Лабораторная работа №11

Операционные системы

Голощапов Ярослав

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. 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

При перечислении имён файлов текущего каталога можно использовать следующие символы: – \* — соответствует произвольной, в том числе и пустой строке; – ? — соответствует любому одинарному символу; – [c1-c1] — соответствует любому символу, лексикографически находящемуся между символами c1 и с2. Например, – echo \* — выведет имена всех файлов текущего каталога, что представляет собой простейший аналог команды ls; – ls *.c — выведет все файлы с последними двумя символами, совпадающими с .c. – echo prog.? — выведет 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. Такие символы, как ’ < > \* ? |  ” &, являются метасимволами и имеют для командного процессора специальный смысл. Снятие специального смысла с метасимвола называется экранированием метасимвола. Экранирование может быть осуществлено с помощью предшествующего метасимволу символа , который, в свою очередь, является метасимволом. Для экранирования группы метасимволов нужно заключить её в одинарные кавычки. Строка, заключённая в двойные кавычки, экранирует все метасимволы, кроме $, ’ , , “. Например, – echo \* выведет на экран символ *, – echo ab’*|*’cd выведет на экран строку ab*|\*cd.

# 4 Выполнение лабораторной работы

Я создал файл one.sh для первой программы, а также текстовые файлы, откуда будет браться информация (рис. fig. 1). Написал код программы (рис. fig. 2).

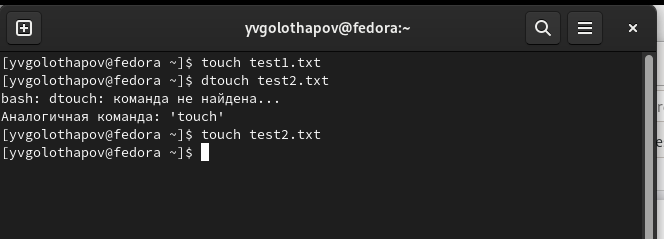


Рис. 1: Создание файлов

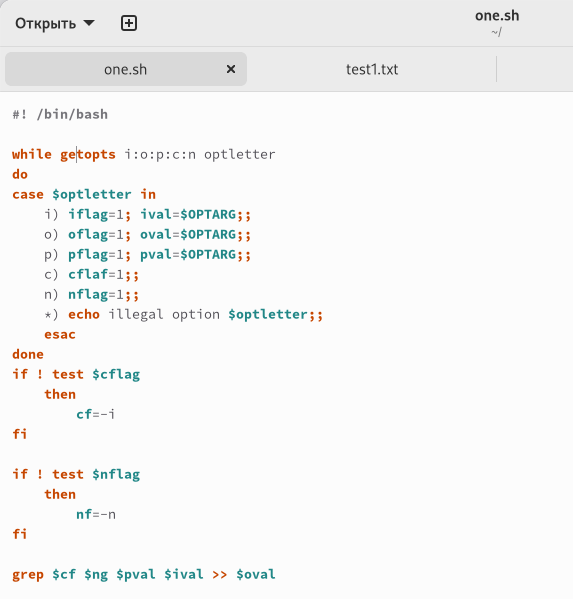


Рис. 2: Тело программы

Запустил программу и проверил, работает ли она (рис. fig. 3). Программа работает (рис. fig. 4), (рис. fig. 5).

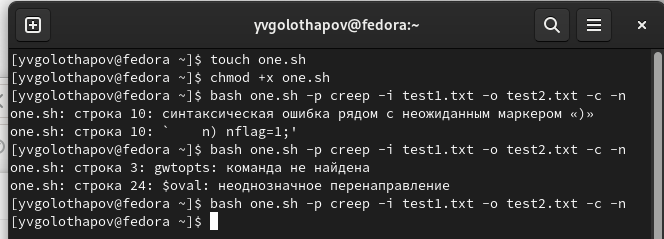


Рис. 3: Запуск программы

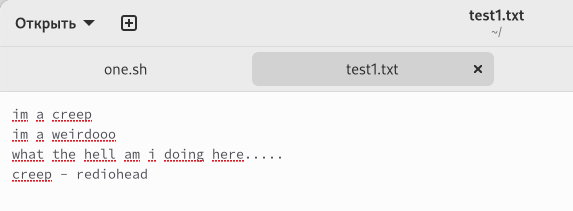


Рис. 4: Исходный текстовый файл

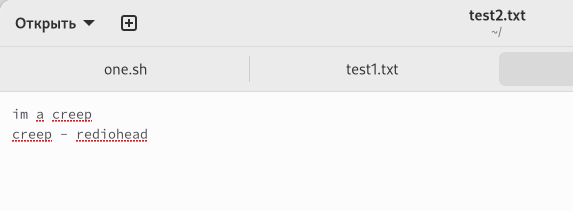


Рис. 5: Результат работы программы

Создал файлы для второй программы: two.sh и two.c и заполнил код (рис. fig. 6), (рис. fig. 7).

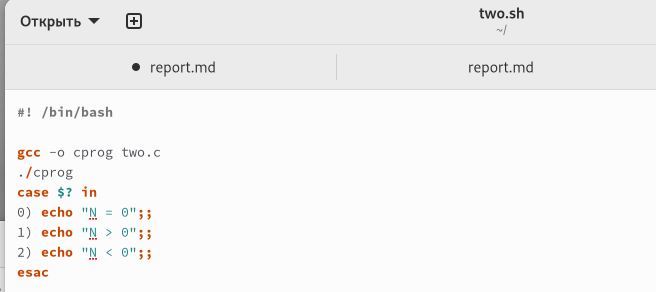


Рис. 6: Код основной программы

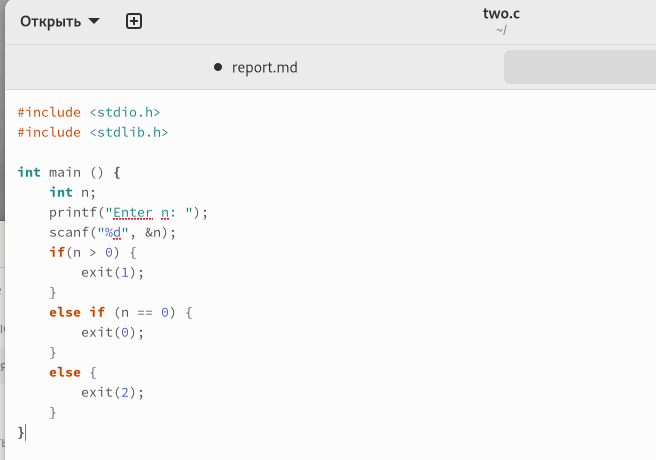


Рис. 7: Код подпрограммы

Проверил работу командного файла (рис. fig. 8).

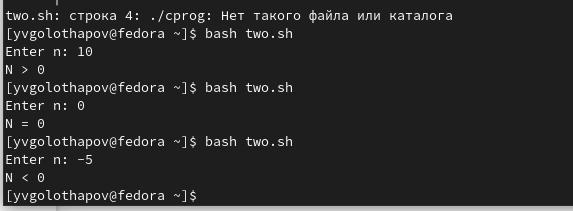


Рис. 8: Проверка работы

Создал третий файл three.sh и написал код (рис. fig. 9).



Рис. 9: Создание третьего файла

Запустил файл. Программа работает (рис. fig. 10), (рис. fig. 11).

Рис. 10: Запуск программы

Рис. 10: Запуск программы

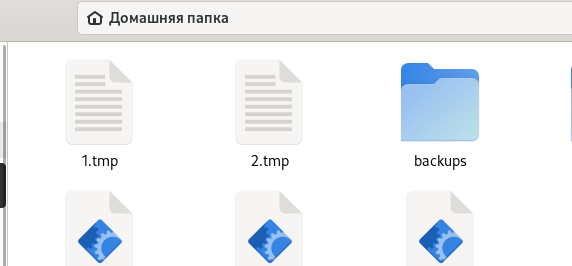


Рис. 11: Созданные файлы

Создал файл four.sh (рис. fig. 12).



Рис. 12: Файл 4

Программа работает (рис. fig. 13), (рис. fig. 14).

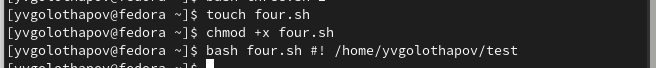


Рис. 13: Запуск

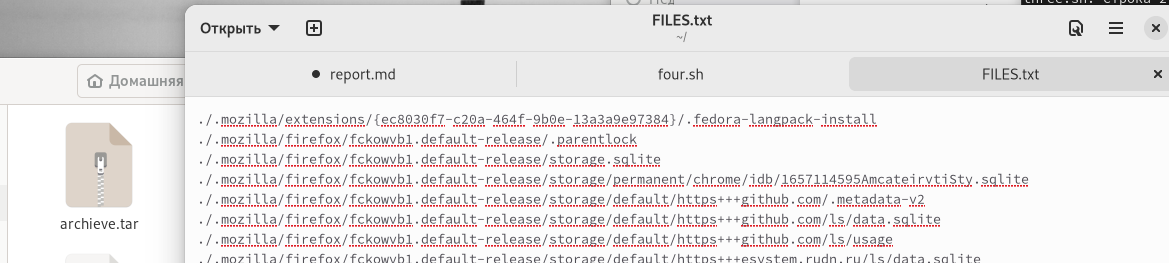


Рис. 14: Результат работы программы

# 5 Выводы

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

:::