Отчет по лабораторной работе 12

Программирование в командном процессоре ОС UNIX. Командные файлы.

Власов Артем Сергеевич

Содержание

# 1 Цель работы

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

# 2 Задание

Выполнить последовательность действий по заданному сценарию, написать 4 скрипта для разных целей.

# 3 Выполнение лабораторной работы 12.

Создание файла первого скрипта и изменение его прав доступа. (рис. fig. 1).

Создание файла первого скрипта и изменение его прав доступа

Рис. 1: Создание файла первого скрипта и изменение его прав доступа

Код первого скрипта(бэкап в архиве). (рис. fig. 2).

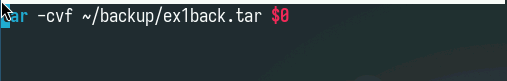


Рис. 2: Первый скрипт

Проверка работы первого скрипта. (рис. fig. 3).

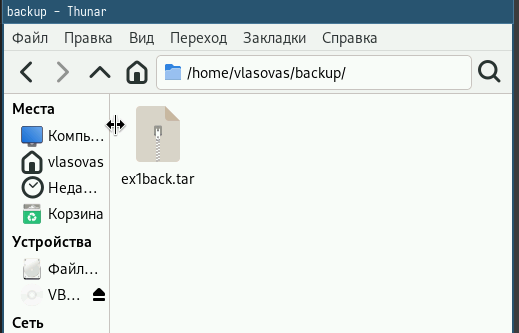


Рис. 3: Проверка

Создание файла второго скрипта и изменение его прав доступа. (рис. fig. 4).

Создание файла второго скрипта и изменение его прав доступа

Рис. 4: Создание файла второго скрипта и изменение его прав доступа

Код второго скрипта(вывод аргументов). (рис. fig. 5).



Рис. 5: Второй скрипт

Проверка работы второго скрипта. (рис. fig. 6).

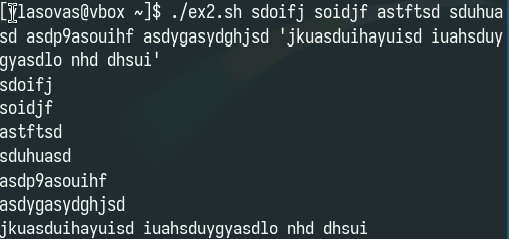


Рис. 6: Проверка

Создание файла третьего скрипта и изменение его прав доступа. (рис. fig. 7).

Создание файла третьего скрипта и изменение его прав доступа

Рис. 7: Создание файла третьего скрипта и изменение его прав доступа

Код третьего скрипта(файлы каталога и их права доступа) (рис. fig. 8).

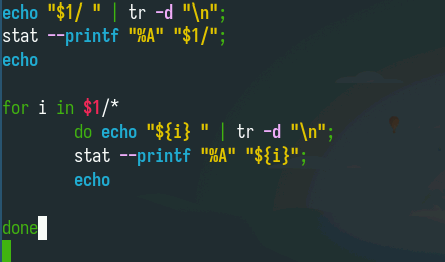


Рис. 8: Третий скрипт

Проверка работа третьего скрипта(рис. fig. 9).

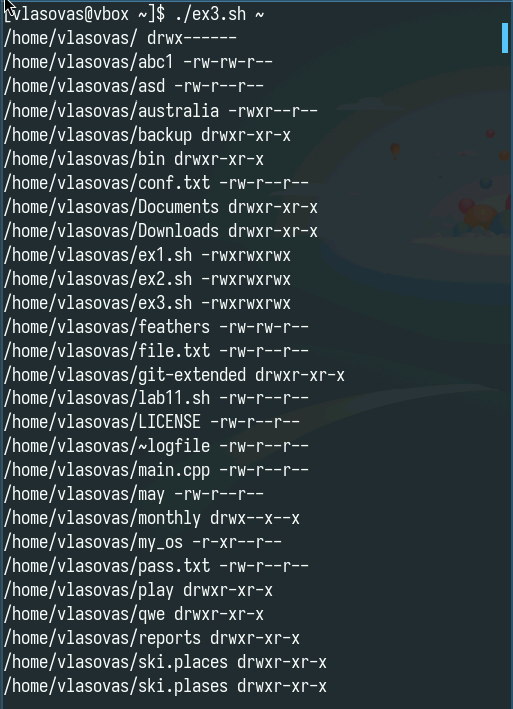


Рис. 9: Проверка

Создание файла четвертого скрипта и изменение его прав доступа.(рис. fig. 10).

Создание файла четвертого скрипта и изменение его прав доступа.

Рис. 10: Создание файла четвертого скрипта и изменение его прав доступа.

Код четвертого скрипта(количество файлов с заданным расширением). (рис. fig. 11).

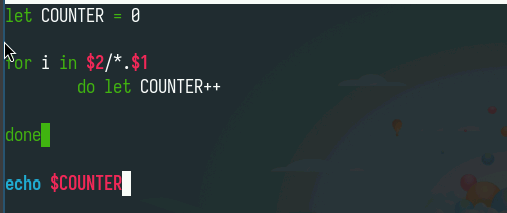


Рис. 11: Четвертый скрипт

Проверка работы четвертого скрипта. (рис. fig. 12).

Проверка

Рис. 12: Проверка

# 4 Контрольные вопросы

Командная оболочка - это интерпретатор команд, обеспечивающий взаимодействие пользователя с операционной системой, примеры: Bash, Zsh, Ksh. POSIX - это стандарт, обеспечивающий совместимость между UNIX-подобными операционными системами. В Bash переменные объявляются как VAR=“значение”, массивы - ARR=(“val1” “val2”) или declare -A DICT=([“key”]=“value”). Оператор let выполняет арифметические вычисления, а read используется для ввода данных пользователем. Bash поддерживает арифметические операции: +, -, \*, /, %, \*\*, ++, –. Конструкция (( )) предназначена для выполнения арифметических операций и сравнений. Стандартные переменные включают $HOME, $PATH, $USER, $PWD, $?, $$. Метасимволы - это специальные символы (\*, ?, >, |, & и др.), имеющие особое значение в shell. Экранировать метасимволы можно с помощью обратного слэша  или кавычек. Скрипты создаются как файлы .sh, делаются исполняемыми через chmod +x и запускаются ./script.sh. Функции определяются как name() { команды } или function name { команды }. Тип файла проверяется операторами -f (обычный файл) и -d (каталог). Команда set управляет параметрами shell, typeset задает атрибуты переменных, unset удаляет переменные. Параметры передаются через $1, $2…, $@ (все аргументы), $# (количество аргументов). Специальные переменные включают $0 (имя скрипта), $! (PID фонового процесса), $\_ (последний аргумент).

# 5 Выводы

Мы изучили основы программирования в оболочке ОС UNIX. Научились писать небольшие командные файлы.

# Список литературы