Отчёт по лабораторной работе №11

Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Щербак Маргарита Романовна

2022

# 1 Цель работы:

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

## 1.1 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;  
   – -p 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 -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 выдавать 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 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).

# 2 Теоретическое введение:

*Командные процессоры (оболочки)*  
Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, 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 программой совместимы с операторами оболочки Борна; - BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает 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 операционных 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 оболочках большинство команд будет совпадать с описанными ниже.  
Оболочка bash поддерживает встроенные арифметические функции. Команда let является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению. Простейшее выражение — это единичный терм (term), обычно целочисленный.

# 3 Выполнение лабораторной работы:

1. **Первое задание.** Для выполнения первого задания я создала файл prog1.sh, в котором буду писать скрипт, открыла текстовой редактор emacs. Также дала созданному файлу право доступа на выполнение (+х).  
   Считывать данные я буду из файла conf.txt, поэтому просмотрели его содержимое перед выполнением задания.  
   Создала файл outputf.txt, в который считывала содержимое файла conf.txt.  
   При выполнении задания взяла в качестве шаблона из файла conf.txt строку №23: man\_db.conf, “db” будет служить шаблоном для параметра -С.  
   (Рис. 1 - Рис. 5).

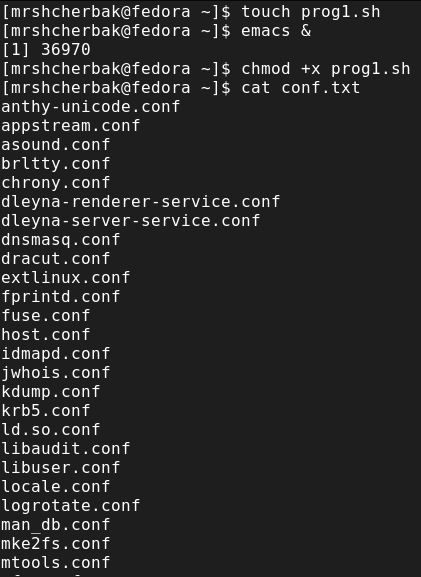


Рис. 1: Создание файла, в котором написан скрипт первого задания, предоставление права доступа на выполнение этому файлу и просмотр содержимого файла, который будем читать.

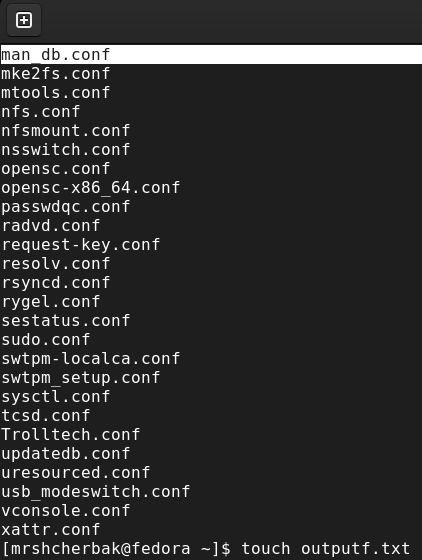


Рис. 2: Создание файла, в который будем выводить данные из указанного файла.

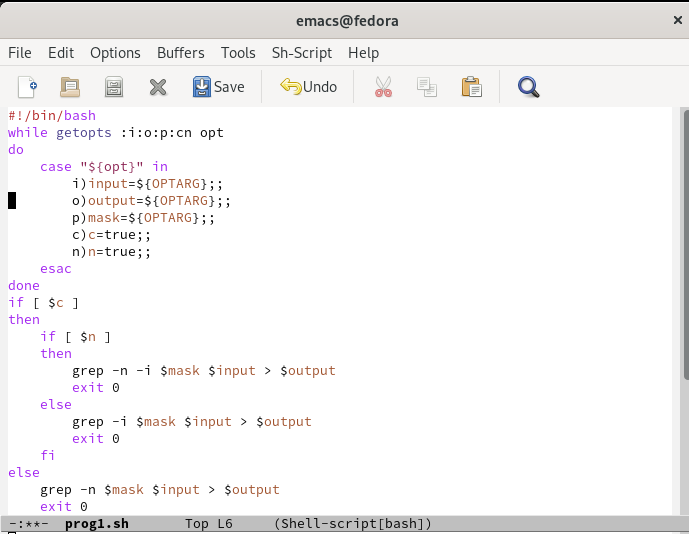


Рис. 3: Скрипт 1го задания

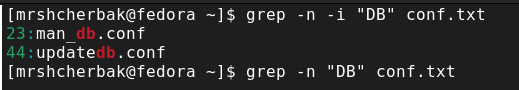


Рис. 4: Выполнение

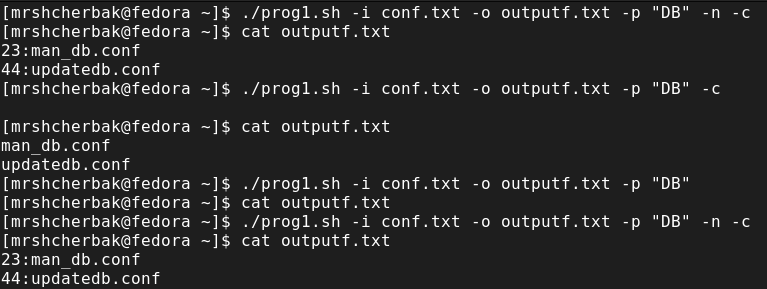


Рис. 5: Проверка работы скрипта с параметрами (ключами)

Делаем вывод, что скрипт работает корректно, используя команды getopts grep, я написала командный файл, который анализирует командную строку с ключами и ищет в указанном файле нужные строки, определяемые ключом -p.

1. **Второе задание.** Создала файл prog2.sh, в котором писала второй скрипт, и открыла его в редакторе emacs. Также создала файл prog2.c, в котором писала на языке Си программу, выводящую число и определяющую, является ли оно больше нуля, меньше нуля или равно нулю.  
   Предоставила право доступа на выполнение файлу prog2.sh.  
   (Рис. 6 - Рис. 9).

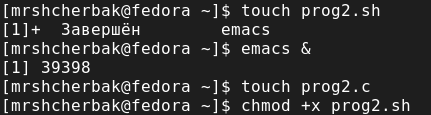


Рис. 6: Создание файла, в котором написан скрипт второго задания, предоставление права доступа на выполнение этому файлу

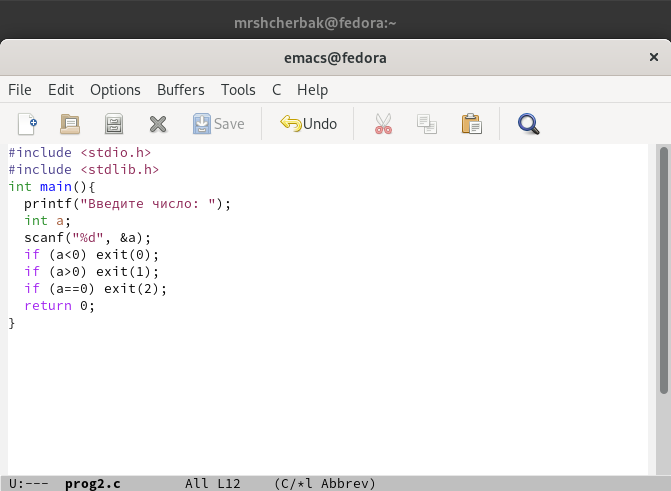


Рис. 7: Программа на языке Си

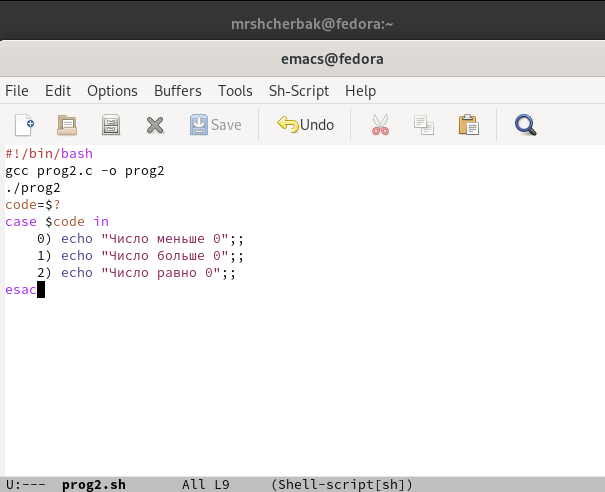


Рис. 8: Скрипт 2го задания

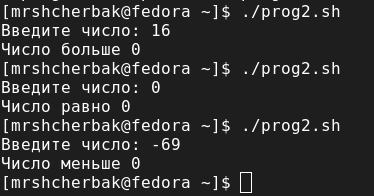


Рис. 9: Проверка выполнения скрипта

Таким образом, мы видим, что задание выполнено успешно.

1. **Третье задание.** Создала файл prog3.sh, в котором писала третий скрипт, и открыла его в редакторе emacs (C-x C-s). Предоставила право доступа на выполнение файлу prog3.sh. (Рис. 11).  
   Написала командный файл, создающий указанное число файлов, пронумерованных последовательно от 1 до N (1.tmp, 2.tmp, 3.tmp,4.tmp и т.д.). Этот же командный файл удаляет все созданные им файлы (Рис. 10).

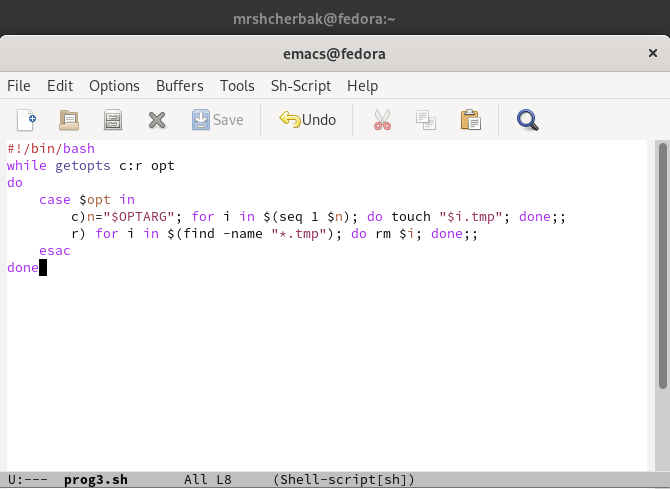


Рис. 10: Скрипт 3го задания

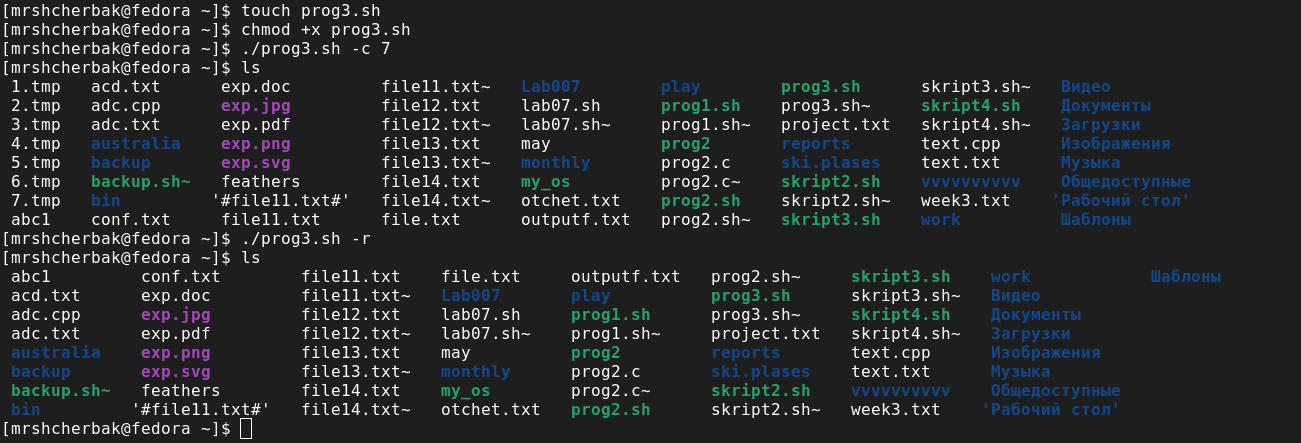


Рис. 11: Проверка выполнения скрипта

Скрипт работает корректно.

1. **Четвёртое задание.** Создала файл prog4.sh, в котором писала четвёртый скрипт, и открыла его в редакторе emacs (C-x C-s). Предоставила право доступа на выполнение файлу prog4.sh. (Рис. 13).  
   Написала командный файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории. Запаковываются те файлы, которые были изменены менее недели тому назад. (Рис. 12).

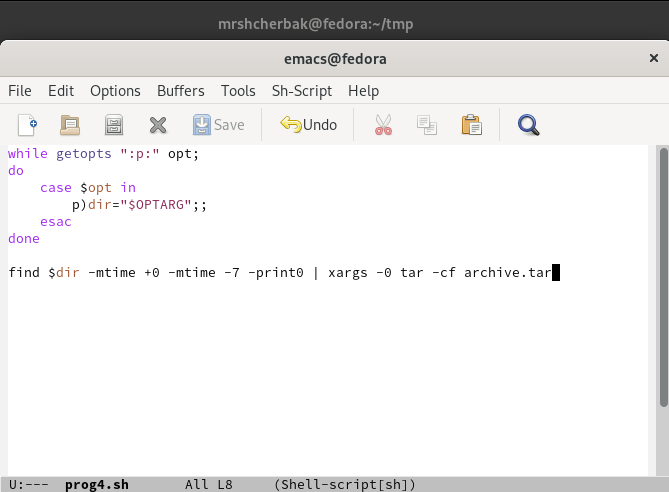


Рис. 12: Скрипт 4го задания

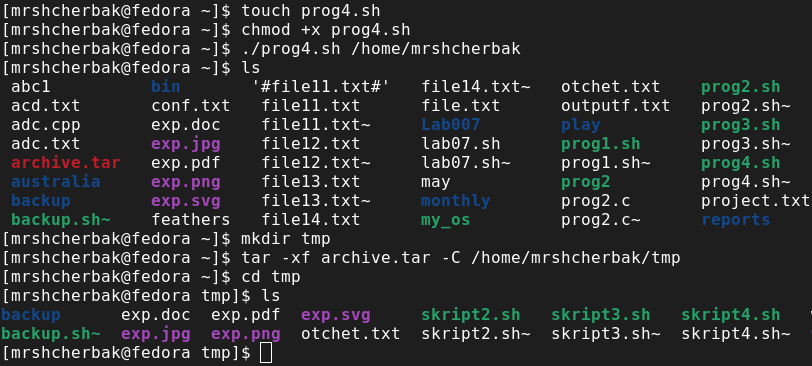


Рис. 13: Проверка работы скрипта

В папке tmp, созданной в домашнем каталоге, находятся файлы, изменённые менее недели тому назад.

# 4 *Контрольные вопросы:*

1. Команда getopts осуществляет синтаксический анализ командной строки, выделяя флаги, ииспользуется 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 − эт осписок 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. Еслик оманда getopts может распознать аргумент, то она возвращает истину. Принято включать getopts в цикл while и анализировать введённые данные с помощью оператора case. Функция getopts включает две специальные переменные среды −OPTARG и OPTIND. Если ожидается доплнительное 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 введённых пользователем данных.
2. При перечислении имён файлов текущего каталога можно использовать следующие символы:  
   1.−соответствует произвольной, в том числе и пустой строке;  
   2.?−соответствует любому одинарному символу;  
   3.[c1-c2] − соответствует любому символу, лексикографически находящемуся между символами с1 и с2.  
   Например, 1.1 echo − выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды ls; 1.2. ls.c−выведет все файлы с последними двумя символами, совпадающими с.c. 1.3. echoprog.?−выведет все файлы, состоящие из пяти или шести символов, первыми пятью символами которых являются prog.. 1.4.[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 иwhile. 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 ОСUNIX 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. 16 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 ОСUNIX 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,неравный нулю(т.е.ложь).  
   Примеры бесконечных циклов: while true do echo hello andy done until false do echo hello mike done.
6. Строка if test-fmans/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.

# 5 Выводы

Таким образом, в ходе ЛР№11 я изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.