

Отчёт по лабораторной работе №11

Операционные системы

Гусева Светлана Алексеевна

Содержание

1	Цель работы	5
2	Задание	6
3	Выполнение лабораторной работы	7
4	Контрольные вопросы	12
5	Выводы	16
	Список литературы	17

Список таблиц

Список иллюстраций

3.1	Создание папки backup.	7
3.2	Скрипт и выполнение задания 1.	8
3.3	Скрипт и выполнение задания 2.	8
3.4	Скрипт задания 3.	9
3.5	Результат выполнения задания 3.	9
3.6	Результат выполнения задания 3.	10
3.7	Результат выполнения задания 3.	10
3.8	Скрипт и выполнение задания 4.	11

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. Выполнение лабораторной работы

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.

Создание папки backup (рис. 3.1).

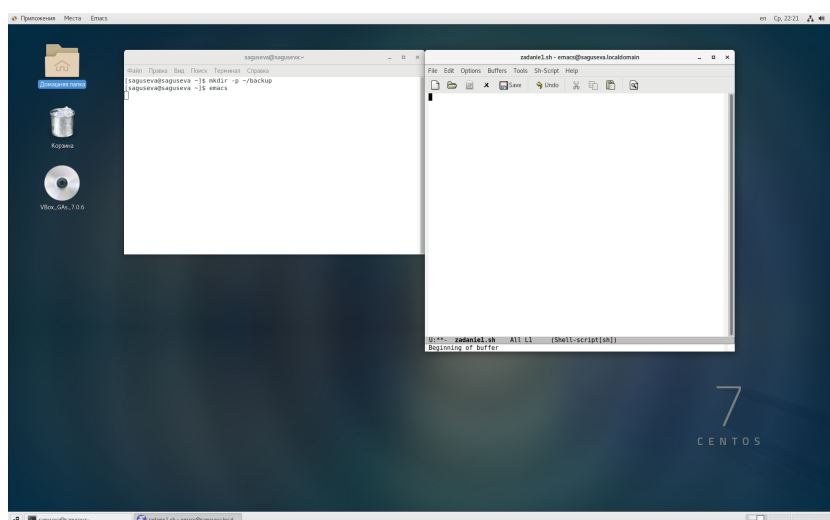


Рис. 3.1: Создание папки backup.

Скрипт и результат выполнения задания 1 (рис. 3.2).

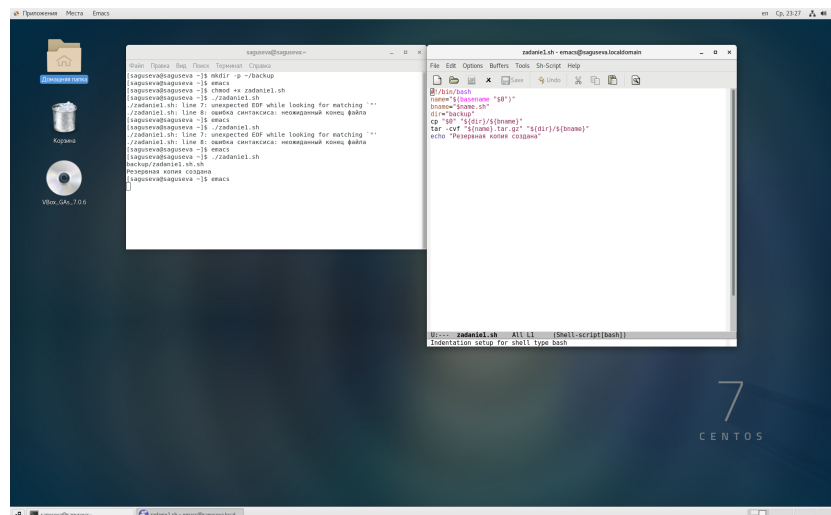


Рис. 3.2: Скрипт и выполнение задания 1.

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. Скрипт и результат выполнения задания 2 (рис. 3.3).

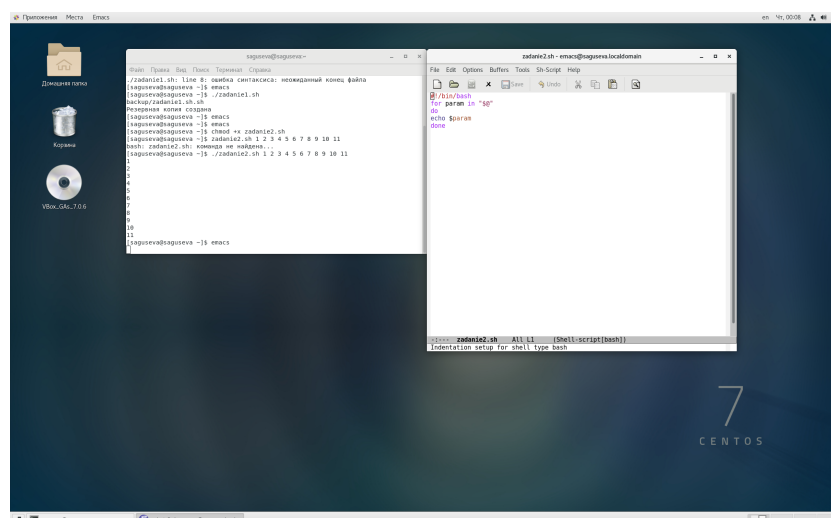
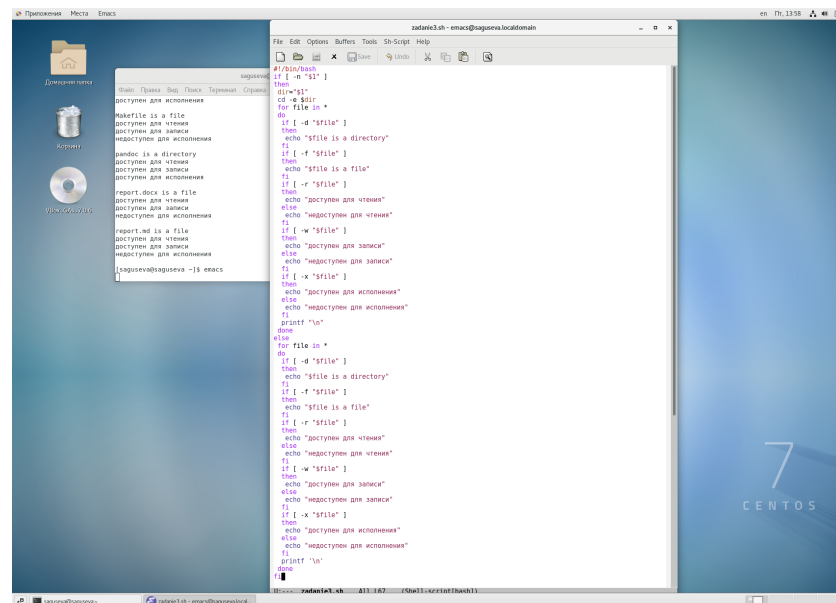


Рис. 3.3: Скрипт и выполнение задания 2.

3. Написать командный файл — аналог команды ls (без использования самой этой команды и команды dir). Требуется, чтобы он выдавал информацию о нужном каталоге

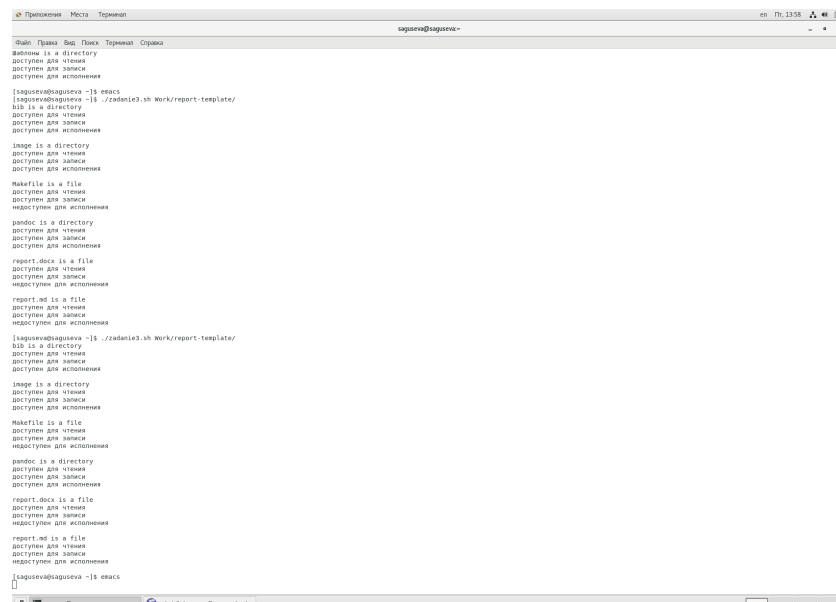
и выводил информацию о возможностях доступа к файлам этого каталога. (рис. 3.4).

Скрипт выполнения задания 3 (рис. 3.4), результат выполнения (рис. 3.5, 3.6, 3.7).



```
#!/bin/bash
if [ -n "$1" ]
then
    dir="$1"
    cd -> $dir
    for file in *
    do
        if [ -d "$file" ]
        then
            echo "$file is a directory"
        fi
        if [ -f "$file" ]
        then
            echo "$file is a file"
        fi
        if [ -r "$file" ]
        then
            echo "доступен для чтения"
        else
            echo "недоступен для чтения"
        fi
        if [ -w "$file" ]
        then
            echo "доступен для записи"
        else
            echo "недоступен для записи"
        fi
        if [ -x "$file" ]
        then
            echo "доступен для исполнения"
        else
            echo "недоступен для исполнения"
        fi
        printf '\n'
    done
else
    for file in *
    do
        if [ -d "$file" ]
        then
            echo "$file is a directory"
        fi
        if [ -f "$file" ]
        then
            echo "$file is a file"
        fi
        if [ -r "$file" ]
        then
            echo "доступен для чтения"
        else
            echo "недоступен для чтения"
        fi
        if [ -w "$file" ]
        then
            echo "доступен для записи"
        else
            echo "недоступен для записи"
        fi
        if [ -x "$file" ]
        then
            echo "доступен для исполнения"
        else
            echo "недоступен для исполнения"
        fi
        printf '\n'
    done
fi
```

Рис. 3.4: Скрипт задания 3.



```
Вашлон is a directory
доступен для чтения
доступен для записи
доступен для исполнения

[laguseva@laguseva ~]$ ewacs
[laguseva@laguseva ~]$ ./zadanie3.sh Work/report-template/
b1b is a directory
доступен для чтения
доступен для записи
доступен для исполнения

Image is a directory
доступен для чтения
доступен для записи
доступен для исполнения

Makefile is a file
доступен для чтения
доступен для записи
недоступен для исполнения

randoc is a directory
доступен для чтения
доступен для записи
доступен для исполнения

report.docx is a file
доступен для чтения
доступен для записи
недоступен для исполнения

report.md is a file
доступен для чтения
доступен для записи
недоступен для исполнения

[laguseva@laguseva ~]$ ./zadanie3.sh Work/report-template/
b1b is a directory
доступен для чтения
доступен для записи
доступен для исполнения

Image is a directory
доступен для чтения
доступен для записи
доступен для исполнения

Makefile is a file
доступен для чтения
доступен для записи
недоступен для исполнения

randoc is a directory
доступен для чтения
доступен для записи
доступен для исполнения

report.docx is a file
доступен для чтения
доступен для записи
недоступен для исполнения

report.md is a file
доступен для чтения
доступен для записи
недоступен для исполнения

[laguseva@laguseva ~]$ ewacs
```

Рис. 3.5: Результат выполнения задания 3.

```
Файл Правка Вид Поиск Терминал Справка
[серики@серикина ~]$ ls -la
[серики@серикина ~]$ ./zadani2.sh
backdir is a directory
доступен для чтения
доступен для записи
доступен для исполнения

gh-2.27.0-linux-386.rpm is a file
доступен для чтения
доступен для записи
недоступен для исполнения

install-1.1-2020419 is a directory
доступен для чтения
доступен для записи
доступен для исполнения

install-tl-unix.tar.gz is a file
доступен для чтения
доступен для записи
недоступен для исполнения

lab07.sh is a file
доступен для чтения
доступен для записи
недоступен для исполнения

lab07.sh is a file
доступен для чтения
доступен для записи
недоступен для исполнения

lab101.sh is a file
доступен для чтения
доступен для записи
недоступен для исполнения

lab102.sh is a file
доступен для чтения
доступен для записи
недоступен для исполнения

lab103.sh is a file
доступен для чтения
доступен для записи
недоступен для исполнения

lab104.sh is a file
доступен для чтения
доступен для записи
недоступен для исполнения

mdpdf is a directory
доступен для чтения
доступен для записи
доступен для исполнения

os-intro is a directory
доступен для чтения
доступен для записи
доступен для исполнения

randoc-3.1.2 is a directory
доступен для чтения
доступен для записи
доступен для исполнения

randoc-3.1.2-linux-wd04.tar.gz is a file
доступен для чтения
доступен для записи
недоступен для исполнения

[серики@серикина ~]$
```

Рис. 3.6: Результат выполнения задания 3.

```
Файл Правка Вид Поиск Терминал Справка
[серики@серикина ~]$
доступен для чтения
доступен для записи
доступен для исполнения

zadani1.sh is a file
доступен для чтения
доступен для записи
доступен для исполнения

zadani1.sh.tar.gz is a file
доступен для чтения
доступен для записи
недоступен для исполнения

zadani2.sh is a file
доступен для чтения
доступен для записи
доступен для исполнения

zadani2.sh is a file
доступен для чтения
доступен для записи
доступен для исполнения

zadani3.sh is a file
доступен для чтения
доступен для записи
доступен для исполнения

Видео is a directory
доступен для чтения
доступен для записи
доступен для исполнения

Документы is a directory
доступен для чтения
доступен для записи
доступен для исполнения

Загрузки is a directory
доступен для чтения
доступен для записи
доступен для исполнения

Избранное is a directory
доступен для чтения
доступен для записи
доступен для исполнения

Музыка is a directory
доступен для чтения
доступен для записи
доступен для исполнения

Общедоступные is a directory
доступен для чтения
доступен для записи
доступен для исполнения

Рабочий стол is a directory
доступен для чтения
доступен для записи
доступен для исполнения

Шаблоны is a directory
доступен для чтения
доступен для записи
доступен для исполнения

[серики@серикина ~]$
```

Рис. 3.7: Результат выполнения задания 3.

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. Скрипт и результат выполнения задания 4 (рис. 3.8).

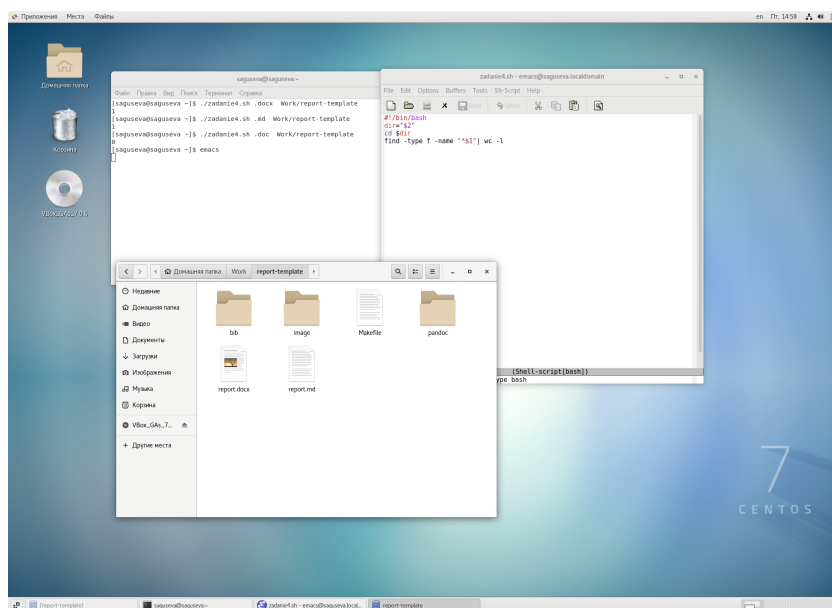


Рис. 3.8: Скрипт и выполнение задания 4.

4. Контрольные вопросы

1. Объясните понятие командной оболочки. Приведите примеры командных оболочек. Чем они отличаются? Командная оболочка — программа, через которую пользователь или администратор управляет операционной системой и установленными программами, используя командную строку. Примеры и отличия командных оболочек:
 - оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций; – C-оболочка (или csh) — надстройка на оболочкой Борна, использующая Сподобный синтаксис команд с возможностью сохранения истории выполнения команд; – оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна; – BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation).
2. Что такое POSIX? POSIX (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? Значениями переменной могут быть число, строка или отдельные символы. Пример определения переменной: `path="$HOME"` Чтение переменной: `"${path}"` Пример определения массива: `a=(1 2 3 4 5)` В массивах можно хранить числа и строки одновременно, например: `b=(1 2 "three" 4 "five")` Обращение к элементу массива с номером `i`: `${a[i]}` Чтобы вывести все элементы массива, требуется использовать символ `@`: `${a[@]}`
4. Каково назначение операторов `let` и `read`? `let` — это встроенная функция bash, которая

- позволяет производить базовые арифметические операции. Read — это встроенная команда bash, которая считывает строку из стандартного ввода (или из файлового дескриптора) и разбивает строку на слова.
5. Какие арифметические операции можно применять в языке программирования bash? Сложение, вычитание, умножение, деление, возведение в степень, остаток от деления.
 6. Что означает операция (())? Внутри двойных скобок записываются арифметические действия или условия.
 7. Какие стандартные имена переменных Вам известны? HOME — имя домашнего каталога пользователя. Если команда cd вводится без аргументов, то происходит переход в каталог, указанный в этой переменной. IFS — последовательность символов, являющихся разделителями в командной строке, например, пробел, табуляция и перевод строки (new line). MAIL — командный процессор каждый раз перед выводом на экран промптера проверяет содержимое файла, имя которого указано в этой переменной, и если содержимое этого файла изменилось с момента последнего ввода из него, то перед тем как вывести на терминал промптер, командный процессор выводит на терминал сообщение You have mail (у Вас есть почта). TERM — тип используемого терминала. LOGNAME — содержит регистрационное имя пользователя, которое устанавливается автоматически при входе в систему.
 8. Что такое метасимволы? Символы, которые имеют особое назначение. Например, * соответствует произвольной, в том числе и пустой строке.
 9. Как экранировать метасимволы? Экранирование может быть осуществлено с помощью предшествующего метасимволу символа \, который, в свою очередь, является метасимволом.
 10. Как создавать и запускать командные файлы? Последовательность команд может быть помещена в текстовый файл. Такой файл называется командным. Далее этот файл можно выполнить по команде: bash командный_файл [аргументы] Чтобы не вводить каждый раз последовательности символов bash, необходимо изменить код защиты этого командного файла, обеспечив доступ к этому файлу по выполнению. Это может быть сделано с помощью команды chmod +x имя_файла Теперь можно вызывать свой

командный файл на выполнение, просто вводя его имя с терминала.

11. Как определяются функции в языке программирования bash? Функция определяется с помощью ключевого слова `function`, после которого следует имя функции и список команд, заключённых в фигурные скобки.
12. Каким образом можно выяснить, является файл каталогом или обычным файлом? Если выражение `-d "$file"` истинно, то файл является каталогом. Если выражение `-f "$file"` истинно, то файл является файлом.
13. Каково назначение команд `set`, `typeset` и `unset`? `Set` — это встроенная команда оболочки, которая позволяет отображать или устанавливать переменные оболочки и среды. `Typeset` имеет четыре опции для работы с функциями: `-f` — перечисляет определённые на текущий момент функции; `-ft` — при последующем вызове функции иницирует её трассировку; `-fx` — экспортирует все перечисленные функции в любые дочерние программы оболочек; `-fu` — обозначает указанные функции как автоматически загружаемые. Автоматически загружаемые функции хранятся в командных файлах, а при их вызове оболочка просматривает переменную `FPATH`, отыскивая файл с одноимёнными именами функций, загружает его и вызывает эти функции. `Unset` удаляет переменную по имени до конца текущей сессии. Удалить функцию можно с помощью команды `unset` с флагом `-f`.
14. Как передаются параметры в командные файлы? При вызове командного файла через пробел. В тексте командного файла первый из них обозначается записью `$1`, второй записью `$2`, третий записью `$3` и т.д. до `$9`.
15. Назовите специальные переменные языка bash и их назначение. `$*` — отображается вся командная строка или параметры оболочки; `$?` — код завершения последней выполненной команды; `$$` — уникальный идентификатор процесса, в рамках которого выполняется командный процессор; `$!` — номер процесса, в рамках которого выполняется последняя вызванная на выполнение в командном режиме команда; `$-` — значение флагов командного процессора; `${#}` — возвращает целое число — количество слов, которые были результатом `$`; `${#name}` — возвращает целое значение длины строки в переменной `name`; `${name[n]}` — обращение к `n`-му элементу массива; `${name[*]}` — пе-

речисляет все элементы массива, разделённые пробелом; `${name[@]}` — то же самое, но позволяет учитывать символы пробелы в самих переменных; `${name:-value}` — если значение переменной `name` не определено, то оно будет заменено на указанное `value`; `${name:value}` — проверяется факт существования переменной; `${name=value}` — если `name` не определено, то ему присваивается значение `value`; `${name?value}` — останавливает выполнение, если имя переменной не определено, и выводит `value` как сообщение об ошибке; `${name+value}` — это выражение работает противоположно `${name-value}`. Если переменная определена, то подставляется `value`; `${name#pattern}` — представляет значение переменной `name` с удалённым самым коротким левым образцом (`pattern`); `${#name[*]}` и `${#name[@]}` — эти выражения возвращают количество элементов в массиве `name`.

5. Выводы

В процессе выполнения мной были изучены основы программирования в оболочке ОС UNIX/Linux и получены навыки написания небольших командных файлов.

Список литературы

<https://bestprogrammer.ru/izuchenie/komanda-set-linux?ysclid=lhklmm0xlv908627071> }