Отчёт по лабораторной работе №12

Операционные системы

Самсонова Мария Ильинична

Содержание

[Цель работы 1](#_Toc104547245)

[Задание 1](#_Toc104547246)

[Выполнение лабораторной работы 2](#_Toc104547247)

[Вывод 14](#_Toc104547248)

[Ответы на контрольные вопросы 14](#_Toc104547249)

# Цель работы

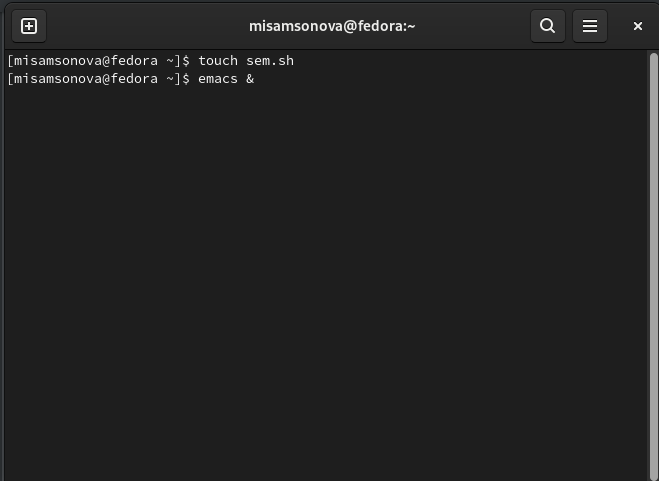
Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

# Задание

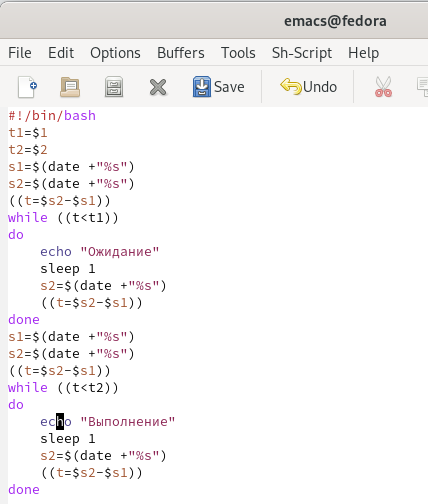
1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени t2<>t1, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запустить командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой (> /dev/tty#, где # — номер терминала куда перенаправляется вывод), в котором также запущен этот файл, но не фоновом, а в привилегированном режиме. Доработать программу так, чтобы имелась возможность взаимодействия трёх и более процессов.
2. Реализовать команду man с помощью командного файла. Изучите содержимое каталога /usr/share/man/man1. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе 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 соответствующего файла нет в каталоге man1.
3. Используя встроенную переменную $RANDOM, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтите, что $RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767.4.

# Выполнение лабораторной работы

1. Для начала написали 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 мы создали файл: sem.sh (рис. -@fig:001 ) и написали соответствующий скрипт (рис. -@fig:002 ).

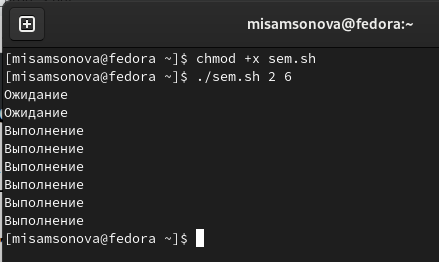


Создание файла



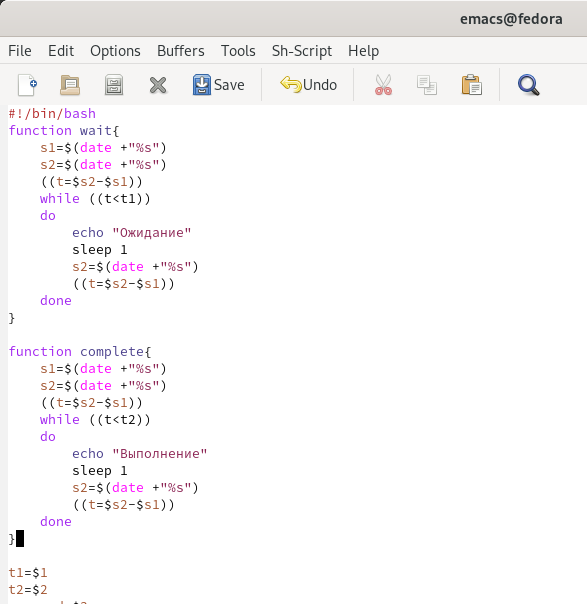
Скрипт №1

* Далее мы проверили работу написанного скрипта (команда«./sem.sh47»), предварительно добавив право на исполнение файла (команда **«chmod +x sem.sh»**) (рис. -@fig:003 ). Скрипт работает корректно.

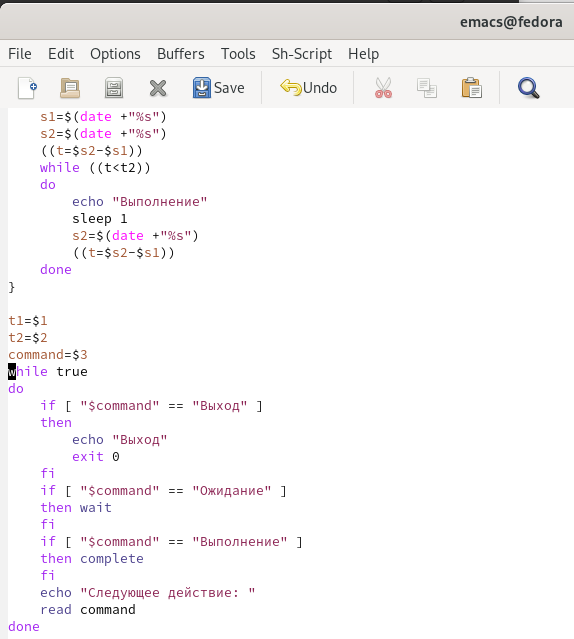


Проверка работы скрипта

* После этого мы изменили скрипт так, чтобы его можно было выполнить в нескольких терминалах и проверили его работу (например, команда **«./sem.sh 2 3 Ожидание > /dev/pts/1 &»**) (рис. -@fig:004 , -@fig:005 , -@fig:006 ). Однако у нас не получилось проверить работу скрипта, так как было отказно в доступе.



Изменённый скрипт №1

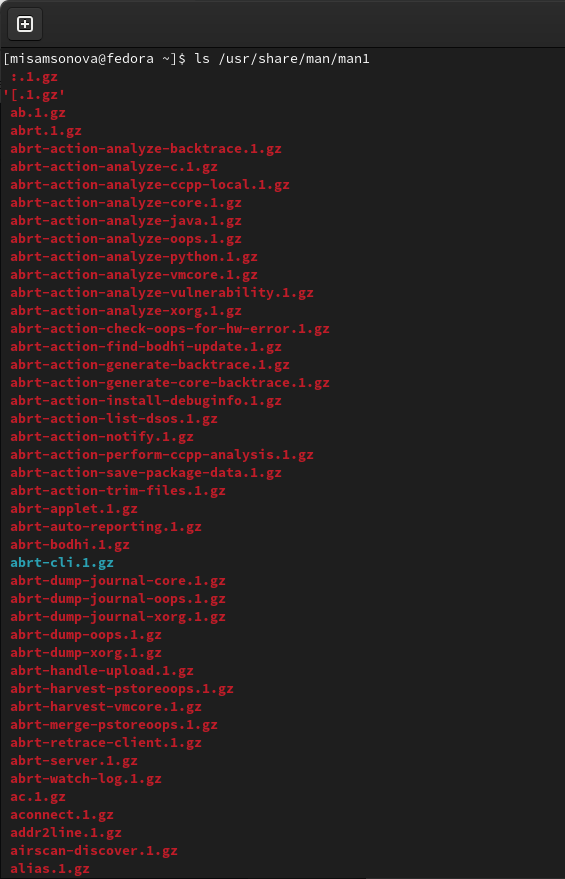


Изменённый скрипт №1

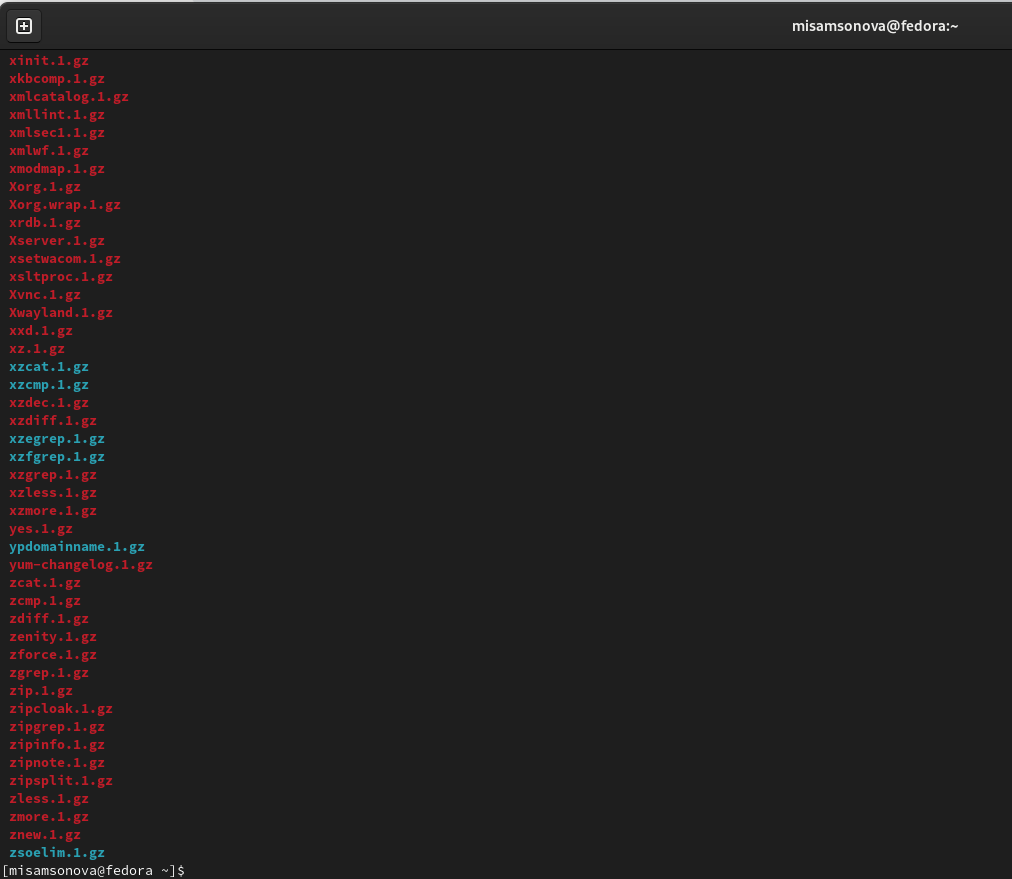


Проверка работы скрипта

1. Теперь реализовали команду man с помощью командного файла. Изучили содержимое каталога /usr/share/man/man1 (рис. -@fig:007 , -@fig:008 ). В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе 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 соответствующего файла нет в каталоге man1.

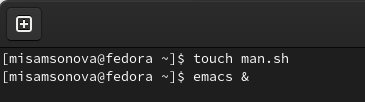


Реализацимы команды man



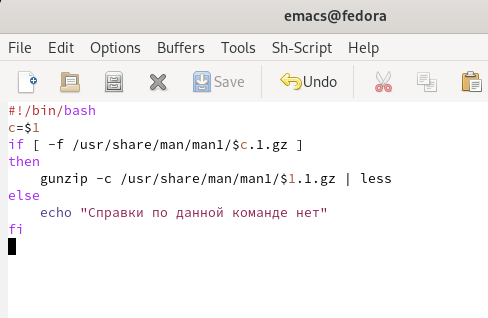
Реализацимы команды man

* Для данной задачи мы создали файл: man.sh (рис. -@fig:009 ) и написали соответствующий скрипт.

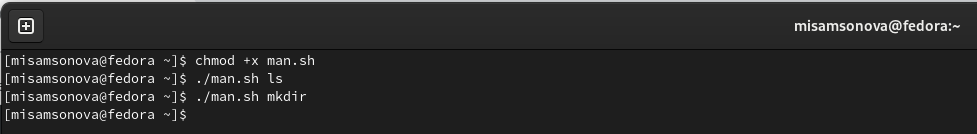


Создание файла

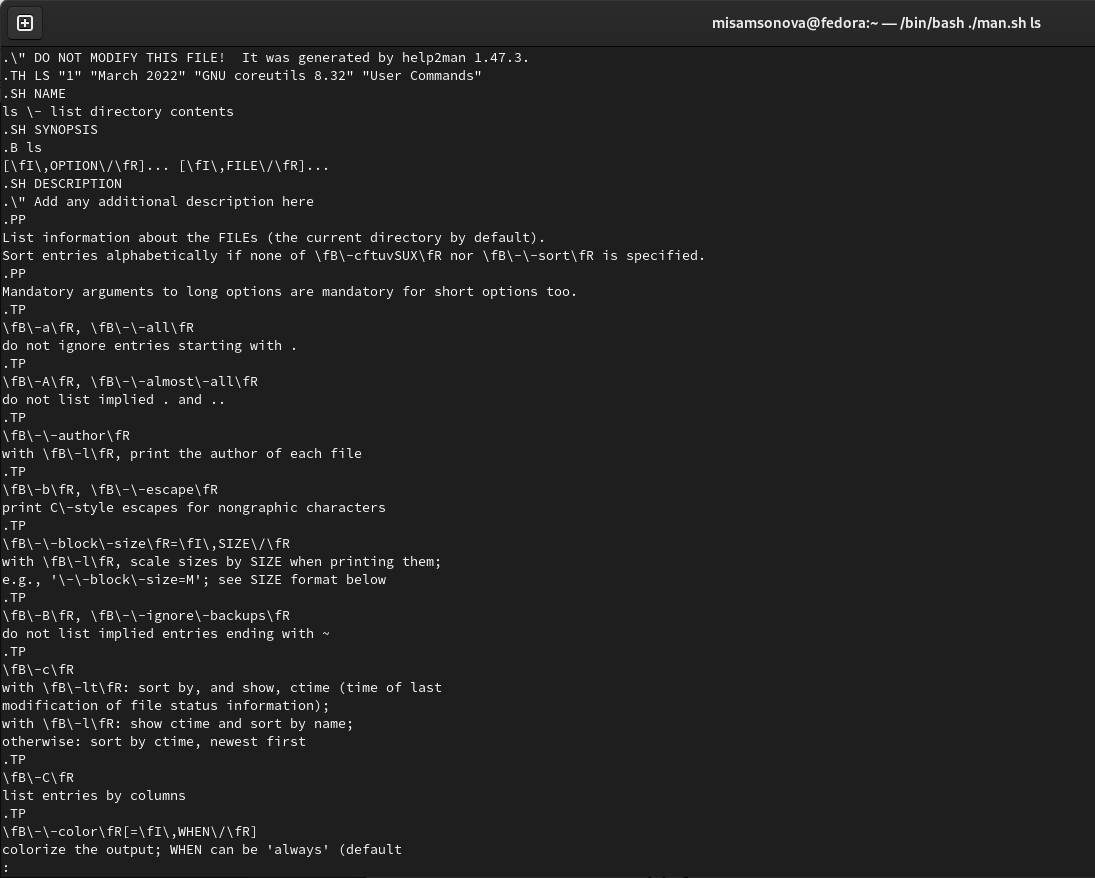
* Далее мы проверили работу написанного скрипта (команды **«./man.sh ls»** и **«./man.sh mkdir»**), предварительно добавив право на исполнение файла (команда **«chmod +x man.sh»**) (рис. -@fig:010 ). Скрипт работает корректно.



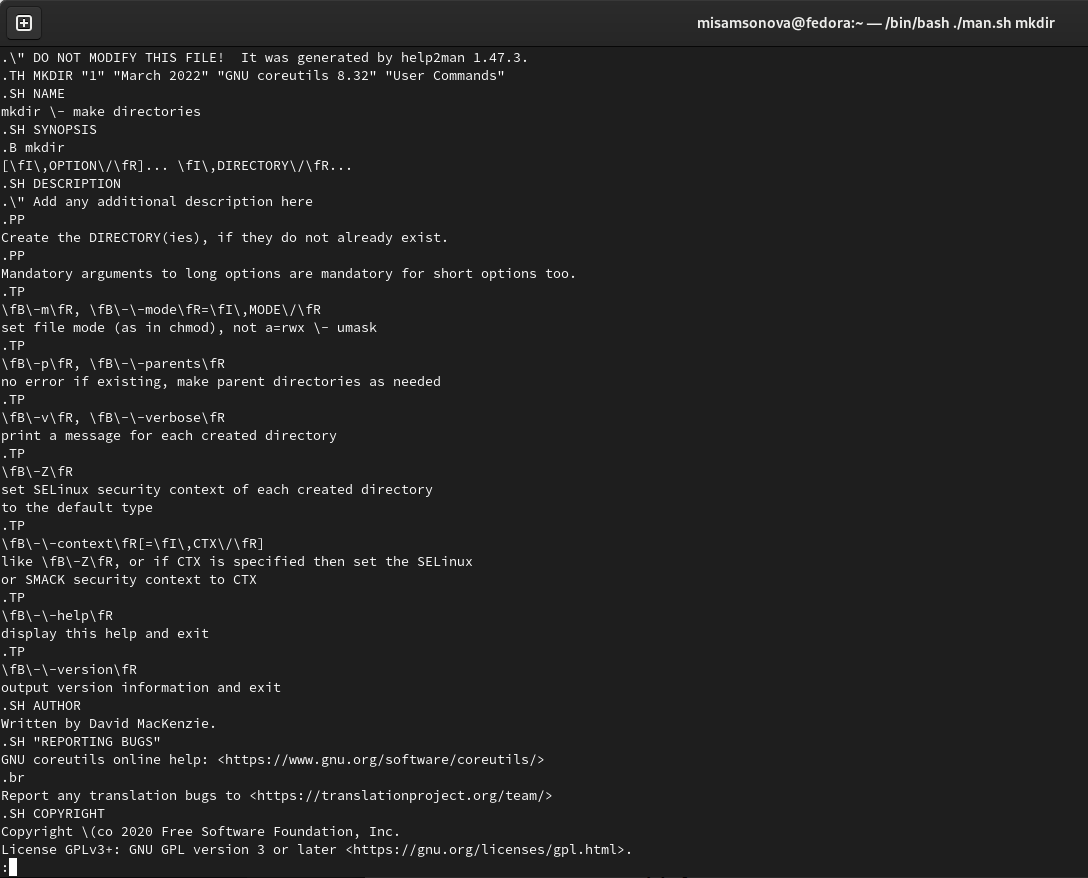
Скрипт №2



Проверка работы скрипта

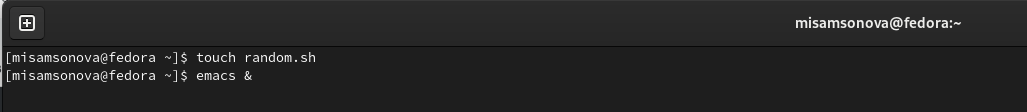


Проверка работы скрипта

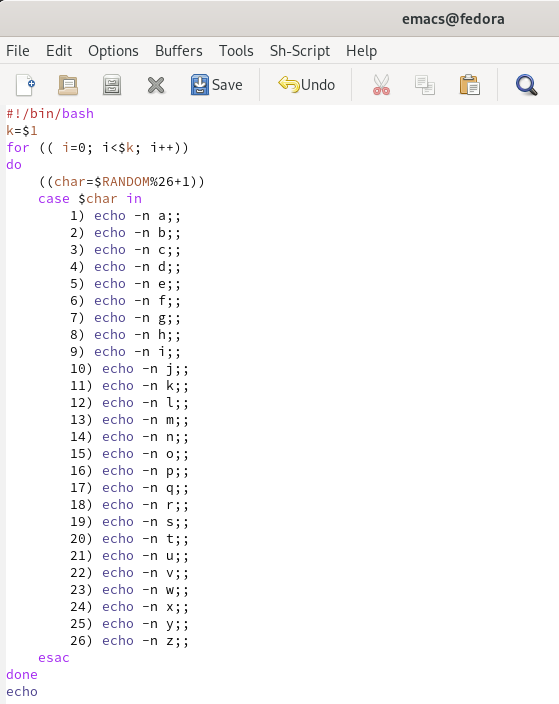


Проверка работы скрипта

1. Используя встроенную переменную $RANDOM, написали командный файл, генерирующий случайную последовательность букв латинского алфавита. Для данной задачи мы создали файл: random.sh (рис. -@fig:014 ) и написали соответствующий скрипт (рис. -@fig:015 ).

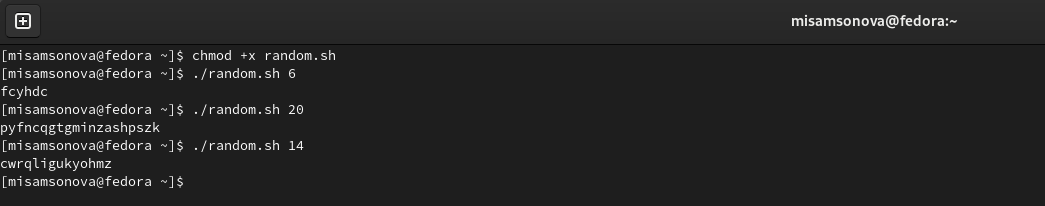


Создание файла



Скрипт №3

Далее мы проверили работу написанного скрипта (команды **«./random.sh 6»** и **«./random.sh 20»**), предварительно добавив право на исполнение файла (команда **«chmod +x random.sh»**) (рис. -@fig:014 ). Скрипт работает корректно.



Проверка работы скрипта

# Вывод

В процессе выполнения данной лабораторной работы мы изучили основы программирования в оболочке ОС UNIX и научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

# Ответы на контрольные вопросы

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”]

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

* Первый:

VAR1=“Hello,

“VAR2=” World”

VAR3=“VAR2”

echo “$VAR3”

Результат: Hello, World

* Второй:

VAR1=“Hello,”

VAR1+=” World”

echo “$VAR1”

Результат: Hello, World

1. 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 выравнивания ширины путем заполнения начальными нулями. FIRST и INCREMENT являются необязательными.

1. Результатом данного выражения $((10/3))будет 3, потому что это целочисленное деление без остатка.
2. Отличия командной оболочки zshот bash:

* В zsh более быстрое автодополнение для cdс помощью Тab
* В zsh существует калькулятор zcalc, способный выполнять вычисления внутри 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 разделенный экран vim

1. for((a=1; a<= LIMIT; a++)) синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать $ перед переменными ().
2. 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.