**Отчёт по лабораторной работе №13**

### Дисциплина: Операционные системы

## Джеффри Родригес Сантос

**Содержание**

1. [Цель работы 5](#_TOC_250004)
2. [Задание 6](#_TOC_250003)
3. [Выполнение лабораторной работы 8](#_TOC_250002)
4. [Контрольные вопросы 18](#_TOC_250001)
5. [Выводы 21](#_TOC_250000)

**Список таблиц**

# Цель работы

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

1. **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 осво- бождения, 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 вир- туальном 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 содер- жимое 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 содержимоесправки.Командныйфайлдолженполучать в виде аргумента командной строки название команды и в виде результата выдавать справку об этой командеилисообщение 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.
2. **Выполнениелабораторнойработы**
   1. Написал командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его осво- бождения, использовать его в течение некоторого времени t2<>t1, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Для данной задачи я создал файл: semafor.sh и написал соответствующий скрипт (рис. -fig. 3.1).

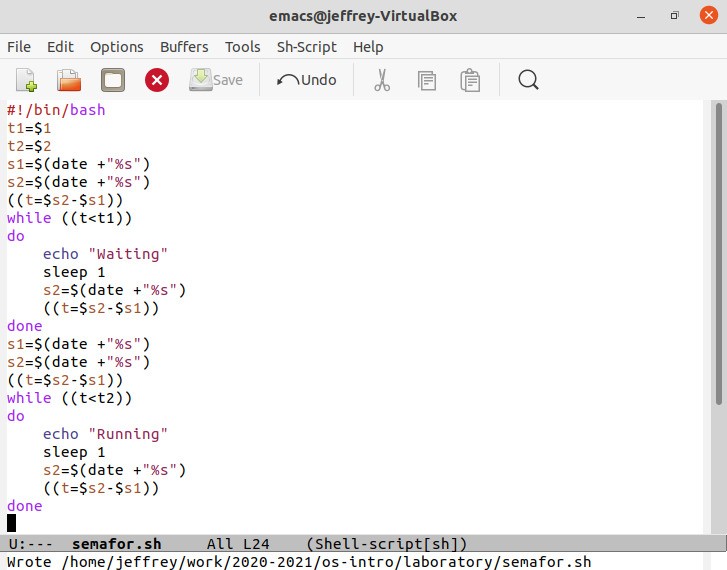


Рис. 3.1: Создал файл и написал первый скрипт

Далее я проверил работу написанного скрипта (команда «./semafor.sh 3 5»), предварительно добавив право на исполнение файла (команда «chmod +x semafor.sh») (рис. -fig. 3.2). Скрипт работает корректно.



Рис. 3.2: Проверил первый скрипт

После этого я изменил скрипт так, чтобы его можно было выполнять в несколь- ких терминалах (рис. -fig. 3.3) (рис. -fig. 3.4) и проверил его работу (например, команда «./semafor.sh 2 4 Ожидание > /dev/pts/1 &»).

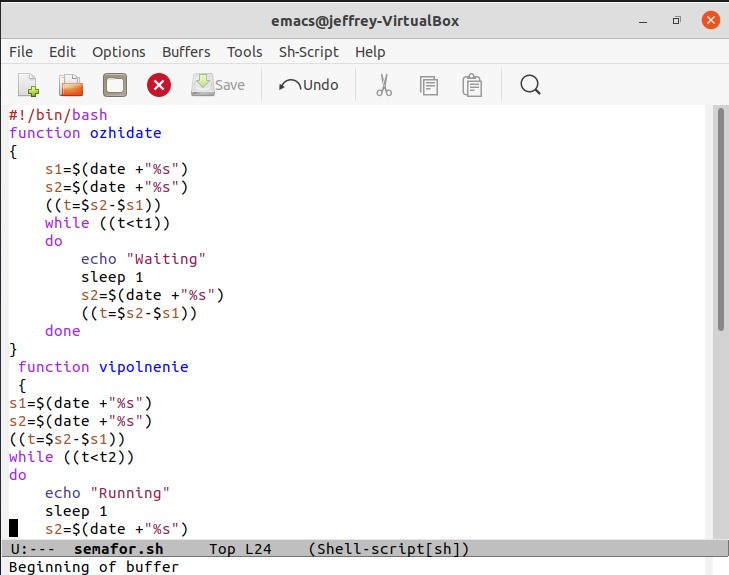
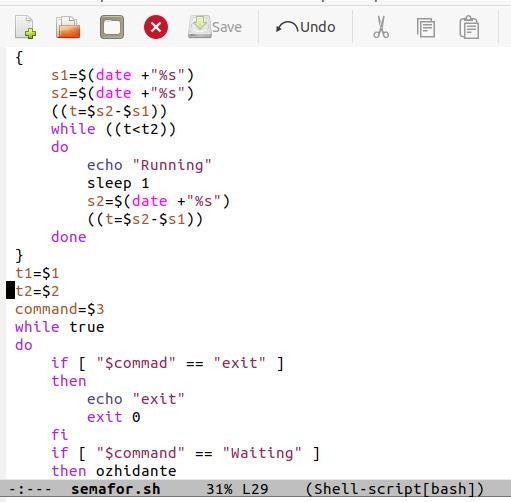


Рис. 3.3: изменил первый скрипт



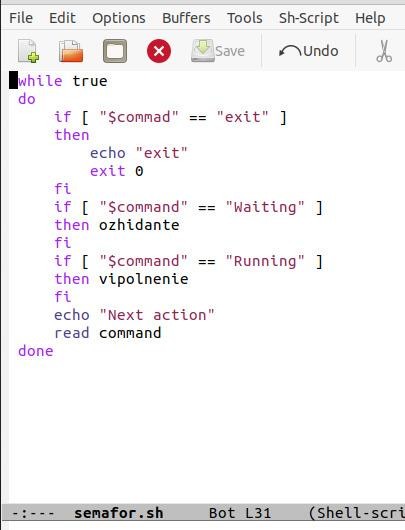
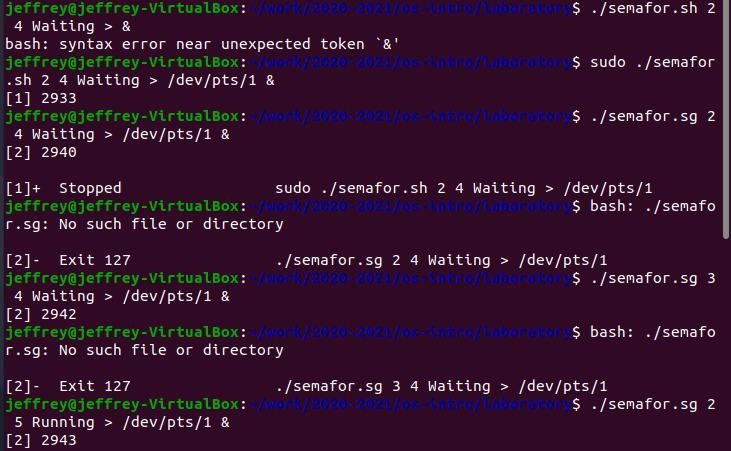


Рис. 3.4: Изменил первый скрипт

Но ни одна команда не работала, так как мне было “Отказано в доступе” (рис.-fig. 3.5). При этом скрипт работает корректно (команда «./semafor.sh 2 4 Ожида- ние»).



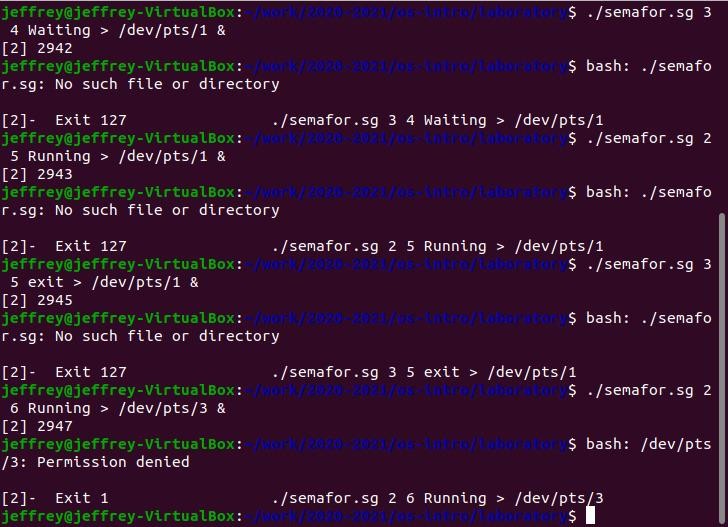


Рис. 3.5: проверил первый скрипт повторно

* 1. Реализовал командуmanспомощьюкомандного файла. Изучил содержимое каталога /usr/share/man/man1 (рис. -fig. 3.6). В нем находятся архивы текстовых файлов, содержащихсправкупо большинствуустановленных 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.

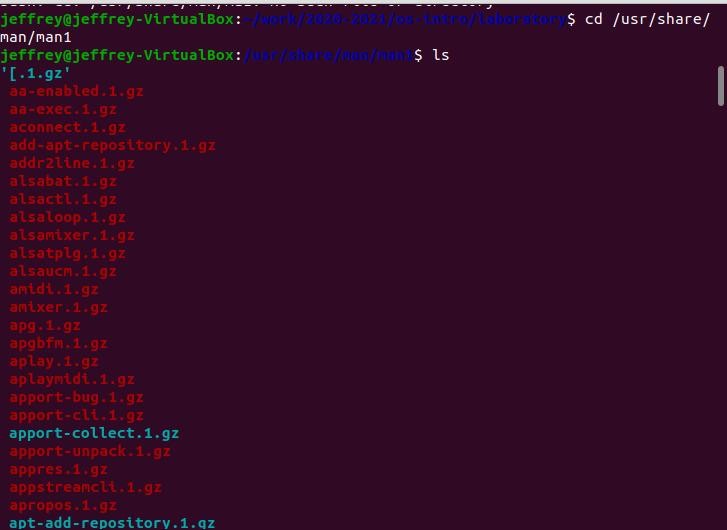


Рис. 3.6: Содержимое каталога /usr/share/man/man1

Для данной задачи я создал файл: man.sh и написал соответствующий скрипт (рис. -fig. 3.7).

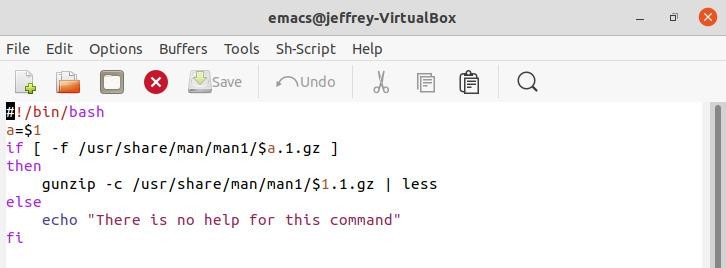


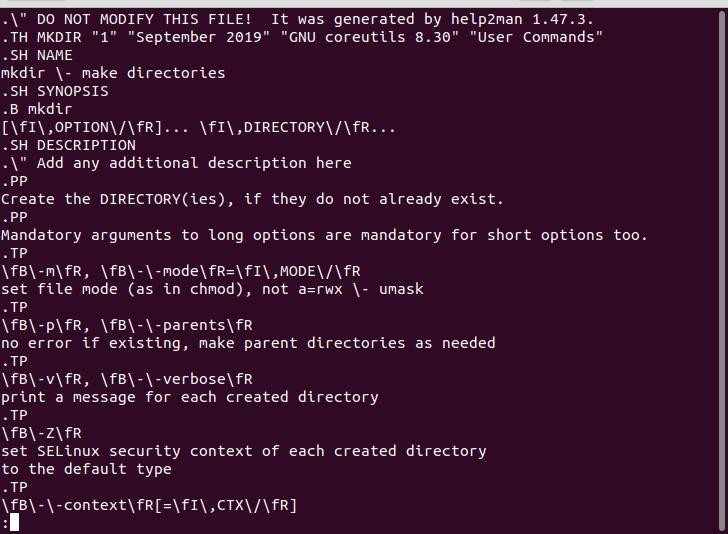
Рис. 3.7: Создал файл и написал второй скрипт

Далее я проверил работу написанного скрипта (команды «./man.sh mkdir»,«./man.sh rm» и «./man.sh car»), предварительно добавив право на исполнение файла (команда «chmod+xman.sh»)(рис.-fig. 3.8)(рис.-fig. 3.9)(рис.- fig. 3.10).

Скрипт работает корректно.



Рис. 3.8: Проверил второй скрипт



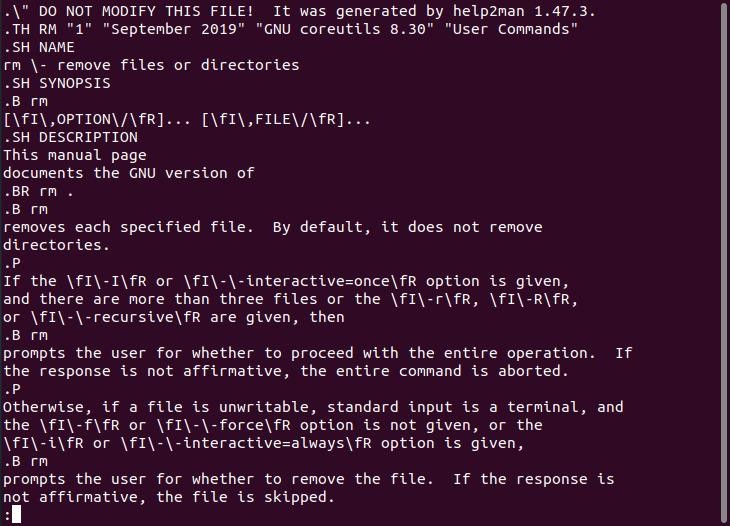


Рис. 3.10: Проверил второй скрипт

* 1. Используя встроенную переменную $RANDOM, написал командный файл, генерирующий случайную последовательность букв латинского алфавита. Для данной задачи я создал файл: random.sh и написал соответствующий скрипт (рис. -fig. 3.11).

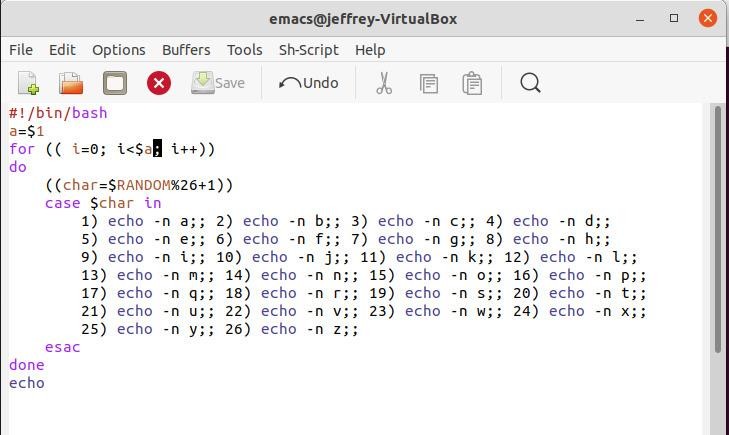


Рис. 3.11: Создал файл и написал третий скрипт

Далее я проверил работу написанного скрипта (команда «./random.sh 15»), предварительно добавив право на исполнение файла (команда «chmod +x random.sh») (рис. -fig. 3.12). Скрипт работает корректно.

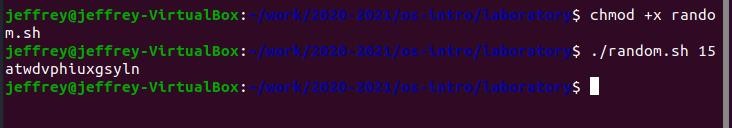


Рис. 3.12: Проверил третий скрипт

1. **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 несколь- кими способами:
* Первый:

VAR1=“Hello,”

V𝐴AR𝑅3= “𝑉 echo “$VAR3”

VAR2=1”VWAoRr2l”d”

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 выравнива- ния 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 zsh отbash:
* В zsh более быстрое автодополнение для cd с помощью Тab
* Вzshсуществуеткалькуляторzcalc, способный выполнять вычисления внут- ри терминала
* В zsh поддерживаются числа с плавающейзапятой
* В zsh поддерживаются структуры данных «хэш»
* В zsh поддерживается раскрытие полного пути на основенеполных данных
* В zsh поддерживается замена части пути
* 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 умолчанию в 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

1. **Выводы**

В ходе выполнения данной лабораторной работы я изучил основы программиро- вания в оболочке ОС UNIX, а также научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.