Лабораторная работа №12

Дисциплина: Операционные системы

Галанова Дарья Александровна

Содержание

1 Цель работы 5

2 Задание 6

3 Выполнение лабораторной работы 7

4 Библиография 17

5 Выводы 18

List of Tables

List of Figures

3.1 Созданиефайла 7
3.2 Скрипт№1..... 8
3.3 Проверкаработыскрипта..... 8
3.4 ИзменённыйскриптN1..... 9
3.5 ИзменённыйскриптN1..... 9
3.6 Проверкаработыскрипта..... 9
3.7 Реализация команды man 10
3.8 Реализация команды man 10
3.9 Созданиефайла 11
3.10 Скрипт№2..... 11
3.11 Проверкаработыскрипта..... 12
3.12 Созданиефайла 12
3.13 Скрипт№3..... 13
3.14 Проверкаработыскрипта..... 13

1 Цель работы

Изучить основы программирования в обо-лочке ОС UNIX. Научиться писать более сложные командные файлы с использо-ванием логических управляющих конструкций и циклов.

2 Задание

1. Сделать отчёт по лабораторной работе №12 в формате Markdown.
2. Научиться писать более сложные командные файлы с использованием ло-гических управляющих конструкций и циклов.

3 Выполнение лабораторной работы

1). Написала командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени t2<>t1, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Для данной задачи я создала файл: sem.sh (Рисунки 3.1) и написала соответствующий скрипт (алгоритм действий представлен на рис. 3.2).

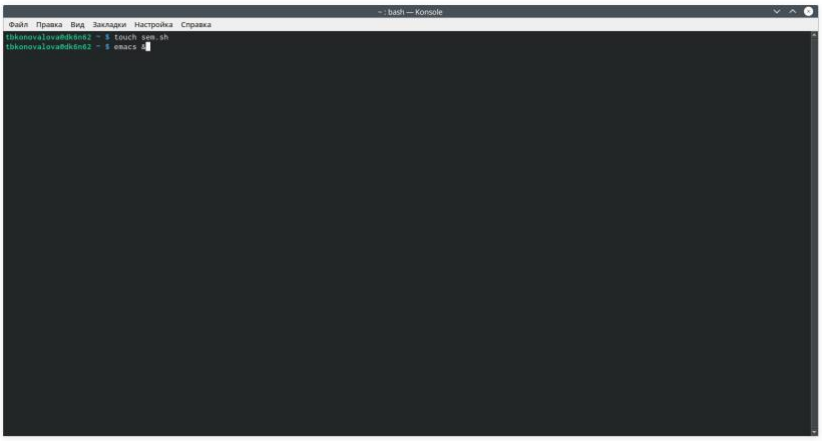

Figure 3.1: Создание файла

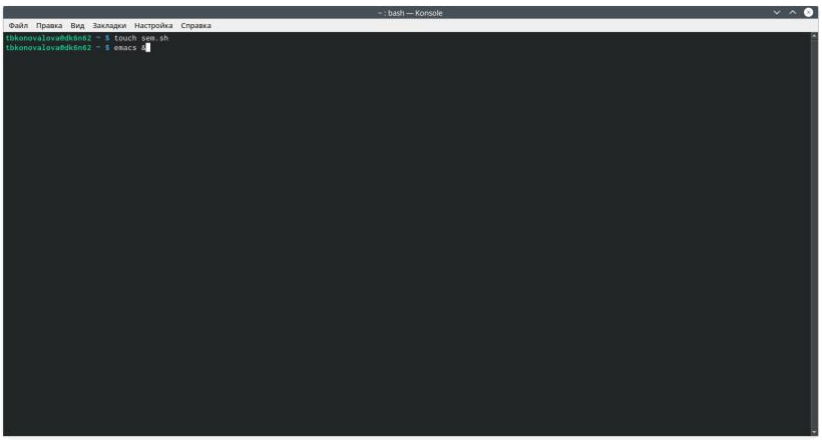

Figure 3.2: Скрипт №1

Далее я проверила работу написанного скрипта (команда«./sem.sh47»), пред-варительно добавив право на исполнение файла (команда «chmod+xsem.sh») (Скриншот 3.3). Скрипт работает корректно.

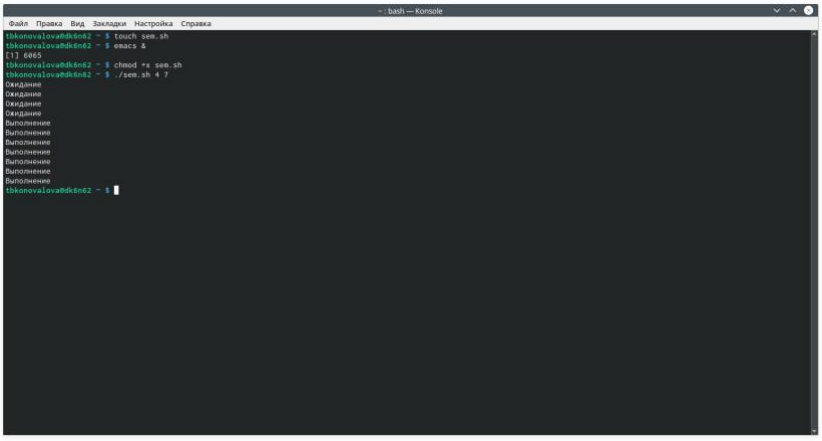

Figure 3.3: Проверка работы скрипта

После этого я изменила скрипт так, чтобы его можно было выполнять в несколь-ких терминалах и проверила его работу (например, команда «./sem.sh2 3 Ожида-ние > /dev/pts/1 &») (алгоритм действий представлен на рис. 3.4 , 3.5 , 3.6). Однако у меня не получилось проверить работу скрипта, так как было отказно в доступе.

Figure 3.4: Изменённый скрипт №1

Figure 3.5: Изменённый скрипт №1

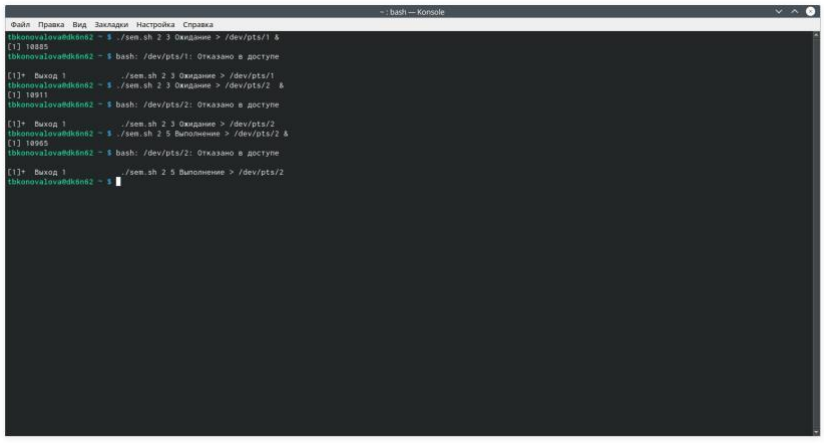

Figure 3.6: Проверка работы скрипта

2). Реализовала команду man с помощью командного файла. Изучила содержи-мое каталога /usr/share/man/man1 (Рисунки 3.7 , 3.8). В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в си-стеме программ и команд. Каждый архив можно открыть командой less сразу же просмотрев содержимое справки. Командный файл должен получать в виде аргумента командной строки название команды и в виде результата выдавать справку об этой команде или сообщение об отсутствии справки, если соответ-ствующего файла нет в каталоге man1.

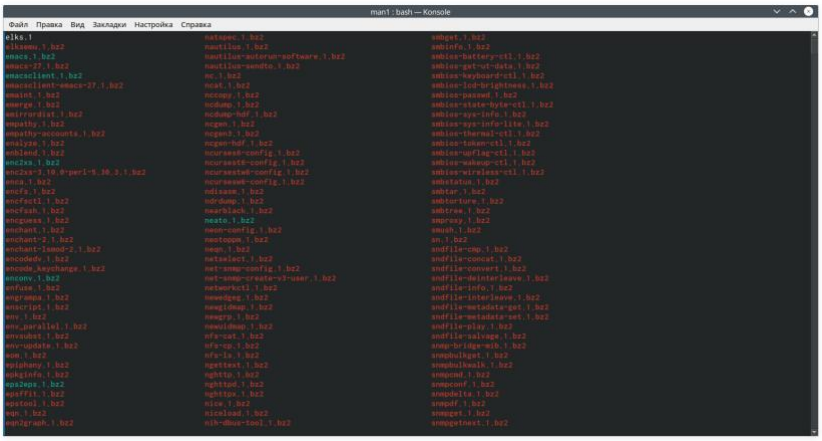

Figure 3.7: Реализация команды man

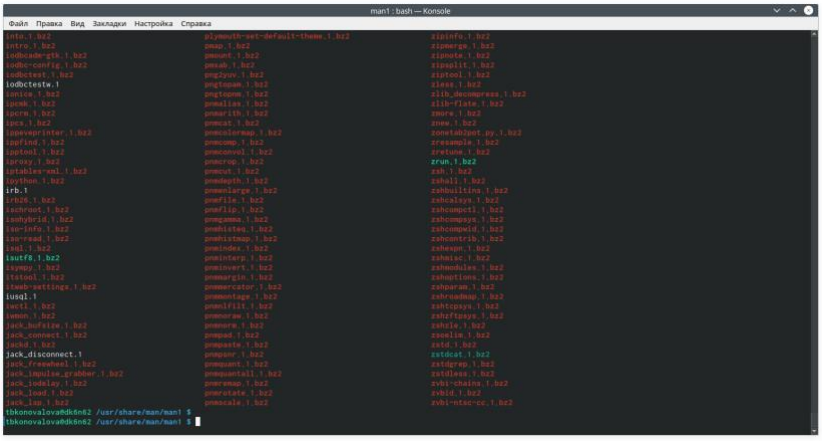

Figure 3.8: Реализация команды man

Для данной задачи я создала файл: man.sh (Рисунки 3.9) и написала соответ-

ствующий скрипт.

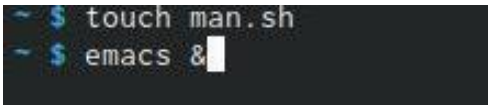

Figure 3.9: Создание файла

Далее я проверила работу написанного скрипта (команды«./man.shls»и «./man.sh mkdir»), предварительно добавив право на исполнение файла (команда «chmod +x man.sh») (Скриншот 3.10). Скрипт работает корректно.

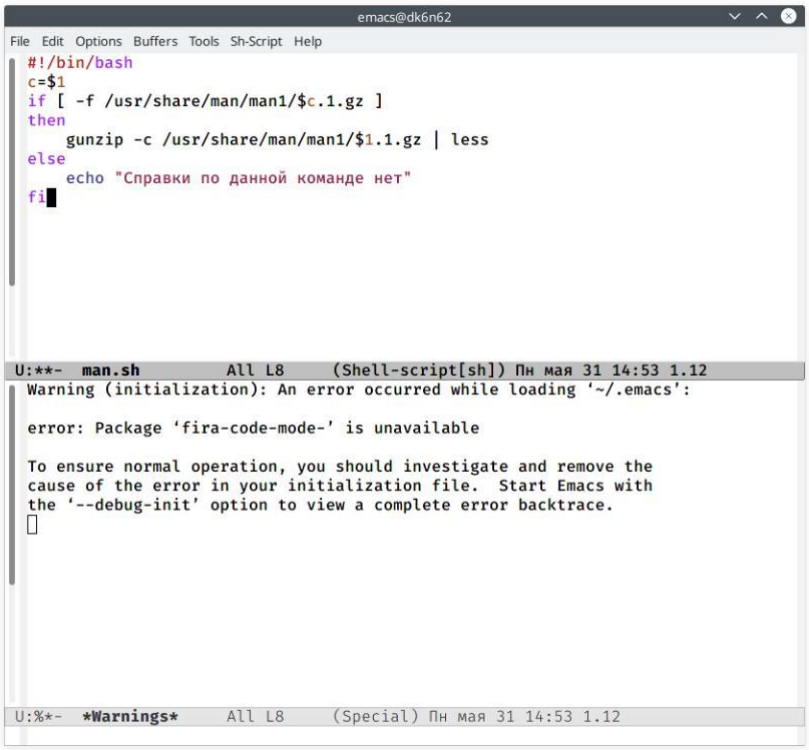

Figure 3.10: Скрипт №2

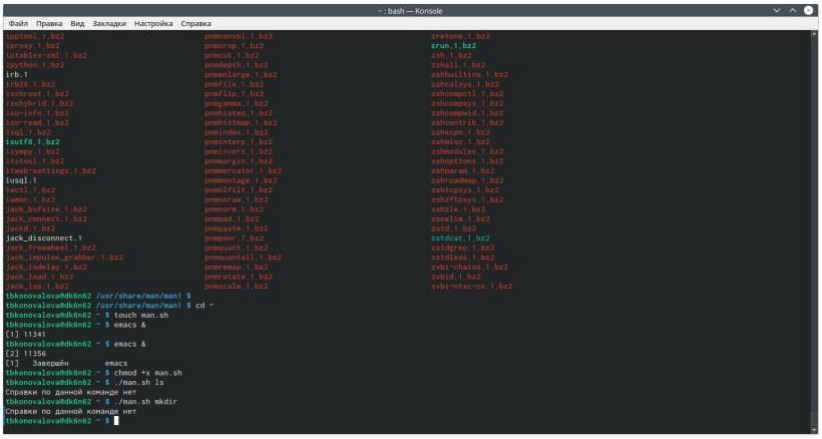

Figure 3.11: Проверка работы скрипта

3). Используя встроенную переменную $RANDOM, написала командный файл, генерирующий случайную последовательность букв латинского алфавита. Для данной задачи я создала файл: random.sh (Рисунок 3.12) и написала соответству-ющий скрипт (Рисунки 3.13).

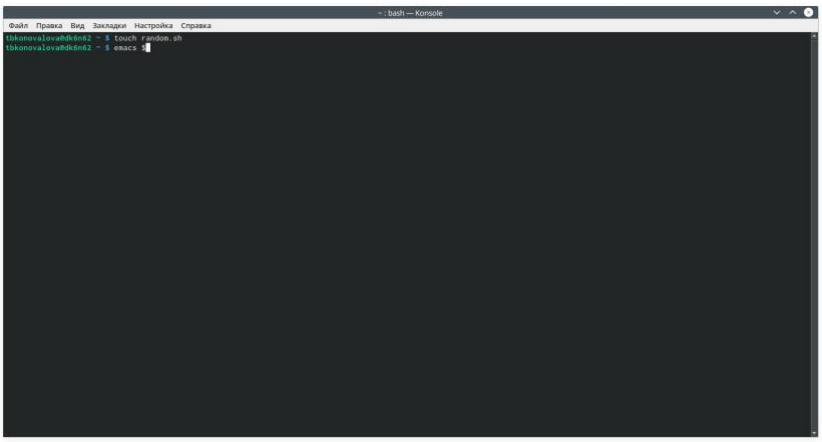

Figure 3.12: Создание файла

Figure 3.13: Скрипт №3

Далее я проверила работу написанного скрипта (команды «./random.sh 7»и «./random.sh 15»), предварительно добавив право на исполнение файла (команда «chmod +x random.sh») (Скриншот 3.14). Скрипт работает корректно.

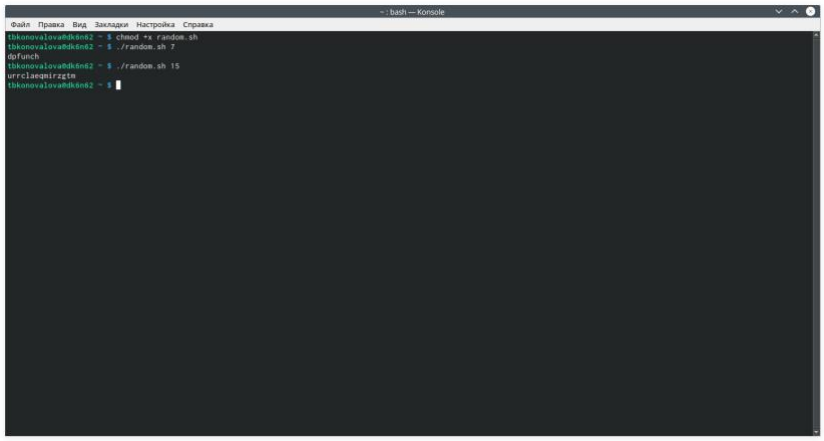

Figure 3.14: Проверка работы скрипта

Контрольные вопросы:

1). while [$1 != “exit”]

В данной строчке допущены следующие ошибки:

- не хватает пробелов после первой скобки [и перед второй скобкой]
- выражение $1 необходимо взять в “ ”, потому что эта переменная может содержать пробелы.

Таким образом, правильный вариант должен выглядеть так: while [“$1”!= “exit”] 2).

Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами:

- Первый:

VAR1=”Hello,

“VAR2=” World”

VAR3=“ 1VAR2”

echo “$VAR3”

Результат: Hello, World

- Второй:

VAR1=“Hello,”

VAR1+=” World”

echo “$VAR1”

Результат: Hello, World

3). Команда seq в Linux используется для генерации чисел от ПЕРВОГО до ПОСЛЕДНЕГО шага INCREMENT.

Параметры:

- seq LAST: если задан только один аргумент, он создает числа от 1 до LAST с шагом шага, равным 1. Если LAST меньше 1, значение is не выдает.

- seq FIRST LAST: когда заданы два аргумента, он генерирует числа от FIRST до LAST с шагом 1, равным 1. Если LAST меньше FIRST, он не выдает никаких выходных данных.

- seq FIRST INCREMENT LAST: когда заданы три аргумента, он генерирует числа от FIRST до LAST на шаге INCREMENT . Если LAST меньше, чем FIRST,

он не производит вывод.

- seq -f «FORMAT» FIRST INCREMENT LAST: эта команда используется для генерации последовательности в форматированном виде. FIRST и INCREMENT являются необязательными.

- seq -s «STRING» ПЕРВЫЙ ВКЛЮЧЕНО: Эта команда используется для STRING для разделения чисел. По умолчанию это значение равно /n. FIRST и INCREMENT являются необязательными.

- seq -w FIRST INCREMENT LAST:эта команда используется для выравнива-ния ширины путем заполнения начальными нулями. FIRST и INCREMENT являются необязательными.

4). Результатом данного выражения $((10/3))будет 3, потому что это целочис-ленное деление без остатка.

5). Отличия командной оболочки zshot bash:

- В zsh более быстрое автодополнение для cdc помощью Tab

- В zsh существует калькулятор zcalc, способный выполнять вычисления внут-ри терминала

- В zsh поддерживаются числа с плавающей запятой

- В zsh поддерживаются структуры данных «хэш»

- В zsh поддерживается раскрытие полного пути на основе неполных данных

- В zsh поддерживаетсязаменачастипути

- В zsh есть возможность отображать разделенный экран, такой же как разде-ленный экран vim

6). for((a=1; a<= LIMIT; a++)) синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать $ перед переменными ().

7). Преимущества скриптового языка bash:

- Один из самых распространенных и ставится по умолчаниюв большинстве дистрибутивах Linux, MacOS
- Удобное перенаправление ввода/вывода
- Большое количество команд для работы с файловыми системами Linux
- Можно писать собственные скрипты, упрощающие работу в Linux

Недостатки скриптового языка bash:

- Дополнительные библиотеки других языков позволяют выполнить больше действий
- Bash не является языков общего назначения
- Утилиты, при выполнении скрипта, запускают свои процессы, которые, в свою очередь, отражаются на быстроте выполнения этого скрипта
- Скрипты, написанные на bash, нельзя запустить на других операционных системах без дополнительных действий.

4 Библиография

1. Программное обеспечение GNU/Linux. Лекция 7. Вопросы лицензирования (Г. Курячий, МГУ);

2. Программное обеспечение GNU/Linux. Лекция 9. Хранилище и дистрибутив (Г. Курячий, МГУ);

3. Программное обеспечение GNU/Linux. Лекция 10. Минимальный набор знаний (Г. Курячий, МГУ);

4. Электронный ресурс: https://infopedia.su/24x10498.html

5. Электронный ресурс: http://5fan.ru/wie

5 Выводы

В ходе выполнения данной лабораторной работы я изучила основы программирования в оболочке ОС UNIX и научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.