Отчёт по лабораторной работе №12

Программирование в командном процессоре ОС UNIX. Командные файлы

Абакарова Милана

Содержание

1 Цель работы **4**

2 Выполнение лабораторной работы **5**

3 Вывод **8**

4 Контрольные вопросы **9**

List of Figures

2.1 Задание 1 … 5
2.2 Задание 2 … 6
2.3 Задание 3 … 6

1 Цель работы

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

2 Выполнение лабораторной работы

1. Написали скрипт, который при запуске делает резервную копию самого себя (то есть файла, в котором содержится его исходный код) в другую директорию backup в моём домашнем каталоге. При этом файл архивируется одним из архиваторов на выбор zip , bzip2 или tar . Способ использования команд архивации узнали, изучив справку.

Комментарий: командой cp копируем файл в директорию ~/backup/, а командой gzip исходный файл архивируется и удаляется (остаётся только архив).

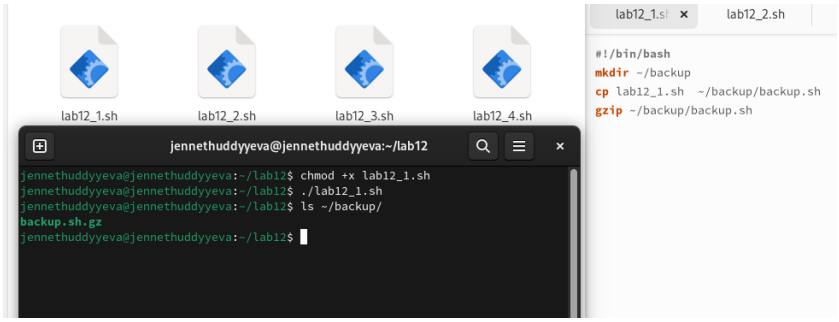

Figure 2.1: Задание 1

2. Написали пример командного файла, обрабатывающего любое произвольное число аргументов командной строки, в том числе превышающее десять. Например, скрипт может последовательно распечатывать значения всех переданных аргументов

for i — для всех переданных аргументов

do echo $1 — выводим первый аргумент

shift — удаляем первый аргумент, смещаем все аргументы

done — конец цикла

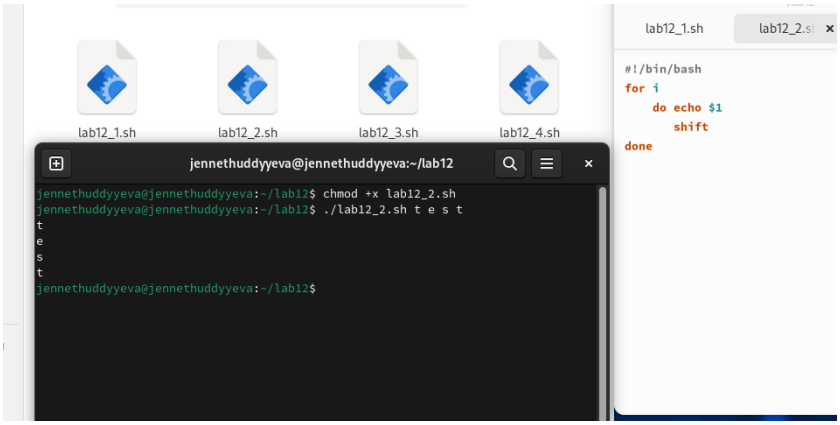

Figure 2.2: Задание 2

3. Написали командный файл — аналог команды ls (без использования самой этой команды и команды dir). Он выдает информацию о нужном каталоге и выводит информацию о возможностях доступа к файлам этого каталога.

Комментарий: если не использовать команду ls или команду dir, то данную задачу легко выполнить с помощью команды find, если указать ей опцию поиска файлов с определенным правом доступа

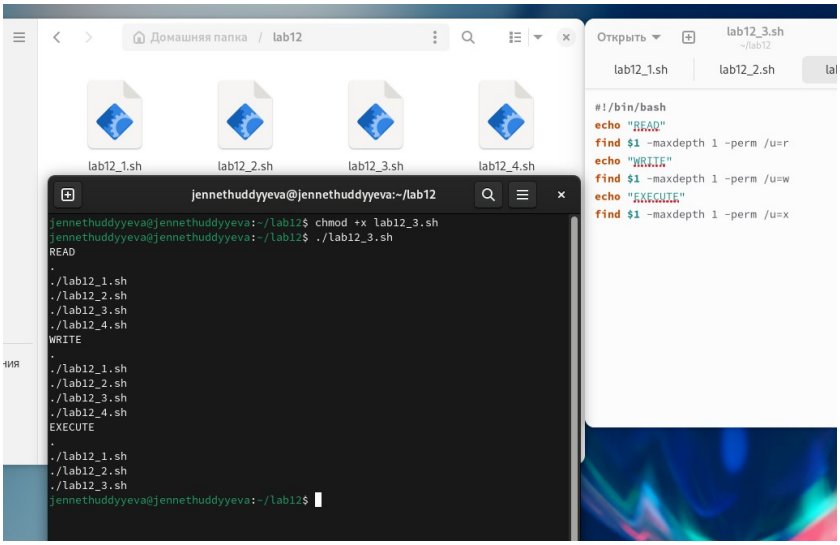

Figure 2.3: Задание 3

4. Написали 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.

Комментарий: ищем командой find в каталоге $1 (первый аргумент) файлы заканчивающиеся “*” на нужное расширение $2 (аргумент второй) передаем вывод | в команду подсчета wc с аргументом считающим слова -l

3 Вывод

В данной работе мы изучили основы программирования в оболочке ОС UNIX/Linux. Научились писать небольшие командные файлы и скрипты на языке bush.

4 Контрольные вопросы

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

a) sh — стандартная командная оболочка UNIX/Linux, содержащая базовый, полный набор функций
b) csh — использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд
c) ksh — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна
d) bash — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна

2. Что такое POSIX? ОтветPOSIX (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? Ответ: Переменные вызываются $var, где var=чему-то, указанному пользователем, неважно что бы то не было, название файла, каталога или еще чего. Для массивов используется команда set -A

4. Каково назначение операторов let и read? Ответ: let — вычисляет далее заданное математическое значение read — позволяет читать значения переменных со стандартного ввода

5. Какие арифметические операции можно применять в языке программирования bash? Ответ: Прибавление, умножение, вычисление, деление), сравнение значений, экспонирование и др.

6. Что означает операция (( ))? Ответ: Это обозначение используется для облегчения программирования для условий bash

7. Какие стандартные имена переменных Вам известны? Ответ: Нам известны HOME, PATH, BASH, ENV, PWD, UID, OLDPWD, PPID, GROUPS, OSTYPE, PS1 - PS4, LANG, HOSTFILE, MAIL, TERM, LOGNAME, USERNAME, IFS и др.

8. Что такое метасимволы? Ответ: Метасимволы это специальные знаки, которые могут использоваться для сокращения пути, поиска объекта по расширению, перед переменными, например «$» или «*» .

9. Как экранировать метасимволы? Ответ: Добавить перед метасимволом метасимвол «»

10. Как создавать и запускать командные файлы? Ответ: При помощи команды chmod. Надо дать права на запуск chmod +x название файла, затем запустить bash ./название файла Например у нас файл lab Пишем: chmod +x lab ./lab

11. Как определяются функции в языке программирования bash? Ответ: Объединяя несколько команд с помощью function

12 Каким образом можно выяснить, является файл каталогом или обычным файлом? Ответ: Можно задать команду на проверку диретория ли это test -d директория

13 Каково назначение команд set, typeset и unset? Ответ: Set — используется для создания массивов Unset — используется для изъятия переменной Typeset — используется для присваивания каких-либо функций

14. Как передаются параметры в командные файлы? Ответ: Добавлением аргументов после команды запуска bash скрипта

15. Назовите специальные переменные языка bash и их назначение. Ответ:

– $* — отображается вся командная строка или параметры оболочки;
– $? — код завершения последней выполненной команды;
– $$ — уникальный идентификатор процесса, в рамках которого выполняется к
– $! — номер процесса, в рамках которого выполняется последняя вызванная н
– $- — значение флагов командного процессора;
– ${#*} — возвращает целое число — количество слов, которые были результа
$*;
– ${#name} — возвращает целое значение длины строки в переменной name;
– ${name[n]} — обращение к n-му элементу массива;
– ${name[*]} — перечисляет все элементы массива, разделённые пробелом;
– ${name[@]} — то же самое, но позволяет учитывать символы пробелы в сами
– ${name:-value}—еслизначениепеременнойnameнеопределено,тоонобудет зам
– ${name:value} — проверяется факт существования переменной;
– ${name=value} — если name не определено, то ему присваивается значение
– ${name?value} — останавливает выполнение, если имя переменной не опред
– ${name+value} — это выражение работает противоположно ${name-
value}. Если переменная определена, то подставляется value;
– ${name#pattern} — представляет значение переменной name с удалённым са
– ${#name[*]}и${#name[@]}—этивыраженияявозвращаютколичествоэлементов