Лабораторная работа №12

Дисциплина: Операционные системы

Жибицкая Евгения Дмитриевна

Содержание

# 1 Цель работы

Изучение основ программирования в оболочке ОС UNIX. Написание небольших программ.

# 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

Перед началом непосредственно написания скриптов изучим принципы работы, узнаем синтаксис и особенности такого вида программирования.

Затем передем к написанию программ. Создадим директорию backup для резервного копирования и файл program1.sh. Откроем его для редактирования(рис. 1).

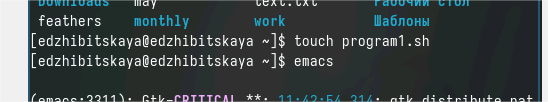


Рис. 1: Создание файла

Напишем первую программу. Копируем файл в нужный нам каталог, переходим в него, архивируем файл и удаляем ненужную версию (рис. 2).

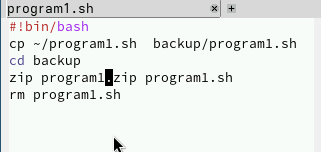


Рис. 2: Программа 1

Дадим файлу право на исполнение(chmod +x) и соответстенно запустим его (рис. 3).

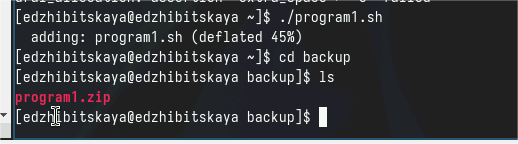


Рис. 3: Запуск программы 1

Создадим файл для второй программы, запустим его в редакторе emacs (рис. 4).

Создание второго файла

Рис. 4: Создание второго файла

Напишем программу, получающую на вход различные данные и циклом выводящую их обратно на экран (рис. 5).

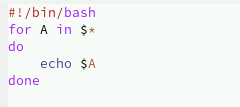


Рис. 5: Программа 2

Запустим программу, предварительно дав ей право на исполнение (рис. 6).

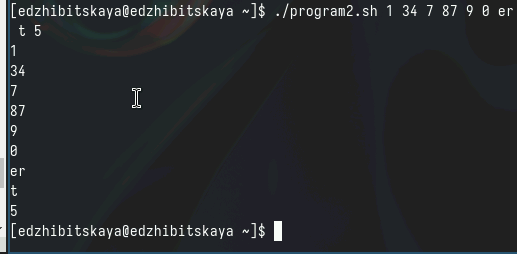


Рис. 6: Запуск программы 2

Перейдем к написанию скрипта 3. Также создадим файл, дадим его необходимые права и откроем в редакторе emacs.

Запустим цикл, который с помощью ключей определяет тип файла(или каталога) и делает проверку на его права(чтение или редактирование) (рис. 7).

Запустим файл, посмотрим на результат (рис. 8).

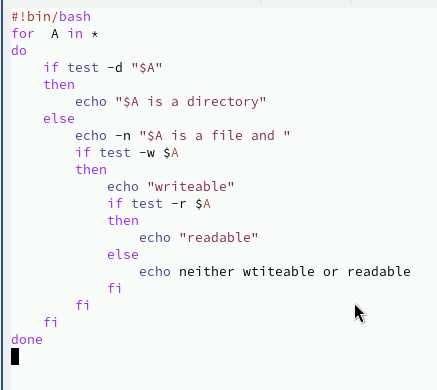


Рис. 7: Программа 3

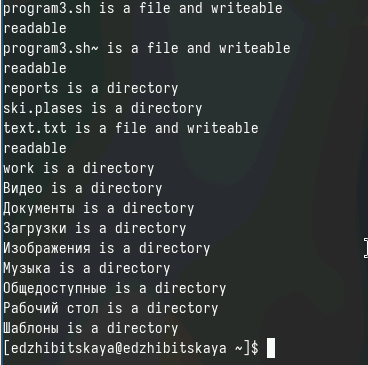


Рис. 8: Запуск программы 3

Перейдем к четвертому скрипту. Создадим файл, наделим его правами и откроем. Напишем программу. Здесь нам надо на вход получить название каталога и расширение, а затем пройтись по всем файлам и посчитать все, у которых такое же расширение (рис. 9).

Запустим программу (рис. 10).

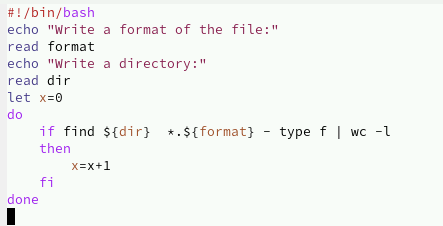


Рис. 9: Программа 4

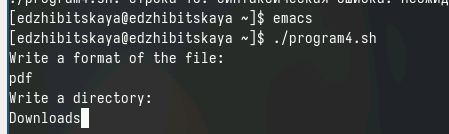


Рис. 10: Запуск программы 4

# 4 Контрольные вопросы

1. Командная оболочка - это интерфейс командной строки, который позволяет пользователю работать с операционной системой путем ввода текстовых команд. Примеры командных оболочек: bash, zsh, sh, ksh. Они отличаются друг от друга своими возможностями и синтаксисом команд.
2. POSIX (Portable Operating System Interface) - это набор стандартов, разработанных IEEE, определяющих интерфейсы для взаимодействия между операционной системой и прикладными программами.
3. Переменные в bash определяются путем присвоения им значения, например: variable=value

Массивы в bash определяются следующим образом: array=(value1 value2 value3)

1. Оператор let используется для выполнения арифметических операций, а оператор read позволяет считывать данные с клавиатуры.
2. В bash можно применять арифметические операции такие как сложение, вычитание, умножение, деление, остаток от деления и т.д.
3. Операция (( )) используется для вычисления арифметических выражений в bash.
4. Некоторые стандартные имена переменных в bash: HOME, PATH, USER, SHELL, PWD и т.д.
5. Метасимволы - это символы, которые имеют специальное значение в командной оболочке, например “\*“,”?“,”[“,”$” и т.д.
6. Для экранирования метасимволов в bash используют обратный слэш ““, например: \*.
7. Командные файлы создаются с помощью текстового редактора и сохраняются с расширением “.sh”. Для их запуска необходимо установить права на выполнение с помощью команды chmod +x, а затем выполнить файл через команду ./название\_файла.sh.
8. Функции в bash определяются следующим образом: function\_name() { команды }
9. Для определения, является ли файл каталогом или обычным файлом, можно использовать команду test:

if [ -d file ]; then echo “Это каталог” else echo “Это обычный файл” fi

1. Команды set, typeset и unset используются для работы с переменными в bash. set устанавливает значения переменных среды, typeset определяет тип переменной и устанавливает ее свойства, unset удаляет переменные.
2. Параметры передаются в командные файлы как аргументы командной строки. Например, $1 - первый параметр, $2 - второй параметр и так далее.
3. Специальные переменные языка bash:

* $0 - название выполняемого файла
* $# - количество аргументов
* $@ - все аргументы в виде списка
* $$ - идентификатор текущего процесса
* $? - код завершения последней выполненной команды

# 5 Выводы

В ходе работы мы познакомились с различным командами и особенностями программирования в оболочке UNIX, освоили написание небольших скриптов.