Лабораторная работа №13

Бабков Дмитрий Николаевич

Содержание

[Цель работы 1](#_Toc73800290)

[Задание 1](#_Toc73800291)

[Ход работы 1](#_Toc73800292)

[Контрольные вопросы 5](#_Toc73800293)

[Вывод 6](#_Toc73800294)

# Цель работы

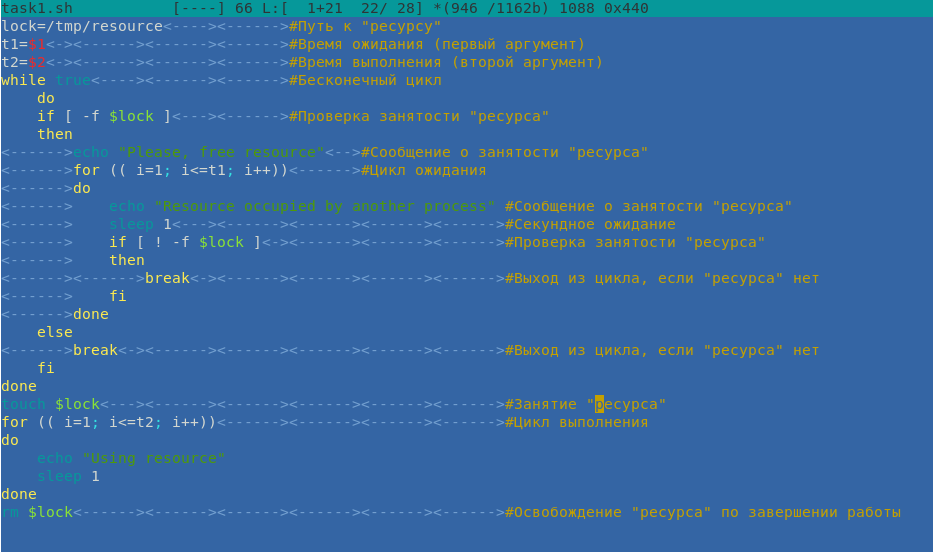
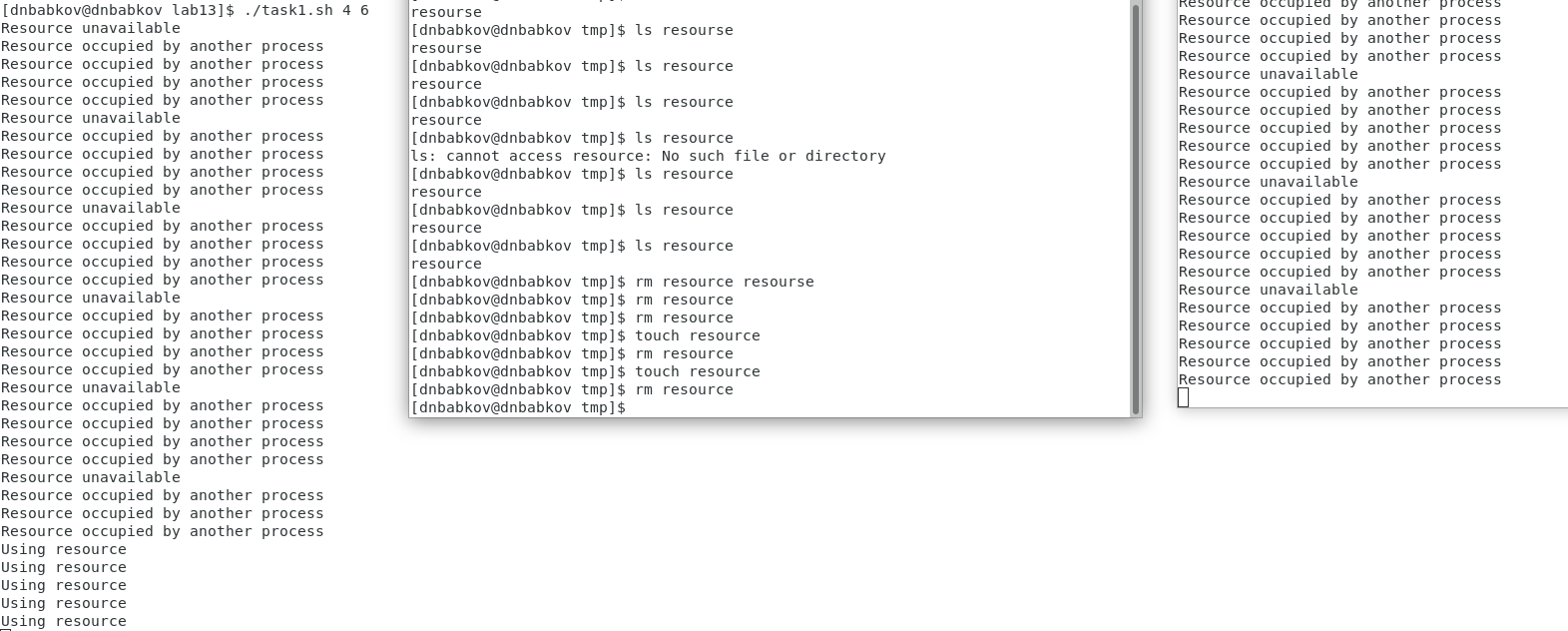
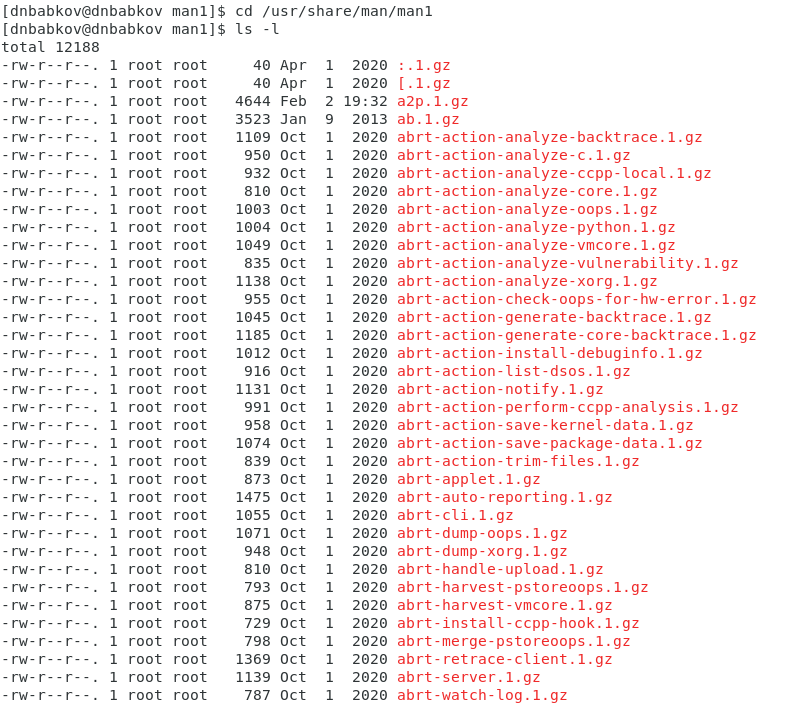
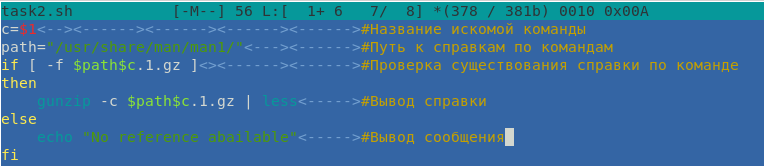
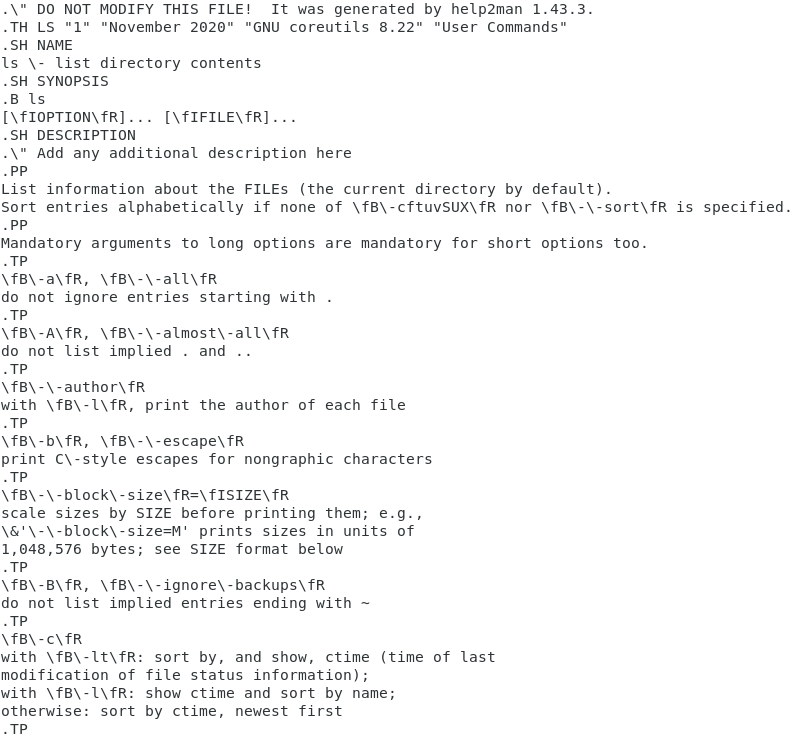
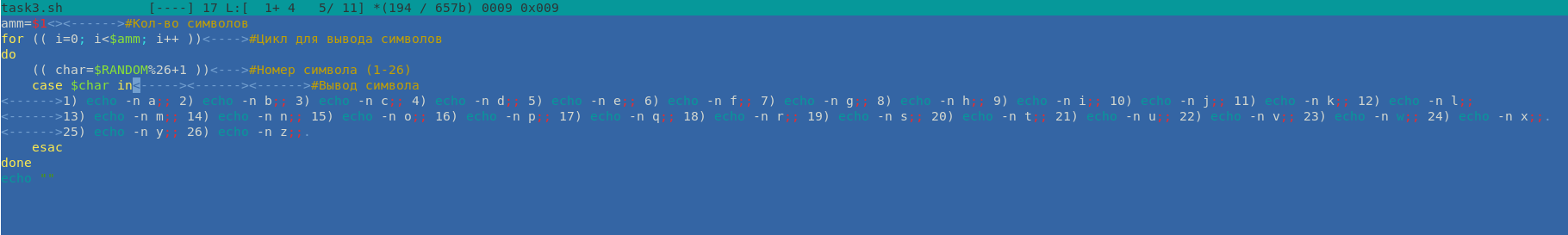
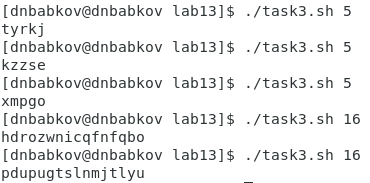
Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

# Задание

Написать командные файлы, выполняющие действия, описанные в задании

# Ход работы

Необходимо выполнить следующие 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.
   * Написал командный файл, которая принимает на вход два параметра - время ожидания и время выполнения. Если командный файл за время ожидания не может получить доступ к “ресурсу”, он выведет сообщение об этом и начнет ожидание заново. После того как “ресурс” стал доступен, командный файл его занимает и начинает в течение времени **t2** выводит сообщение о своей работе. После завершения “ресурс” освобождается. Код командного файла: (Рис.1)  
      Рис.1
   * Результат работы командного файла в нескольких терминалах (предварительно дал файлу разрешение на выполнение с помощью команды **chmod +x имяфайла**) (Рис.2):  
      Рис.2
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.
   * Перед написанием командного файла я изучил содержимое каталога **/usr/share/man/man1** (Рис.3)  
      Рис.3
   * Написал командный файл, который принимает на вход название команды, находит соответствующий архив в указанном выше каталоге, распаковывает его и выводит содержимое справки с помощью конвейера и команды **less** и, если такой справки нет, выводит соответствующее сообщение (Рис.4)  
      Рис.4
   * Результат работы командного файла для команды **ls** (Рис.5)  
      Рис.5
3. Используя встроенную переменную **$RANDOM**, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтите, что эта переменная выдаёт псевдослучайные числа в диапазоне от 0 до 32767.
   * Написал командный файл, который принимает на вход одно число - кол-во выводимых символов. Затем с помощью цикла **for** и оператора **case** выводятся символы в зависимости от значений переменной **$RANDOM**, которая принимает значения от 1 до 26 (Рис.6)  
      Рис.6
   * Результат работы командного файла: (Рис.7)  
      Рис.7

# 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 [ “$1” != “exit” ]
2. Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами:
   * Первый:  
     VAR1=“Hello,”  
     VAR2=" World"  
     VAR3=“VAR2”  
     echo “$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 выравнивания ширины путем заполнения начальными нулями. FIRST и INCREMENT являются необязательными.
4. Результатом выражения $((10/3)) будет 3, потому что это целочисленное деление без остатка.
5. 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 основе неполных данных
   * В zsh поддерживается замена части пути
   * В zsh есть возможность отображать разделенный экран, такой же как разделенный 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 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

# Вывод

В ходе выполнения данной лабораторной работы я научился писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.