Лабораторная работа № 11

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Шулуужук Айраана Вячеславовна НПИбд-02-22

Содержание

# 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 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; – -C — различать большие и малые буквы; – -n — выдавать номера строк.

а затем ищет в указанном файле нужные строки, определяемые ключом -p.

1. 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. Команд- ный файл должен вызывать эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено.
2. 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 ко- мандный файл должен уметь удалять все созданные им файлы (если они существуют).
3. 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).

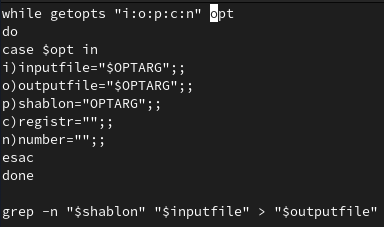
# 3 Теоретическое введение

## 3.1 Командные процессоры (оболочки)

Командный процессор (командная оболочка, интерпретатор команд shell) — это про- грамма, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: – оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций; – С-оболочка (или csh) — надстройка на оболочкой Борна, использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд; – оболочка Корна (или ksh) — напоминает оболочку С, но операторы управления програм- мой совместимы с операторами оболочки Борна; – BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей сов- мещает свойства оболочек С и Корна (разработка компании Free Software Foundation). POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных опера- ционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна. Рассмотрим основные элементы программирования в оболочке bash. В других оболоч- ках большинство команд будет совпадать с описанными ниже.

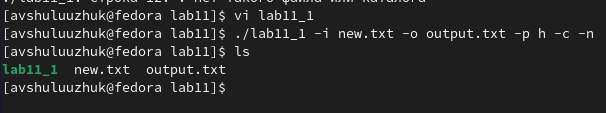
# 4 Выполнение лабораторной работы

Используя команды getopts grep, напишем командный файл, который анализирует командную строку с ключами (рис. ??)



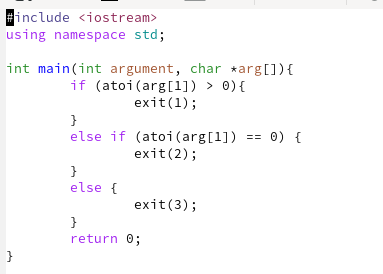
скрипт 1

Скомпилируем данные файл и проверим его работу (рис. ??)



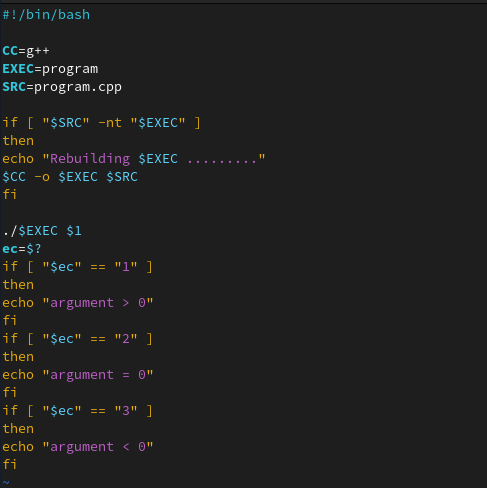
результат работы командного фпйла

Напишем на языке Си программу, которая вводит число и определяет, является ли оно больше нуля, меньше нуля или равно нулю (рис. ??)



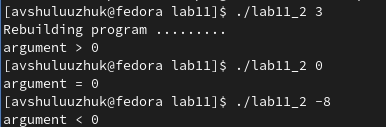
программа сравнения чисел на С

Нашишем командный файл который должен вызывать эту программу и, проанализировав с помощью команды $?, выдать сообщение о том, какое число было введено.(рис. ??)



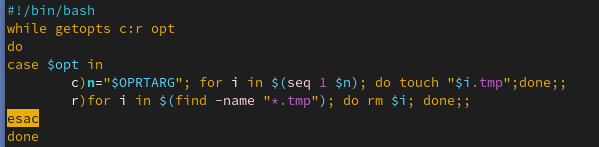
скрипт 2

Проверим работу командного файла, используя число для сравнения в качестве аргумента (рис. ??)



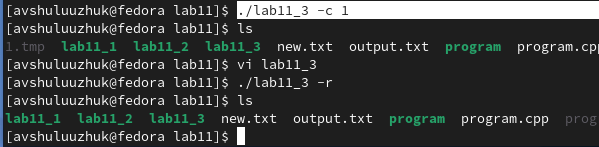
результат запуска скрипта 2

Создаем новый файл lab11\_3 для скрипта 3. Напишем командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до n и удаляющий все созданные им файлы (рис. ??)



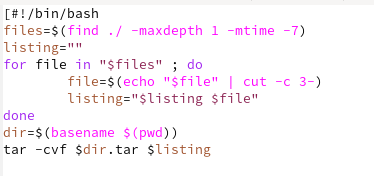
скрипт lab11\_3

Запустим этот файл и создадим файл tmp, а также сразу удалим этот файл (рис. ??)



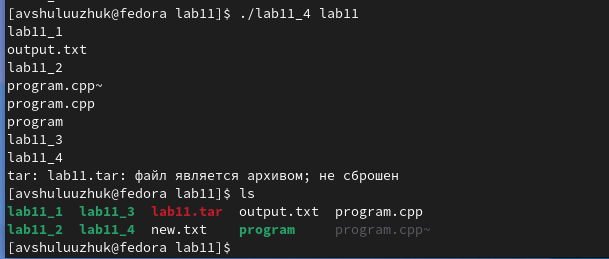
результат запуска скрипта 3

В файле lab11\_4 напишем скрипт, который с помощью команды tar запаковывает в архив все файлы в указанной директории. (рис. ??)



скрипт lab11\_4

Запустим файл и запакуем архив с файлами каталога lab11 (рис. ??)



результат запуска скрипта 4

# 5 Выводы

В ходе выполнения работы мы изучили основы программирования в оболочке ОС UNIX/Linux и научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.