Шаблон отчёта по лабораторной работе 13

дисциплина: Операционные системы

Студент: Леон Атупанья Хосе Фернандо

Содержание

[РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ 1](#_Toc74637858)

[Факультет физико-математических и естественных наук 1](#_Toc74637859)

[Кафедра прикладной информатики и теории вероятностей 1](#_Toc74637860)

[ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 13 1](#_Toc74637861)

[Цель работы 1](#_Toc74637862)

[Задания 1](#_Toc74637863)

[Выполнение лабораторной работы 2](#_Toc74637864)

[Контрольные вопросы 8](#_Toc74637865)

[Выводы 9](#_Toc74637866)

## РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

## Факультет физико-математических и естественных наук

### Кафедра прикладной информатики и теории вероятностей

### ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 13

*дисциплина: Операционные системы*

**Студент: Хосе Фернадо Леон Атупанья** **Группа: НПМбд-02-20**

МОСКВА 2021 г.

# Цель работы

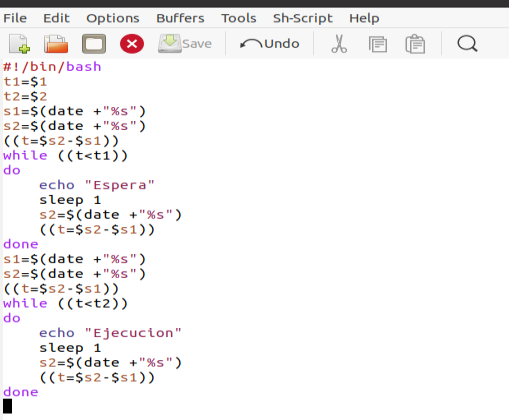
Изучить основы программирования в оболочке операционной системы UNIX. Узнайте, как писать более сложные пакетные файлы, используя логические конструкции управления и циклы.

# Задания

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.

# Выполнение лабораторной работы

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 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 создал файл: semafor.sh и написал соответствующий скрипт (рис. @fig:001).



Используя semafor.sh

Далее я проверил работу написанного скрипта (команда «./semafor.sh 3 5»), предварительно добавив право на исполнение файла (команда «chmod +x semafor.sh») (рис. @fig:002). Скрипт работает корректно.



Провиряем semafor.sh

После этого я изменил скрипт так, чтобы его можно было выполнять в нескольких терминалах (рис. @fig:003) и проверил его работу (например, команда «./semafor.sh 2 4 Ожидание > /dev/pts/1 &»).

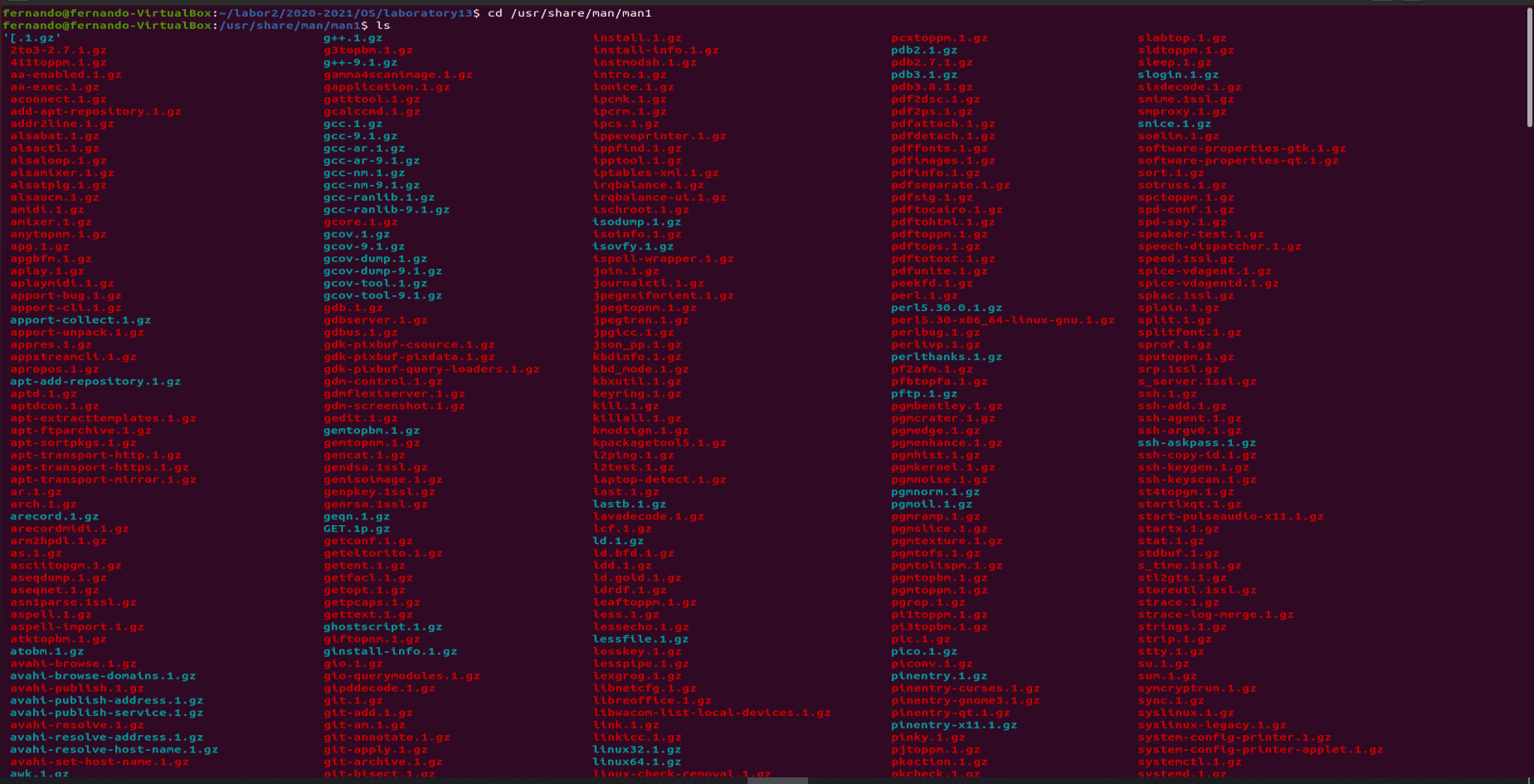
[Изменяем semafor.sh] semafor.sh](imag13/13.2.png){ #fig:001 width=70% }

К сожелению ни одна команда не работала, так как мне было “Отказано в доступе” (рис. @fig:004). При этом скрипт работает корректно (команда «./semafor.sh 2 4 Ожидание»).



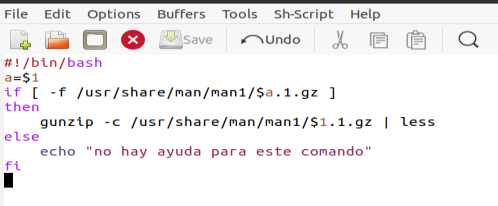
Провиряем semafor.sh

1. Реализовал команду man с помощью командного файла. Изучил содержимое каталога /usr/share/man/man1 (рис. @fig:005). 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.



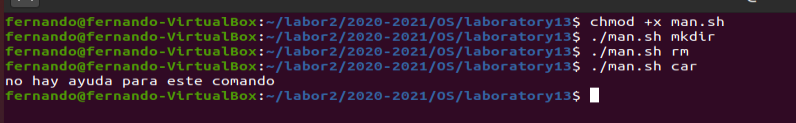
ls /usr/share/man/man1

Для данной задачи я создал файл: man.sh и написал соответствующий скрипт (рис. @fig:006).

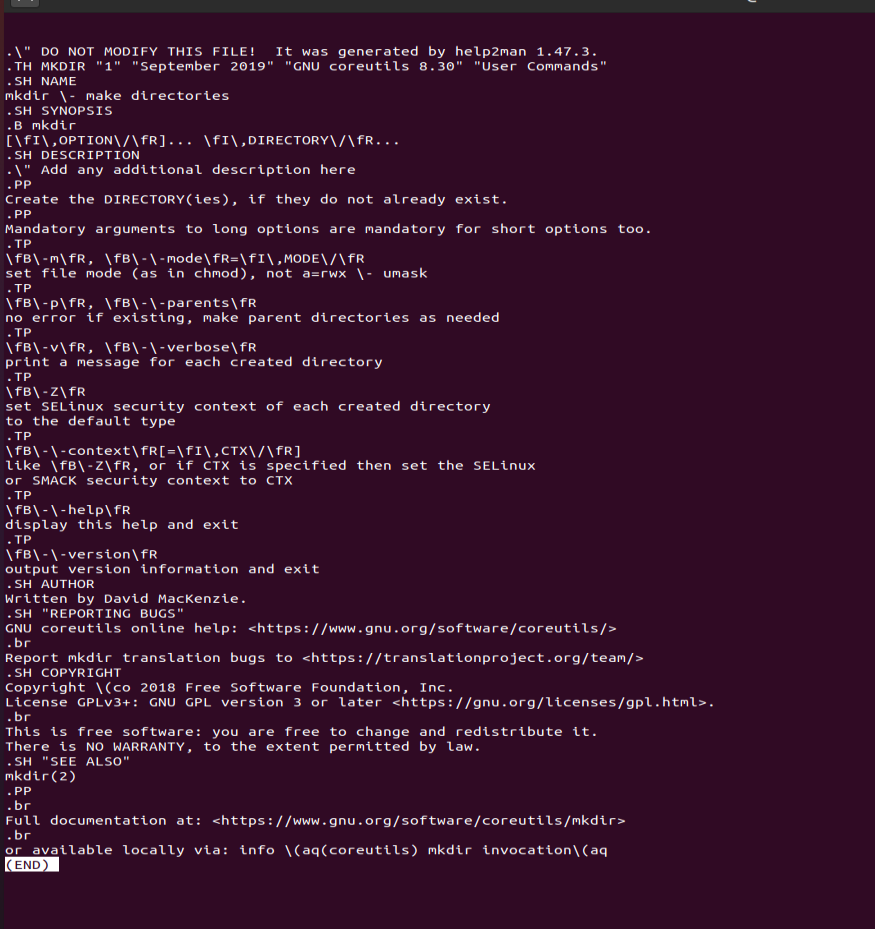


Пишем скрипт

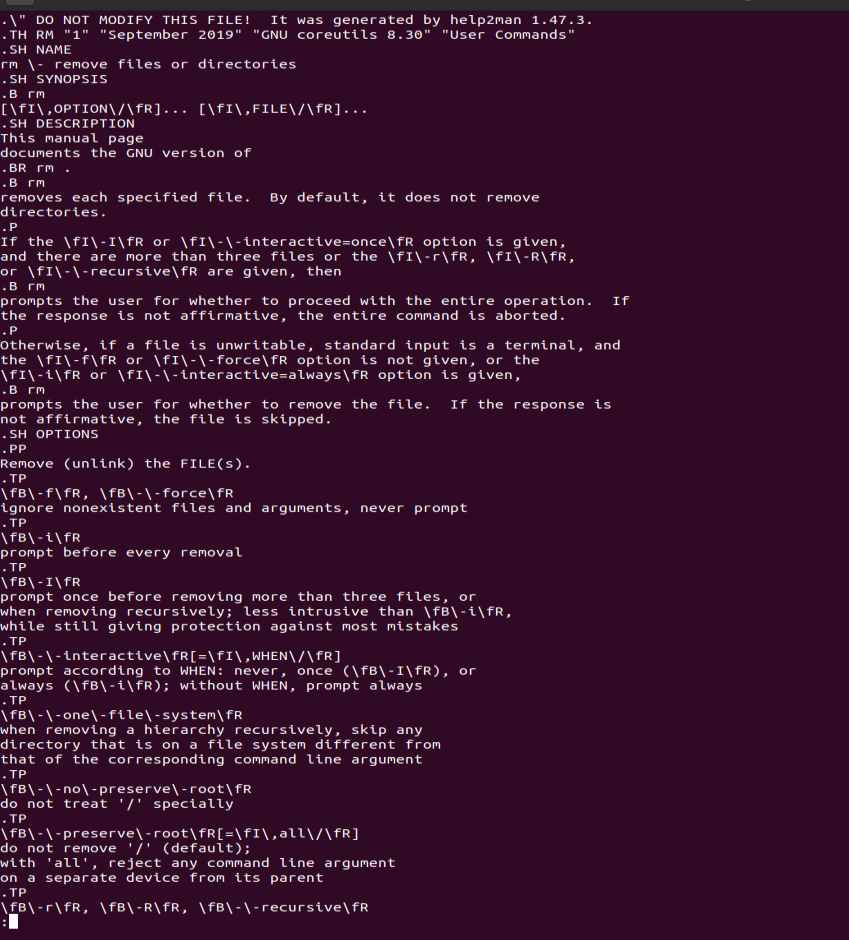
Далее я проверил работу написанного скрипта (команды «./man.sh mkdir», «./man.sh rm» и «./man.sh car»), предварительно добавив право на исполнение файла (команда «chmod +x man.sh») (рис. @fig:007) (рис. @fig:008) (рис. @fig:009). Скрипт работает корректно.



СКРИПТ РАБОТАЕТ

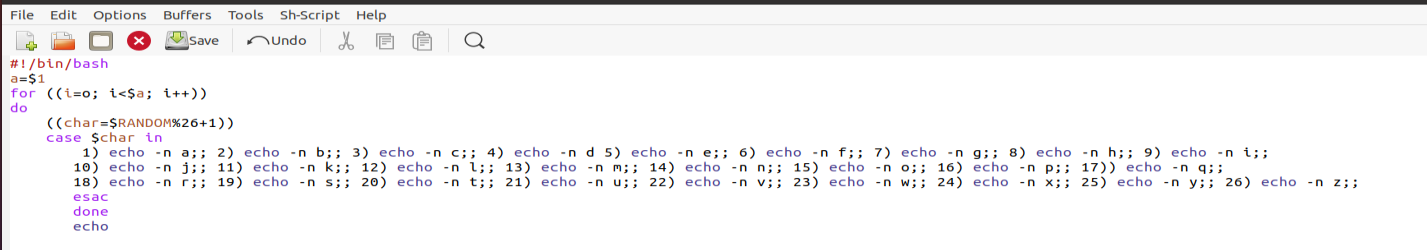


СКРИПТ РАБОТАЕТ



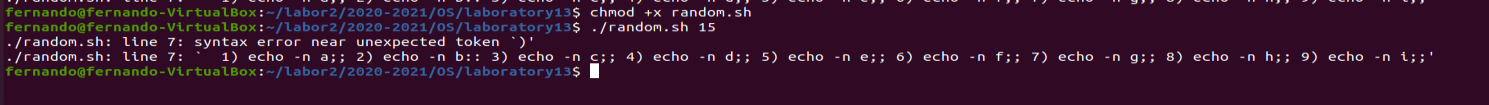
СКРИПТ РАБОТАЕТ

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:010).



Пишем скрипт

Далее я проверил работу написанного скрипта (команда «./random.sh 15»), предварительно добавив право на исполнение файла (команда «chmod +x random.sh») (рис. @fig:011). Скрипт работает корректно.



СКРИПТ РАБОТАЕТ

# 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 zsh от bash:

* В zsh более быстрое автодополнение для cd с 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 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

# Выводы

Во время этой лабораторной работы я изучил основы программирования в оболочке операционной системы UNIX, а также научился писать более сложные пакетные файлы с использованием логических конструкций управления и циклов.