Отчёт по лабораторной работе №12

Программирование в командном процессоре ОС UNIX. Командные файлы

Сергей Витальевич Павлюченков

Содержание

Список иллюстраций

Список таблиц

1 Цель работы

Изучить основы программирования в оболочке ОС UNIX/Linux. Научиться писать небольшие командные файлы.

2 Задание

1. Написать скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в вашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar. Способ использования команд архивации необходимо узнать, изучив справку.
2. Написать пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов.
3. Написать командный файл — аналог команды ls (без использования самой этой команды и команды dir). Требуется, чтобы он выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога.
4. Написать командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки.

3 Выполнение лабораторной работы

Создаю директорию для лабораторной работы и файл для первого задания

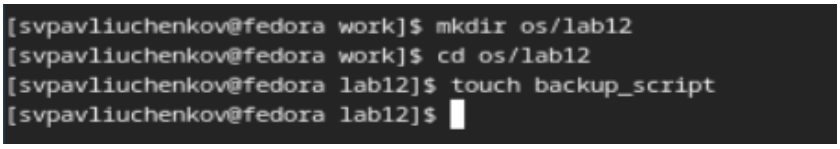

Рис. 3.1: Создание файла

Написал скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в вашем домашнем каталоге. При этом файл архивируется архиватором tar.

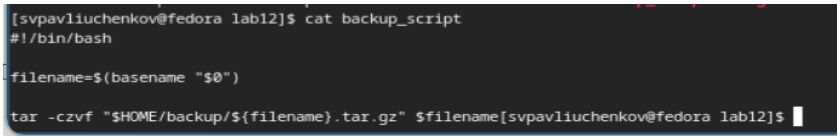

Рис. 3.2: Код программы

Делаю файл исполняемым

Рис. 3.3: Меняю права доступа к файлу

Проверяю правильность выполнения программы

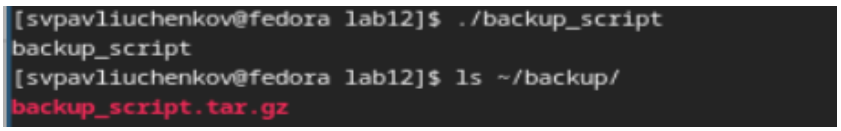

Рис. 3.4: Запуск программы

Создаю файл для второго задания

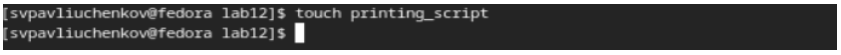

Рис. 3.5: Создание файла

Написал командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Скрипт может последовательно распечатывать значения всех переданных аргументов.

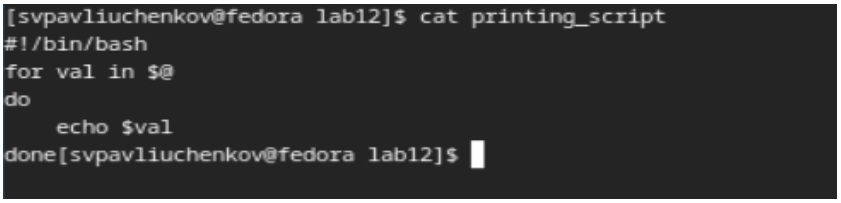

Рис. 3.6: Код программы

Делаю файл исполняемым

Рис. 3.7: Меняю права доступа к файлу

Проверяю правильность выполнения программы

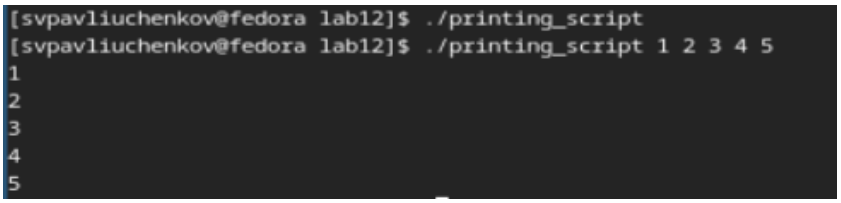

Рис. 3.8: Запуск программы

Создаю файл для третьего задания

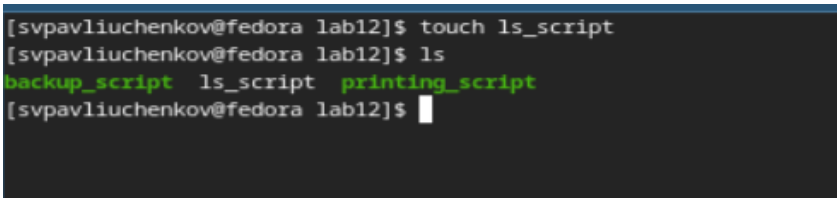

Рис. 3.9: Создание файла

Написал командный файл — аналог команды ls (без использования самой этой команды и команды dir).

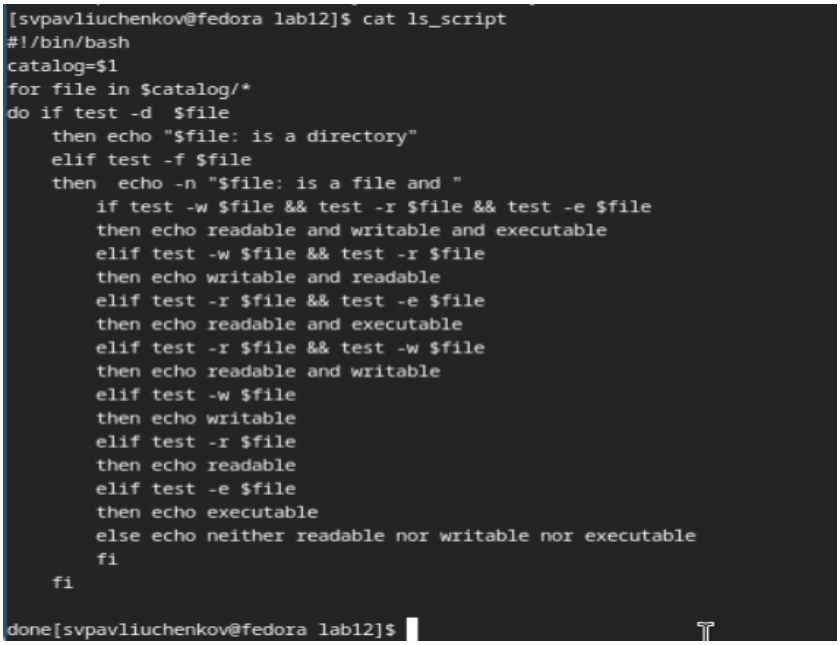

Рис. 3.10: Код программы

Делаю файл исполняемым

Рис. 3.11: Меняю права доступа к файлу

Проверяю правильность выполнения программы

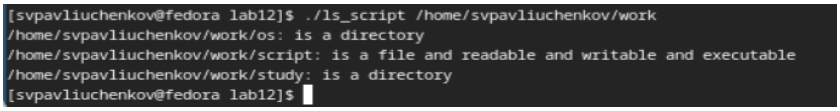

Рис. 3.12: Запуск программы

Создаю файл для четвертого задания

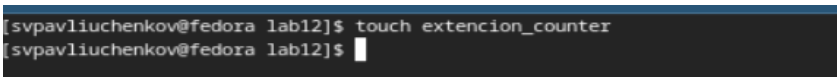

Рис. 3.13: Создание файла

Написал командный файл, который получает в качестве аргумента командной строки формат файла (.txt, .doc, .jpg, .pdf и т.д.) и вычисляет количество таких файлов в указанной директории. Путь к директории также передаётся в виде аргумента командной строки.

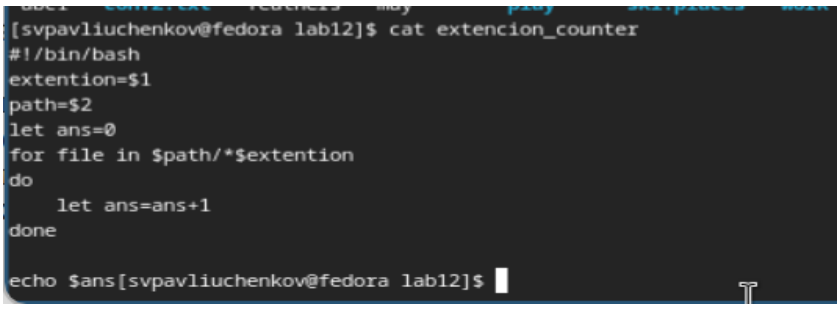

Рис. 3.14: Код программы

Делаю файл исполняемым

Рис. 3.15: Меняю права доступа к файлу

Проверяю правильность выполнения программы

Рис. 3.16: Запуск программы

4 Выводы

Я узнал много нового о bash и программирование в командной строке. Научился создавать командные файлы

5 Контрольные вопросы

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются?

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; – C-оболочка (или csh) — надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд; Главное отличие состоит в разном, но похожем синтаксисом.

2. Что такое POSIX?

POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ

3. Как определяются переменные и массивы в языке программирования bash? Для создания массива используется команда set с флагом -A. А переменные создаются простым присваиванием - a=“g”

4. Каково назначение операторов let и read? let указывает что указанная переменая это число, а read позволяет записывать данные в переменную с прямого ввода (input).

5. Какие арифметические операции можно применять в языке программирования bash?

Доступно сложение, деление, вычитание, умножение, взятие остатка от деления, побитовое сложение и вычитание тоже доступны.

6. Что означает операция (( ))?

Что внутри скобок происходит арифметические вычисления

7. Какие стандартные имена переменных Вам известны?

8. Что такое метасимволы?

Такие символы, как ’ < > * ? | ” &, являются метасимволами и имеют для командного процессора специальный смысл

9. Как экранировать метасимволы? Поставить перед метасимволом обратный слеш - \
10. Как создавать и запускать командные файлы?

Для создание командного файла нужно указать внутри файла для какой оболочки написан текст внутри файла, а для запуска файл нужно сделать исполнительным.

11. Как определяются функции в языке программирования bash?

Группу команд можно объединить в функцию. Для этого существует ключевое слово function, после которого следует имя функции и список команд, заключённых в фигурные скобки. Удалить функцию можно с помощью команды unset с флагом -f.

12. Каким образом можно выяснить, является файл каталогом или обычным файлом? test -f/-d название файла/каталога

13. Каково назначение команд set, typeset и unset? Первая команда используется для создания массивов Команда typeset имеет четыре опции для работы с функциями: – -f — перечисляет определённые на текущий момент функции; – -ft — при последующем вызове функции инициирует её трассировку; – -fx — экспортирует все перечисленные функции в любые дочерние программы оболочек; – -fu — обозначает указанные функции как автоматически загружаемые. Автоматически загружаемые функции хранятся в командных файлах, а при их вызове оболочка просматривает переменную FPATH, отыскивая файл с одноимёнными именами функций, загружает его и вызывает эти функции. Изъять переменную из программы можно с помощью команды unset.
14. Как передаются параметры в командные файлы?

Они записываются в память и их можно достать оттуда с помощью $номер параметра, или достать все сразу $*

15. Назовите специальные переменные языка bash и их назначение. – $* — отображается вся командная строка или параметры оболочки; – $! — номер процесса, в рамках которого выполняется последняя вызванная на выполнение в командном режиме команда; – $$ — уникальный идентификатор процесса, в рамках которого выполняется командный процессор;