. Learning the bash Shell: Unix Shell Programming. O'Reilly Media, 2005. 354 с.
3. Zarrelli G. Mastering Bash. Packt Publishing, 2017. 502 с.
4. Robbins A. Bash Pocket Reference. O'Reilly Media, 2016. 156 с.
5. Таненбаум Э. Архитектура компьютера. 6-е изд. СПб.: Питер, 2013. 874 с.
6. Таненбаум Э., Бос Х. Современные операционные системы. 4-е изд. СПб.: Питер, 2015. 1120 с.