Лабораторная работа №13

Операционные системы

Тойчубекова Асель Нурлановна

Содержание

# 1 Цель работы

Целью данной лабораторной работы является изучить основы программирования в оболочке ОС UNIX. Также научиться 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. Командный 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 командный 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

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.

# 4 Выполнение лабораторной работы

Для начало я создаю первый файл, в котором буду писать программу и открою его в редакторе gedit. (рис. 1).

Создание файла

Рис. 1: Создание файла

Редактирую файл, пишу командный файл, который используя команды getopts grep анализирует командную строку с некоторыми ключами. Используя цикл while программа анализирует все флаги, запичывая данные в нужные файлы. В конце проверяются использования опций c и n и присваиваются определенным переменным. Осуществляется команда grep, которая в данном случае берет и записывает в новый файл текст который совпал с шаблонным. (рис. 2).

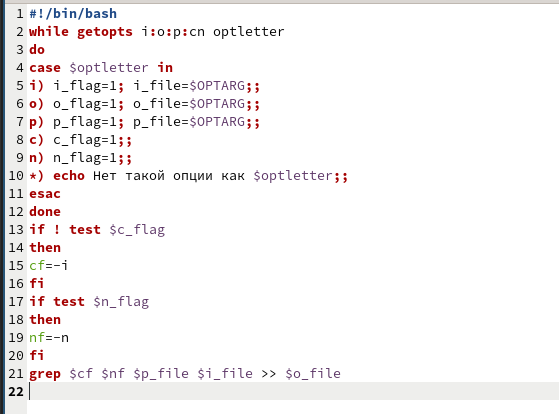


Рис. 2: Редактирование файла

Даю права на выполнение и запускаю программу. (рис. 3).

Запуск программы

Рис. 3: Запуск программы

Мы видим, что программа удачно сработала и строки с шаблоном “Черный” из текста в файле output.txt записался в файл input.txt. (рис. 4).

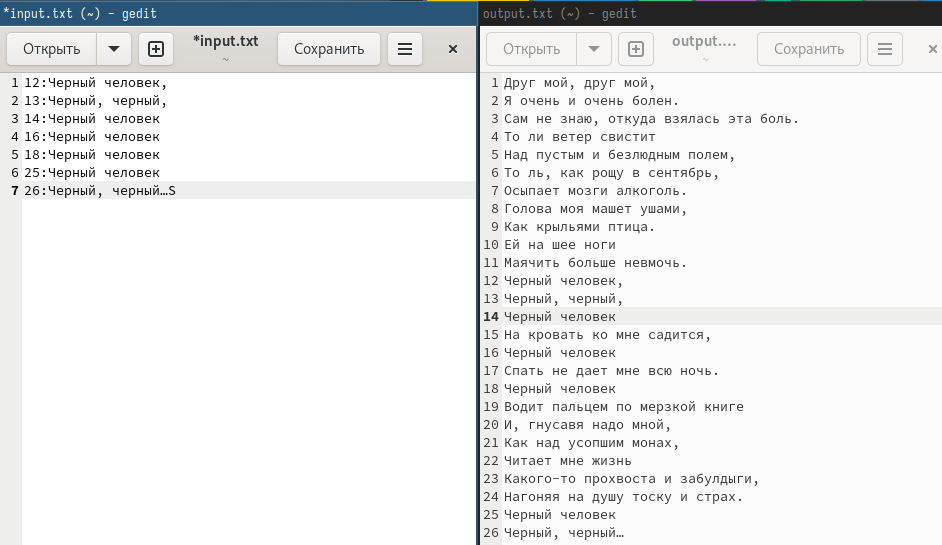


Рис. 4: Вывод программы

Создаю файлы для второй программы и открываю их в редакторе. (рис. 5).

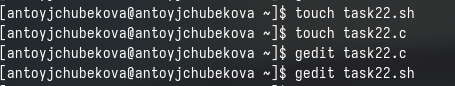


Рис. 5: Создание файла

В файле с расширением .c пишу программу на си, которая вводит число и определяет, является ли оно больше нуля, мент=ьше нуля, или равно нулю. Затем завершающаяся с помощью функции exit(n), передавая информацию о коде завершения в оболочку. Для этого с помощью if проверяю числа и вывожу соответствующий exit. (рис. 6).

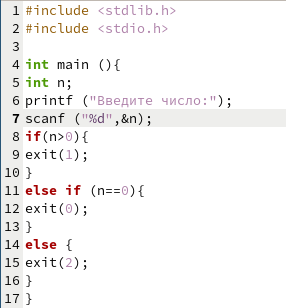


Рис. 6: Редактирование файла

Открываю файл с расширением .sh и создаю командный файл, который анализирует программу на си с помощью команды $? и выдает сообщения о том, какое число было введено. Для этого я использую команду gcc -o чтобы программма на си закомпилировалась в новый файл, затем с помощью case проверяю какое значения в этом файле и вывожу соответствующее сообщение. (рис. 7).

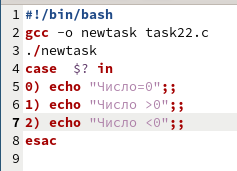


Рис. 7: Редактирование файла

Даю право на выполнение и запускаю программу с числами 4,0,-8, мы видим, что все работает корректно. (рис. 8).

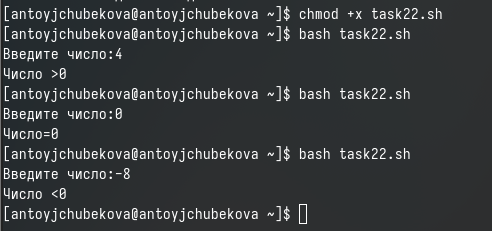


Рис. 8: Запуск программы

Создаю третий файл для написания командного файла и открываю его в редакторе. (рис. 9).

Создание файла

Рис. 9: Создание файла

Создаю командный файл, который создает указанное количество файлов, пронумерованные последовательно от 1 до n, число файлов же передается в аргументы командной строки. Также этот же командный файл должен удалять файлы с похожими именами, если они есть до создания новых. Для этого я запрашиваю у польнователя количество файлов, которые необходимо создать и прохожусь от 1 до этого числа параллельно проверяя есть ли уже файлы с такими именами если да, то удаляю их, а если нет создаю их. (рис. 10).

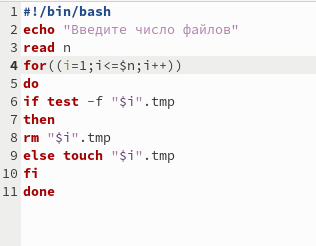


Рис. 10: Редактирование файла

Даю право на выполнение и запускаю программу с числом 4. (рис. 11).

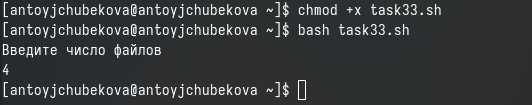


Рис. 11: Запуск программы

Зайдя в свой домашний каталог, можно видеть, что файлы удачно созданы. (рис. 12).

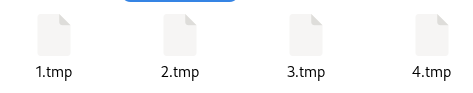


Рис. 12: Новые файлы

Создаю четвертый файл для написания командного файла и открываю его в редакторе. (рис. 13).

Создание файла

Рис. 13: Создание файла

Редактирую файл, пишу команднй файл, который с помощью команды tar запаковывает в архив все файлы в указанной директории, также модифицирует его так, чтобы запаковывались толко те файлы, которые были изменены менее недели назад . Для этого я использую команду find c опцией -mtime -7(чтобы указать, что менее 7 дней) и -mtime +0(чтобы программа не брала в учет сегодняшние файлы), а также type -f(для архивации исключительно файлов) вывод этих команд записываю в новый файл 2xfile.txt. Далее архивирую все файлы с помощью команды tar - cf и записываю вывод в файл archive2x.tar. (рис. 14).

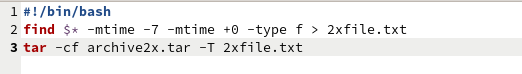


Рис. 14: Редактирование файла

Даю право на выполнение и запускаю программу. (рис. 15).

Запуск программы

Рис. 15: Запуск программы

Перейдя в домашний каталог видим, что программа сработала корректно. (рис. 16).

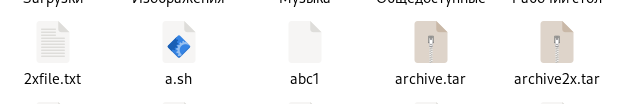


Рис. 16: Архив файлов

# 5 Ответы на вопросы

1. Команда getopts осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных.
2. Метасимвол позволяют использовать шаблоны для сопоставления файлов, основанных на их именах и других атрибутах.
3. Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости отрезультатов 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).
6. Строка if test -f mans/i.$s в командном файле оболочки Bash выполняет проверку с помощью утилиты test (или [) для определения того, является ли указанный файл обычным файлом.
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 Выводы

В ходе выполнения лабораторной работы № 13 я изучила основы программирования в оболочке ОС UNIX. Также научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов. Создала четыре командных файла и проверила их работу.