Отчёта по лабораторной работе №9:

Программирование в командном процессоре ОС UNIX. Командные файлы

Слуцкая Евгения Александровна

Содержание

# 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 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 программой совместимы с операторами оболочки Борна;
* BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает 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.

# 4 Выполнение лабораторной работы

1. Напишем скрипт, который при запуске будет делать резервную копию самого себя в другую директорию backup в вашем домашнем каталоге. При этом файл должен архивироваться одним из архиваторов на выбор zip, bzip2 или tar.

Для этого введем gedit prog1.sh, чтобы вызвать gedit редактор и создать файл для скрипта. После ввожу скрипт необходимый заданию (рис. 1):

mkdir -p ~/backup  
tar -cvf ~/backup/backup.tar prog1.sh

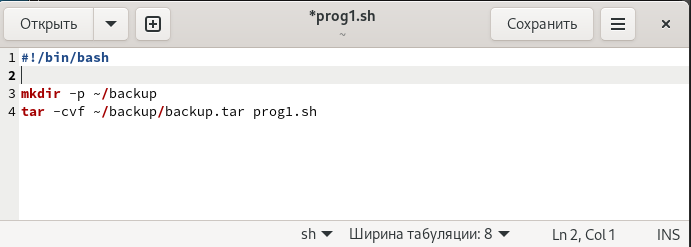


Рис. 1: Скрипт для создания архивной копии

Сделаем файл исполняемым с помощью chmod +x prog1.sh и проверяем, сработал ли скрипт. На скриншоте можно увидеть, что все работает (рис. 2).

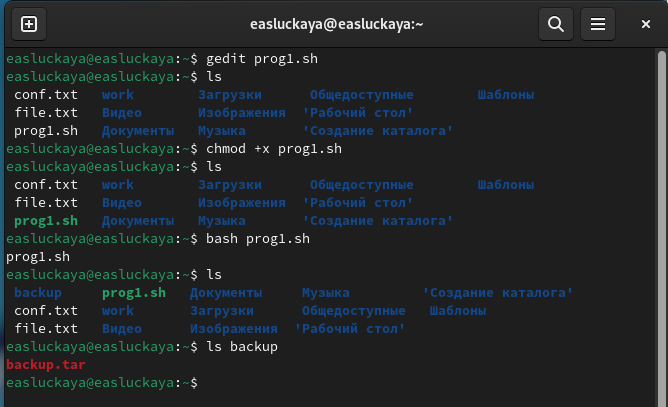


Рис. 2: Результат работы скрипта №1

1. Напишем пример командного файла, обрабатывающего любое произвольное число аргументов командной строки. (рис. 3):

for arg in $\*  
do  
 echo $arg  
done

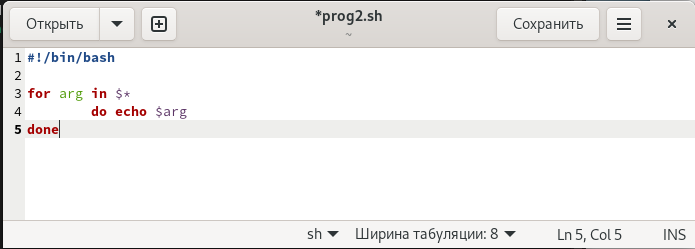


Рис. 3: Скрипт вывода аргументов командной строки

Делаем файл исполняемым и выводим результат. В данном случае, будет выводится последовательность аргументов командной строки (рис. 4).

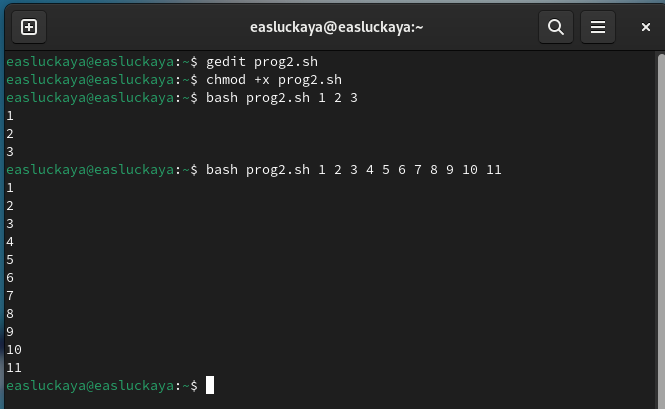


Рис. 4: Результат работы скрипта №2

1. Напишем командный файл — аналог команды ls (без использования самой этой команды и команды dir), который выдавал информацию о нужном каталоге и выводил информацию о возможностях доступа к файлам этого каталога (рис. 5):

for arg in $\*  
do  
 echo $arg  
done



Рис. 5: Скрипт аналога команды ls

Делаем файл исполняемым и выводим результат (рис. 6).

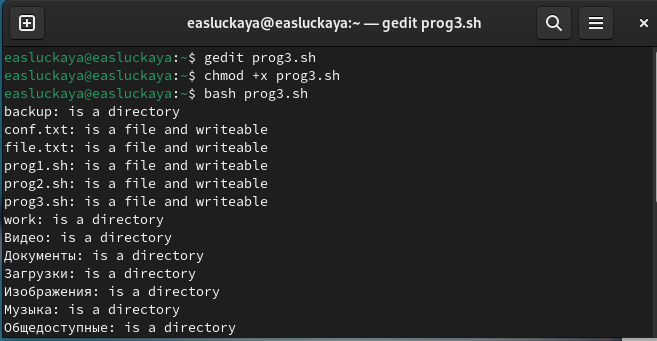


Рис. 6: Результат работы скрипта №3

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. (рис. 7):

format=""  
directory=""  
echo "Напишите формат файла"  
read format  
echo "Напишите директорию"  
read directory  
find "${directory}" -name "\*.${format}" -type f | wc -l  
ls



Рис. 7: Скрипт подсчета количества файлов нужного формата

Делаем файл исполняемым и выводим результат (рис. 8).

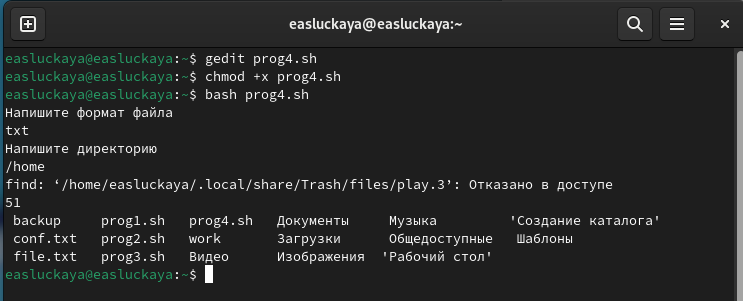


Рис. 8: Результат работы скрипта №4

# 5 Контрольные вопросы

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

**Командная оболочка** – это интерфейс между пользователем и операционной системой, который позволяет пользователю взаимодействовать с операционной системой путем ввода текстовых команд. Примеры командных оболочек включают Bash (Bourne Again Shell), Zsh (Z Shell), Fish (Friendly Interactive Shell) и другие. Они отличаются по своим возможностям, синтаксису, встроенным функциям и поддерживаемым расширениям.

1. **Что такое POSIX?**

**POSIX** (Portable Operating System Interface) – это семейство стандартов, разработанных для обеспечения совместимости между различными операционными системами Unix. Он определяет общие интерфейсы для программирования на языке C, командной строки и управления файлами.

1. **Как определяются переменные и массивы в языке программирования bash?**

В языке программирования bash переменные определяются путем присваивания значений их именам. Например:

* Переменные: variable\_name=value
* Массивы: array\_name[index]=value

1. **Каково назначение операторов let и read?**

**Оператор let** используется для выполнения арифметических выражений в bash. **Оператор read** используется для считывания значений из стандартного ввода и присваивания их переменным.

1. **Какие арифметические операции можно применять в языке программирования bash?**

В языке программирования bash можно применять стандартные арифметические операции, такие как сложение, вычитание, умножение и деление.

1. **Что означает операция (( ))?**

Операция (( )) в bash используется для выполнения арифметических вычислений.

1. **Какие стандартные имена переменных Вам известны?**

Некоторые стандартные имена переменных в bash:

* HOME: домашний каталог текущего пользователя.
* PWD: текущий рабочий каталог.
* PATH: список каталогов, в которых операционная система ищет исполняемые файлы.
* USER: имя текущего пользователя.

1. **Что такое метасимволы?**

**Метасимволы** – это символы, которые имеют специальное значение в контексте командной строки или шаблонов файлов. Некоторые примеры метасимволов включают \*, ?, [ ], { }, |, ; и &.

1. **Как экранировать метасимволы?**

Для экранирования метасимволов в bash используется обратная косая черта \. Например, чтобы использовать символ \* как обычный символ, его можно экранировать так: \\*.

1. **Как создавать и запускать командные файлы?**

Для создания и запуска командных файлов в bash можно использовать текстовый редактор для создания файла с расширением .sh, затем присвоить ему права на выполнение с помощью команды chmod +x filename.sh, и, наконец, запустить файл с помощью команды ./filename.sh.

1. **Как определяются функции в языке программирования bash?**

Функции в языке программирования bash определяются с использованием ключевого слова function или просто с именем функции, после чего идет блок кода. Например:

function my\_function {  
 # Код функции  
 }

1. **Каким образом можно выяснить, является файл каталогом или обычным файлом?**

Для определения, является ли файл каталогом или обычным файлом, можно использовать команду test. Например:

* Проверка на каталог: test -d filename
* Проверка на обычный файл: test -f filename

1. **Каково назначение команд set, typeset и unset?**

**Команды set, typeset и unset** используются для работы с переменными в bash:

* set: устанавливает значения и флаги для параметров командной строки.
* typeset: используется для объявления переменных с определенными свойствами, такими как readonly или integer.
* unset: удаляет значения переменных.

1. **Как передаются параметры в командные файлы?**

Параметры передаются в командные файлы в виде аргументов командной строки. Они доступны внутри скрипта через специальные переменные $1, $2, $3 и так далее, где $1 содержит первый аргумент, $2 – второй и т.д.

1. **Назовите специальные переменные языка bash и их назначение.**

Некоторые специальные переменные языка bash и их назначение:

* $0: имя текущей выполняемой программы.
* $#: количество аргументов, переданных скрипту.
* $?: код возврата последней выполненной команды.
* $$: PID (идентификатор процесса) текущего скрипта.
* $!: PID последнего запущенного фонового процесса.

# 6 Выводы

В данной лабораторной работе мы изучили основы программирования в оболочке ОС UNIX/Linux, а также научились писать небольшие командные файлы.

# Список литературы

1. Руководство к лабораторной работе №9.