Лабораторная работа 11

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Генералов Даниил, НПИ-01-21, 1032212280

Содержание

# 1 Цель работы

Целью данной работы является:

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

Требуется написать 4 командных файла:

* принимает аргументы с командной строки и использует их для запуска grep
* запускает другую программу и получает ее код выхода
* создает файлы с названиями, имеющими номер в последовательности
* создает архив из файлов, которые обновлены меньше чем неделю назад

# 2 Теоретическое введение

Командный процессор (*shell*) – это программа на Unix-системах, которая принимает ввод от пользователя и исполняет инструкции. Помимо интерактивного использования, она может исполнять список команд, заданный в файле, и она обладает набором команд, достаточным для написания программ разной степени сложности. В этой работе мы продолжаем рассматриваьб этот функционал командного процессора, составляя несколько командных файлов, выполняющих определенные действия.

# 3 Выполнение лабораторной работы

Первая программа представлена на рис. 1. Она принимает несколько аргументов с помощью getopts и запоминает их значения. После этого, если все значения правильные, они подставляются в команду grep, которая читает один файл, ищет там строки по заданному шаблону и выводит их в файл. Это показано на рис. 2.

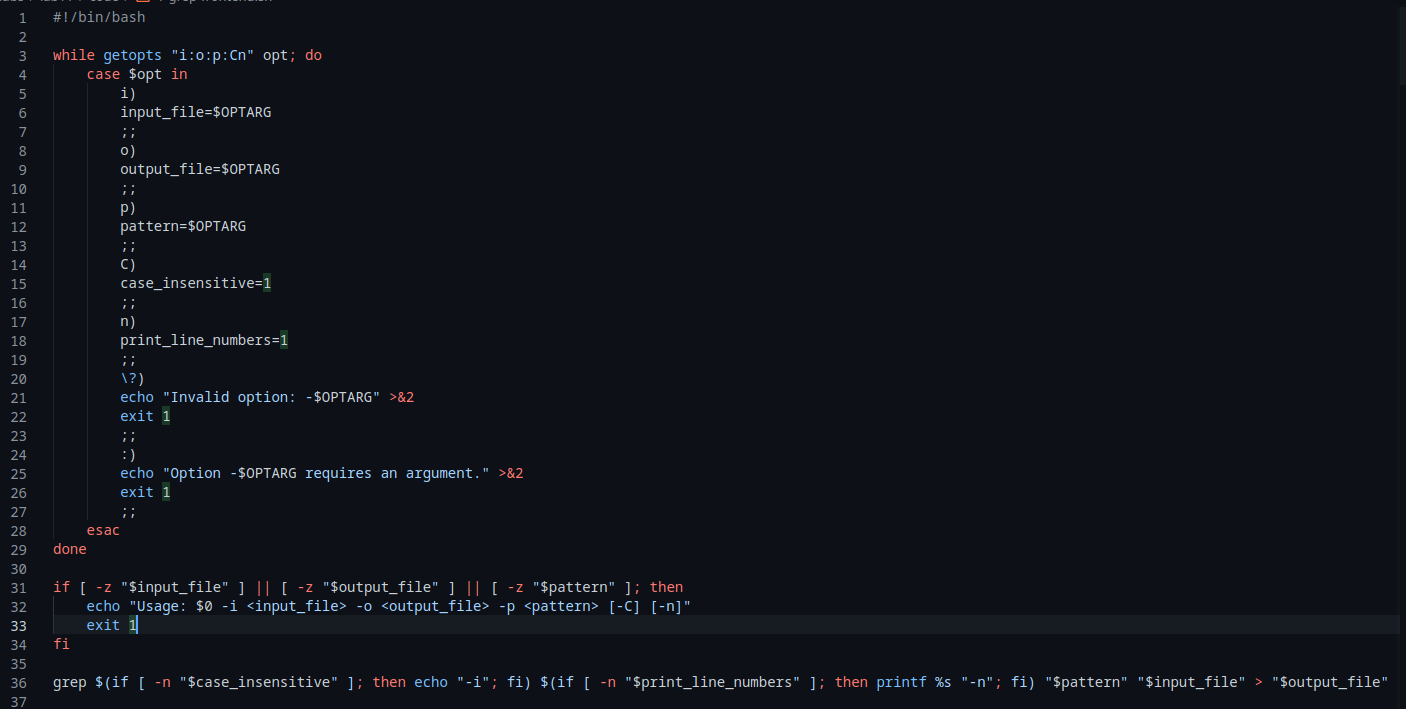


Рис. 1: Программа поиска

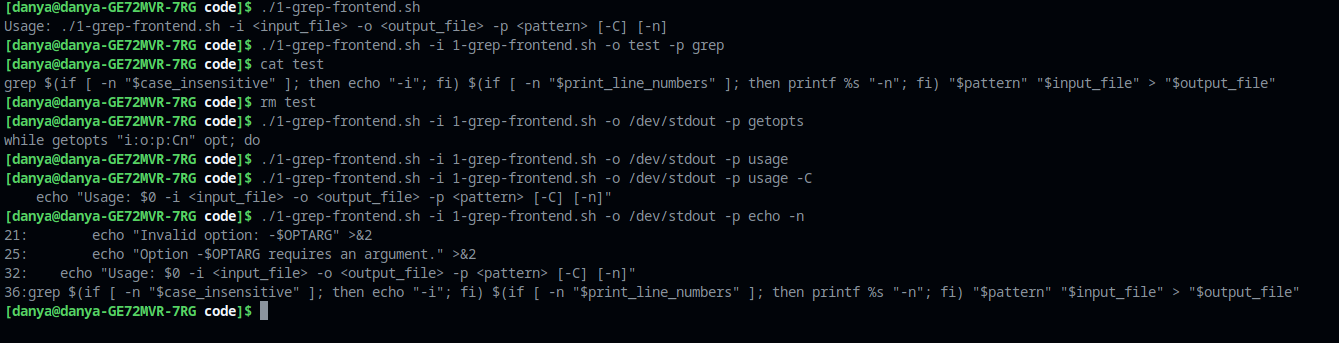


Рис. 2: Результат работы программы поиска

Вторая программа представлена на рис. 3. Она состоит из двух частей – кода на языке Си и командного файла. Код на Си принимает на ввод одно число и затем завершается с кодом возврата, соответствующим отношению этого числа к нулю. Командный файл пишет приглашение, запускает эту программу, затем считывает код возврата и использует его внутри case. В итоге получается вывод, как на рис. 4.

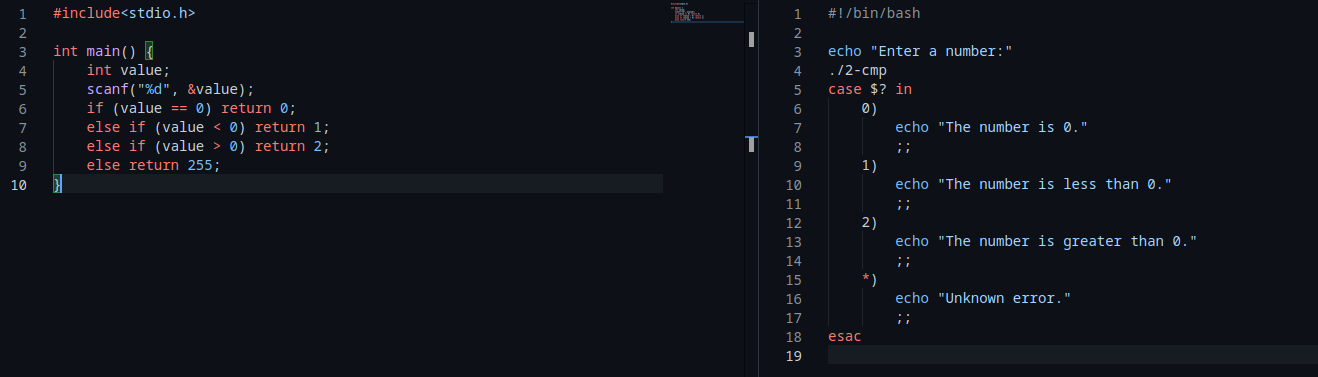


Рис. 3: Программа сравнения чисел

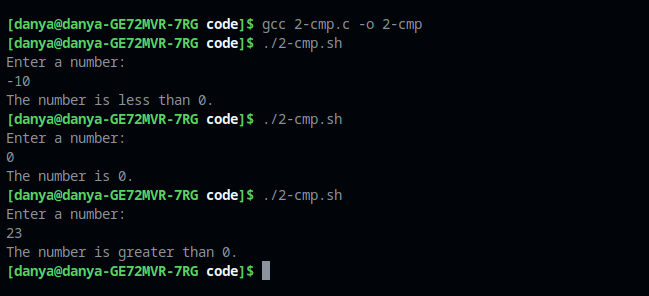


Рис. 4: Результат работы программы сравнения чисел

Третья программа представлена на рис. 5. Сначала она удаляет все файлы с названием \*.tmp. После этого она создает несколько таких файлов, где каждый файл имеет имя <номер файла>.tmp. Проверка того, как это работает, представлена на рис. 6.

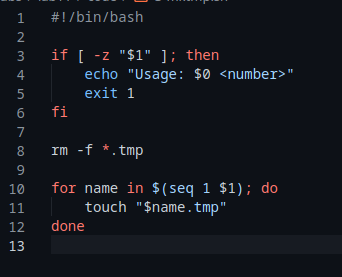


Рис. 5: Программа создания файлов

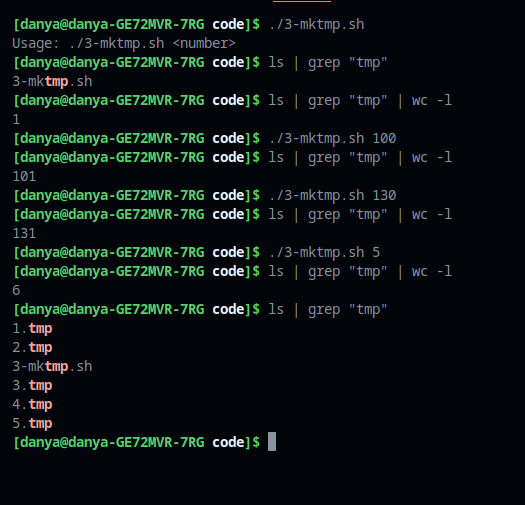


Рис. 6: Результат работы программы создания файлов

Четвертая программа представлена на рис. 7. Сначала она удаляет файл archived.tar, чтобы начать работу с новым файлом. После этого она ищет файлы в указанном каталоге, у которых дата изменения меньше семи дней назад. Каждый такой файл добавляется в архив archived.tar, или, если этот архив еще не создан, то он создается. Проверка того, как это работает, представлена на рис. 8.

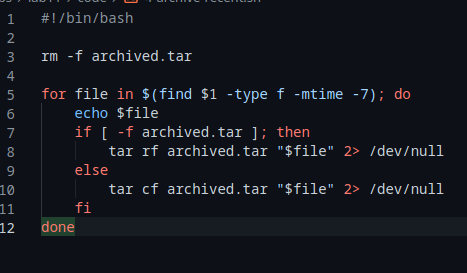


Рис. 7: Программа архивирования

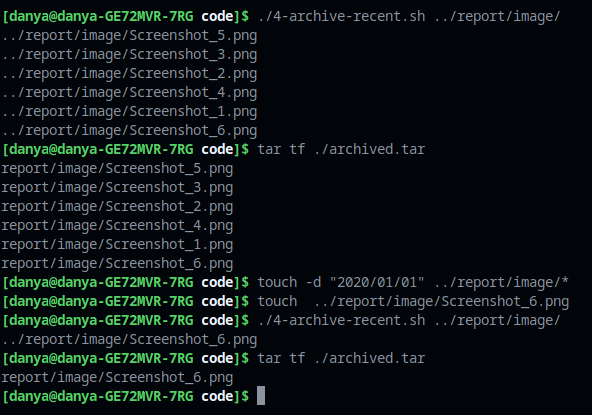


Рис. 8: Результат работы программы архивирования

# 4 Контрольные вопросы

1. Каково предназначение команды getopts?

Эта команда изменяет состояние переменных окружения, чтобы обрабатывать аргументы с командной строки. Каждый запуск этой программы поглощает один аргумент из списка и добавляет его значение в выбранную переменную окружения. Оттуда различная пользовательская логика может извлекать эти значения и обрабатывать их дальше.

1. Какое отношение метасимволы имеют к генерации имён файлов?

Метасимволы, если не изолированы через литеральные строки или обратные слешы, обрабатываются оболочкой как список строк, равных именам файлов, которым соответствует последовательность с метасимволами. Например, если задана команда ls test/\*, и в каталоге test есть файлы foo.txt, bar.txt, baz.txt и директория quux, то оболочка раскроет эту строку как ls test/foo.txt test/bar.txt test/baz.txt test/quux.

1. Какие операторы управления действиями вы знаете?

В Bash используются следующие конструкции для управления потоком выполнения команд:

- `if/then/else/fi`  
- `case/switch/default/esac`  
- `while/do/done`  
- `until/do/done`  
- `for/in/done`

1. Какие операторы используются для прерывания цикла?

Следующие операторы могут прервать текущий цикл:

- `break` продолжит выполнение после конца блока цикла  
- `continue` продолжит выполнение со следующей итерации цикла  
- `return` прервет выполнение всей функции или программы

1. Для чего нужны команды false и true?

Эти команды всегда возвращают одно и тоже выходное значение и больше ничего не делают, что делает их удобными для подстановки конкретных значений в операторы. Например, бесконечный цикл начинается с while true. Помимо этого эти программы можно использовать в ситуациях, где нужно предоставить путь к программе, но эта программа должна не делать ничего. Например, чтобы выключить доступ к оболочке для какого-то пользователя, можно указать его оболочку как /bin/false.

1. Что означает строка if test -f mani.$s, встреченная в командном файле?

Эта строка начинает блок кода, который будет выполнен только если существует файл с названием man$s/$i.$s, где $s и $i – переменные.

1. Объясните различия между конструкциями while и until.

Эти два оператора создают циклы, но цикл while выполняется, пока выражение истинно, а until выполняется, пока выражение ложно. Таким образом, while <условие> – это аналог until !<условие>.

# 5 Выводы

В этой лабораторной работе мы познакомились с основами программирования в Bash. Поскольку командный интерпретатор есть всегда, полезно уметь использовать его возможности не только в интерактивном режиме, но и для написания программ. Те четыре программы, которые мы написали, каждая демонстрируют один из важных компонентов работы с командным интерпретатором.