Лабораторная работа № 13

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Маметкадыров Ынтымак

# Цель работы

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

# Выполнение лабораторной работы

1. Написали командный файл, реализующий упрощённый механизм семафоров. Командный файл в течение некоторого времени t1 дожидается освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использует его в течение некоторого времени t2<>t1, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (рис. 1). Затем проверили работу командного файла (рис. 2).

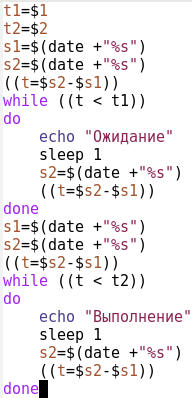


Рис. 1. Листинг командного файла semaf.sh

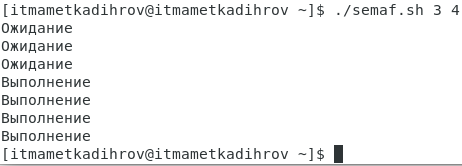
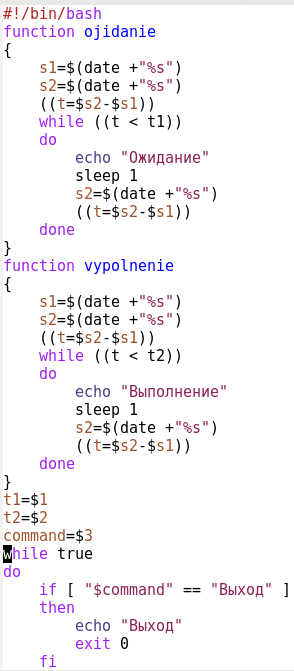


Рис. 2. Работа командного файла semaf.sh

1. Доработали командный файл так, чтобы ее можно было выполнять в нескольких терминалах (рис. 3). Запустили командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой.



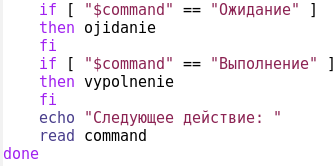


Рис. 3. Командный файл semaf.sh

1. Реализовали команду man с помощью командного файла (рис. 4). Командный файл получает в виде аргумента командной строки название команды и в виде результата выдает справку об этой команде, например, если ввести команду “./man.sh ls” (рис. 6), то выведится справка о команде ls, (рис. 5) или сообщение об отсутствии справки, если соответствующего файла нет в каталоге man1 (рис. 6).

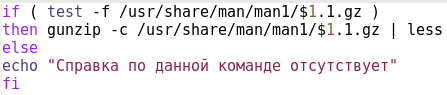


Рис. 4. Командный файл man.sh

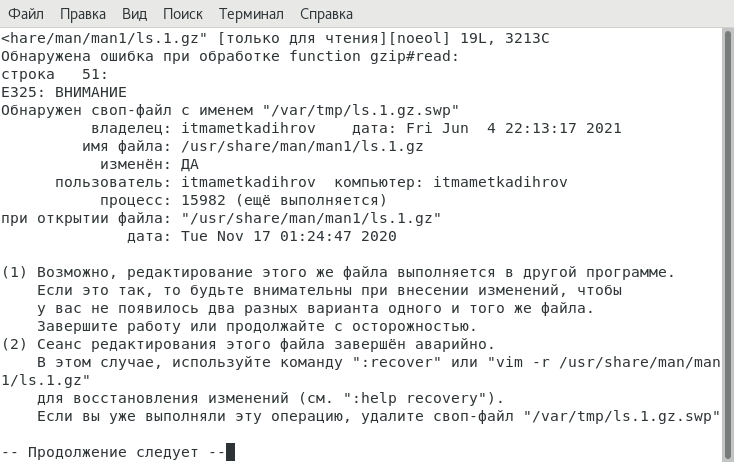


Рис. 5. Результат команды ./man.sh ls

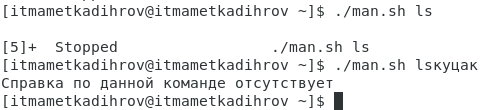


Рис. 6. Сообщение об отсутствии справки

1. Используя встроенную переменную $RANDOM, написали командный файл (рис. 7), генерирующий случайную последовательность букв латинского алфавита (рис. 8).

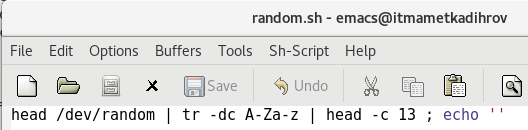


Рис. 7. Командный файл random.sh



Рис. 8. Результат работы командного файла random.sh

# Вывод

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

# Ответы на 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 основе неполных данных
   * В zsh поддерживается замена части пути
   * В zsh есть возможность отображать разделенный экран, такой же как разделенный экран vim
3. for ((a=1; a <= LIMIT; a++)) синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать $ перед переменными ().
4. 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