Отчет по лабораторной работе 12

Лабораторная работа 12

Куркина Евгения Вячеславовна

Содержание

# 1 Цель работы

Здесь приводится формулировка цели лабораторной работы. Формулировки цели для каждой лабораторной работы приведены в методических указаниях.

Цель данной лабораторной работы — Изучить основы программирования в оболочке OC UNIX. Научиться писать более сложные командные файлыс использованием логических управляюзих конструкций и циклов.

# 2 Задание

Написать необходимые командные файлы.

# 3 Выполнение лабораторной работы

1. Создала отдельный файл sem.sh(рис. 1). Написала командный файл, который реализует упращенный механизм семафоров.Команда должна дожидаться освобождения ресурса, выдавая об этом сообщение, а затем использовать его в течении некторого времени, также выдавая информацию об этом (рис. 2).Затем дала права на выполнение, а затем проверила результат работы файла (рис. 3).

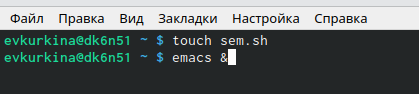


Рис. 1: Создание файла, переход в emacs

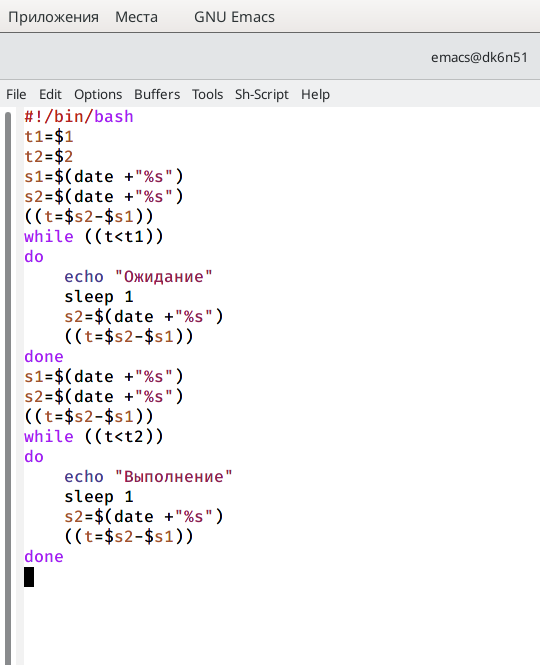


Рис. 2: Текст скрипта

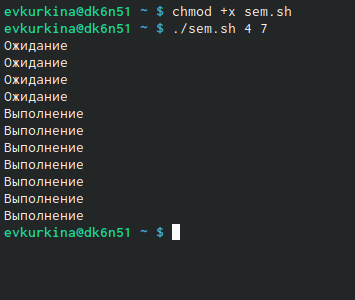


Рис. 3: Права на выполнение и результат работы файла

1.2) Изменила текст скрипта так, чтобы его можно было выполнять сразу в нескольких терминалах (рис. 4)(рис. 5), затем проверила его работу (нет прав доступа для данной команды) (рис. 6).



Рис. 4: Измененный текст скрипта ч1

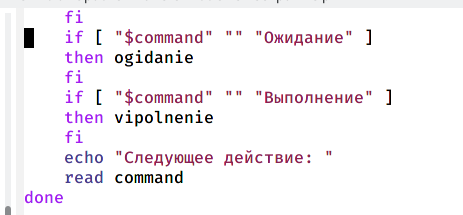


Рис. 5: Измененный текст скрипта ч2

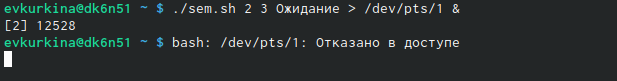


Рис. 6: Проверка командного файла

1. Перешла в каталог man1 (рис. 7), и командой ls просмотрела содержимое (рис. 8). Далее создала файл man.sh (рис. 9). Написала текст скрипта, который получает в виде аргументаназвание команды, и выдавать справку об этой команде или сообщение о том, что справка отсутствует (рис. 10). Затем дала права на исполнение и проверила работу командного файла,получив справку о команде rm (рис. 11) (рис. 12).

Рис. 7: Путь в каталог

Рис. 7: Путь в каталог

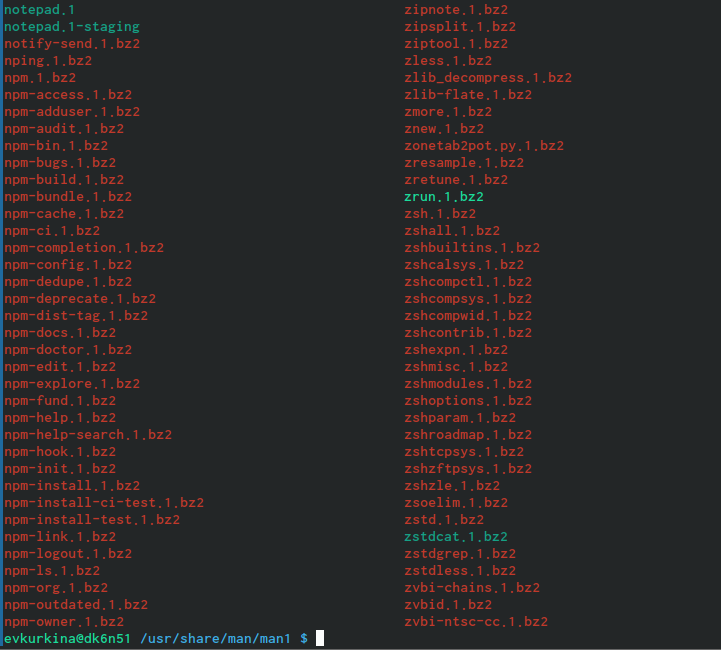


Рис. 8: Список файлов

Рис. 9: Создание нового файла и переход в emacs

Рис. 9: Создание нового файла и переход в emacs

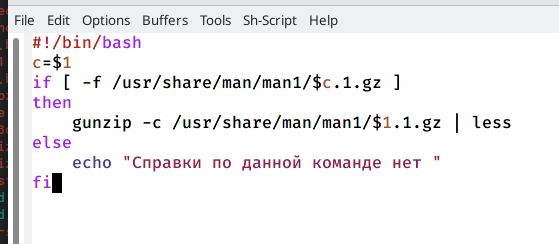


Рис. 10: Текст команндного файла

Рис. 11: Права достпуа, команды проверки файла

Рис. 11: Права достпуа, команды проверки файла



Рис. 12: Справка по команде

1. Создала новый файл с названием random.sh (рис. 13). Написала текст командного файла, кторой с помощью встроенной переменной $RANDOM генерирует случайную последовательностьбукв латинского алфавита (рис. 14). Дала права доступа на исполнение и затем проверила работу скрипта (рис. 15).

Рис. 13: Создание нового файла для скрипта 3

Рис. 13: Создание нового файла для скрипта 3

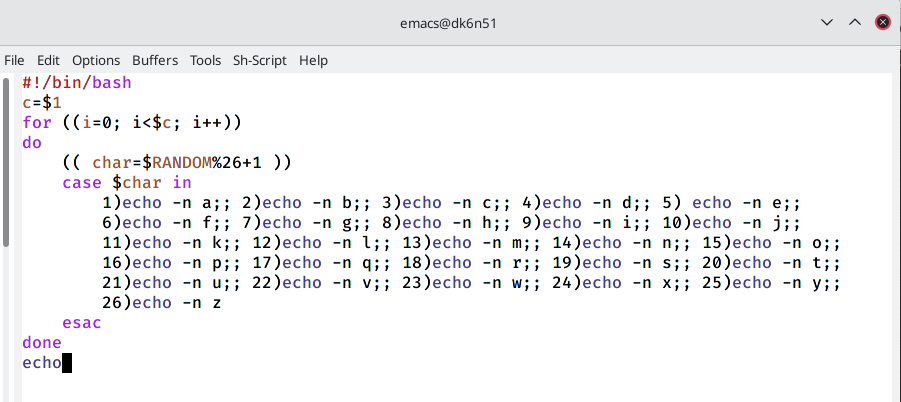


Рис. 14: Текст скрипта3

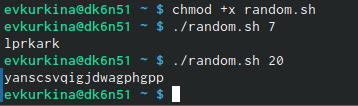


Рис. 15: Права доступа резльтат проверки исполнения файла

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”]

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 данного выражения $((10/3))будет 3, потому что это целочисленное деление без остатка.

5). 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

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 Выводы

Во время выполнения данной лабораторной работы я изучила основы программирования в оболочке OC UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

# Список литературы