Лабораторная работа №10.

Программирование в командном процессоре ОС UNIX. Командные файлы.

Александр Андреевич Шуплецов

Содержание

# 1 Цель работы

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

# 2 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 програм- мой совместимы с операторами оболочки Борна; – BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей сов- мещает 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.

# 3 Выполнение работы

1. Изучим справку tar.

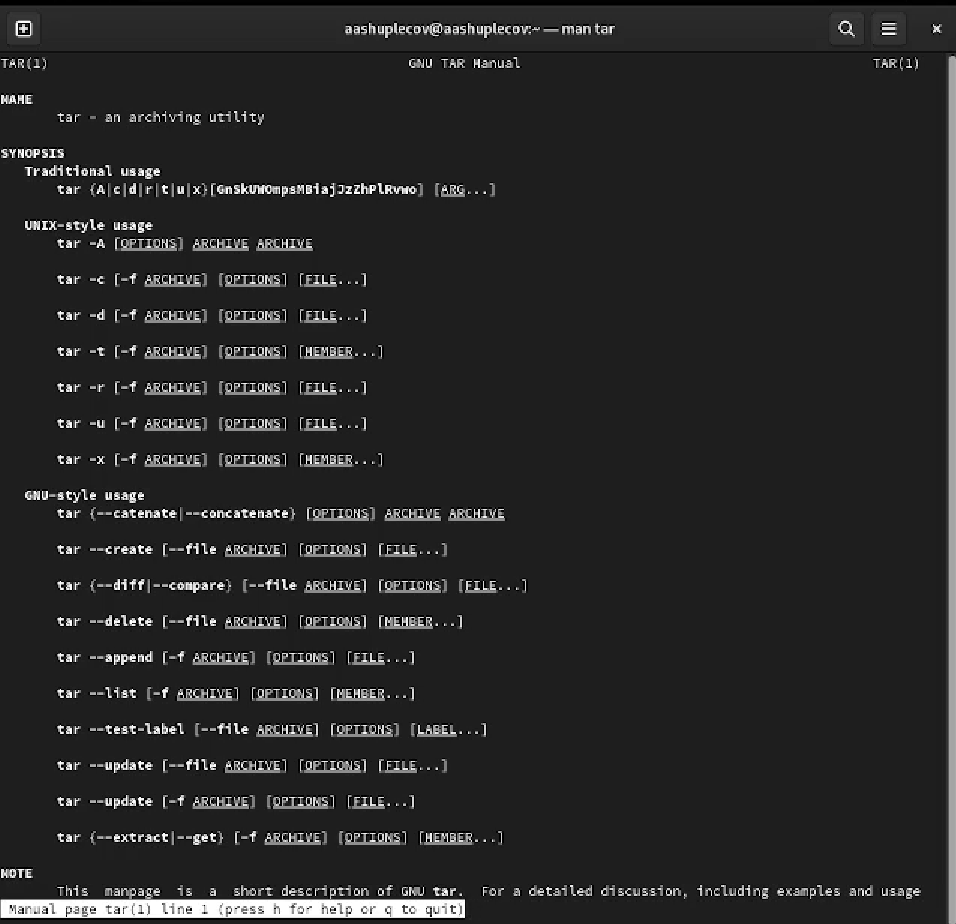


Figure 1: изучение справки tar

1. Напишем скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в нашем домашнем каталоге.

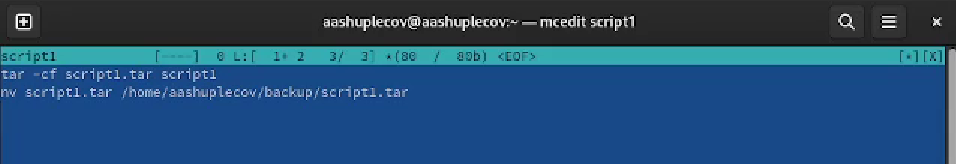


Figure 2: текст скрипта, делающий резервную копию

1. Убедимся, что скрипт, который при запуске будет делать резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в нашем домашнем каталоге, работает.

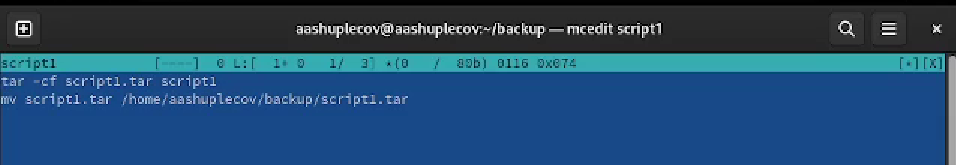


Figure 3: проверка скрипта, делающего резервную копию

1. Напишем пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять.

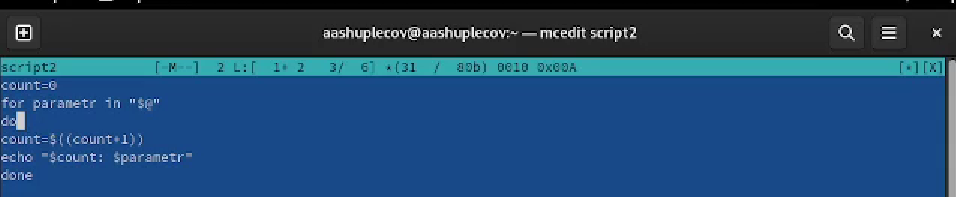


Figure 4: текст скрипта, обрабатывающего любое произвольное число аргументов

1. Убедимся, что скрипт, обрабатывающий любое произвольное число аргументов, работает.

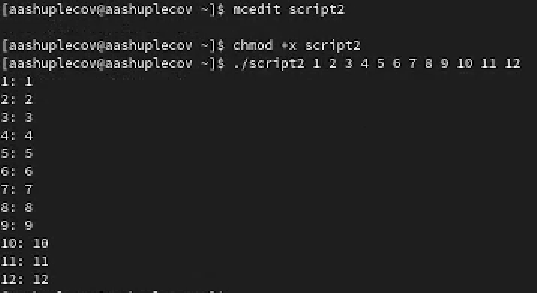


Figure 5: проверка скрипта, обрабатывающего любое произвольное число аргументов

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

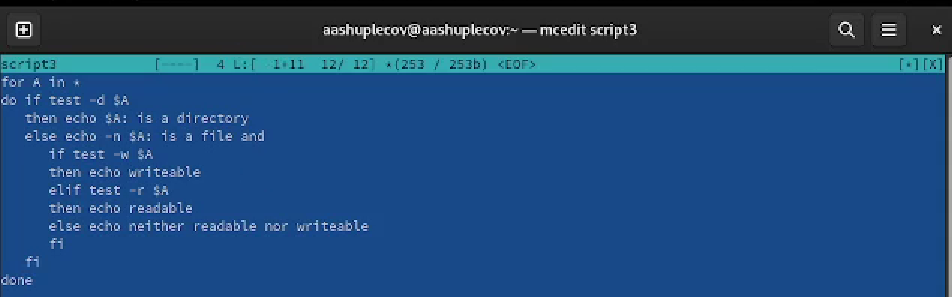


Figure 6: текст аналога команды ls

1. Убедимся, что командный файл - аналог команды ls, работает.

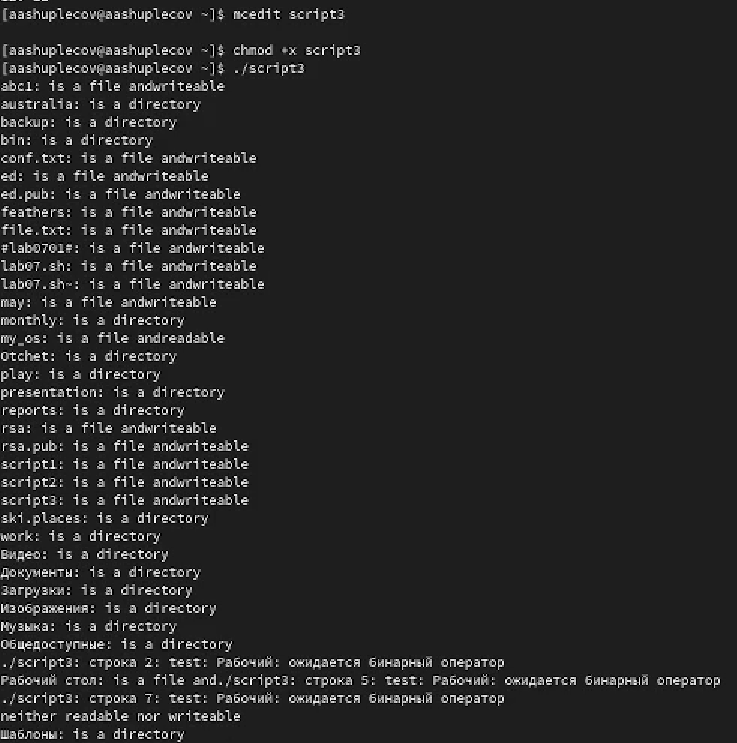


Figure 7: проверка аналога команды ls

1. Напишем 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.

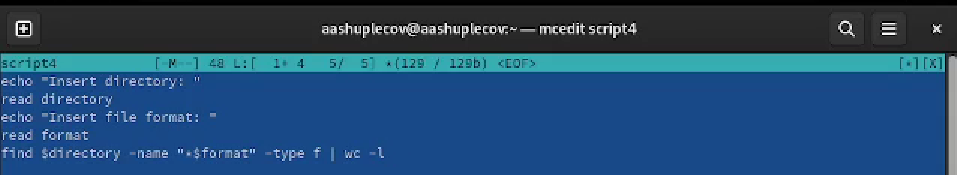


Figure 8: текст командного файла, вычисляющего кол-во файлов в директории

1. Убедимся, что 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, работает

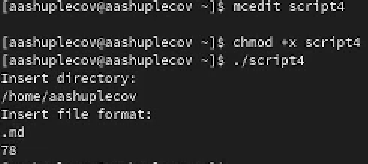


Figure 9: проверка командного файла, вычисляющего кол-во файлов в директории

# 4 Выводы

Я изучил основы программирования в оболочке ОС UNIX/Linux, научился писать небольшие командные файлы.

# Список литературы

Кулябов Д.С. “Материалы к лабораторным работам”