Отчёт по лабораторной работе №12

Операцонные системы

Сячинова Ксения Ивановна

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы

1. Напишем командный файл, реализующий упрощённый механизм семафоров. Командный файл должен будет 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 этого создадим файл: sem.sh и напишем скрипт. (рис. 1), (рис. 2)

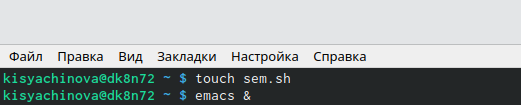


Рис. 1: Создание файла



Рис. 2: Скрипт 1

Далее проверим работу написанного скрипта с помощью команды “./sem.sh 4 7”, но перед этим добавим право на выполнение командой “chmod +x sem.sh”. Скрипт работает верно. (рис. 3)

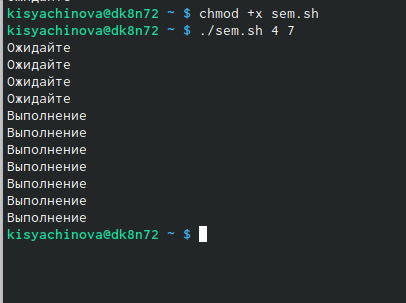


Рис. 3: Проверка скрипта 1

Далее изменим скрипт так, чтобы его можно было выполнять в нескольких термиалах.(рис. 4)

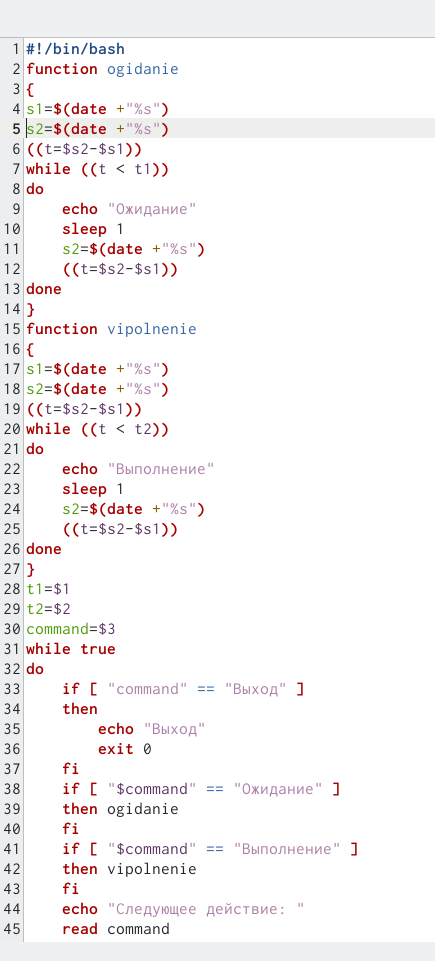


Рис. 4: Изменённый скрипт

Проверим работу с помощью комманды “./sem.sh 2 3 Ожидание>/dev/pts/1 &”. Проверить работа данного скрипта не удалось, так как было отказано в доступе.(рис. 5)

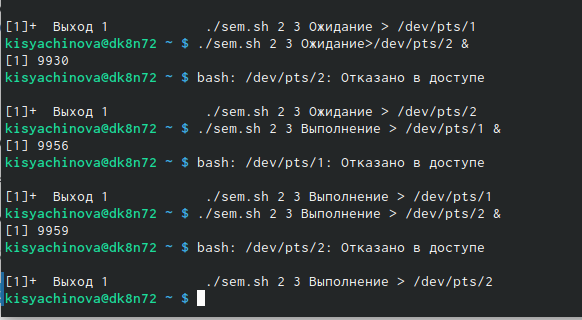


Рис. 5: Проверка изменённого скрипта

1. Реализуем команду (man с) помощью командного файла. Изучим 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.(рис. 6)

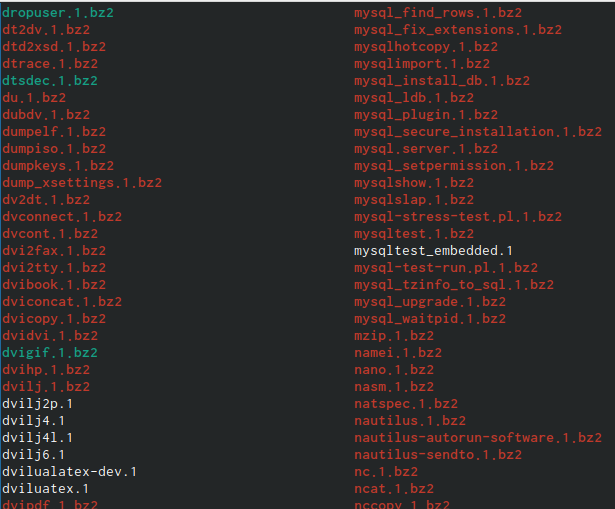


Рис. 6: Реальзация команды man

Далее я создала файл “man.sh”, в котором буду писать следйющий скрипт. (рис. 7)

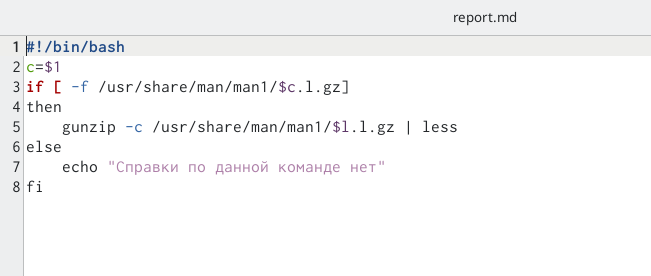


Рис. 7: Скрипт 2

Проверим работу скрипта, дадим ему право на выполнение “chmod +x man.sh” и проверим с помощью команды “./man.sh ls”, “./man.sh mkdir”(рис. 8)

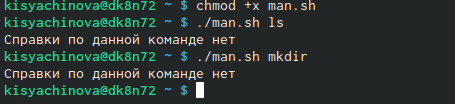


Рис. 8: Проверка скрипта 2

1. Используем встроенную переменную $RANDOM, напишем командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтём, что $RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767. Создадим файл для напсиания тертьего скрипта (random.sh).(рис. 9)

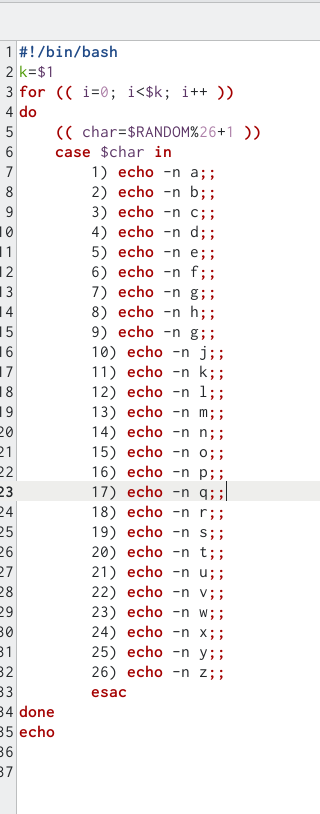


Рис. 9: Скрипт 3

Затем даём право на выполнение и с помощью команды “./random 5” и “./random 12” проверяем выполнение работы.(рис. 10)

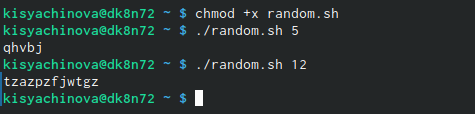


Рис. 10: Проверка 3 скрипта

# 3 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”]
2. Чтобы объединить несколько строк в одну, можно воспользоваться несколь- кими способами:

* Первый: VAR1=”Hello, “VAR2=” World” VAR3=“𝑉 𝐴𝑅1VAR2” 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 выравнива- ния ширины путем заполнения начальными нулями. FIRST и INCREMENT являются необязательными.

1. Результатом данного выражения $((10/3))будет 3, потому что это целочис- ленное деление без остатка.
2. Отличия командной оболочки zshот bash:

* В zsh более быстрое автодополнение для cdс помощью Тab
* В zsh существует калькулятор zcalc, способный выполнять вычисления внут- ри 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 разде- ленный 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 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

В ходе выполнение данной лабораторной работы я изучила основы программирования в оболочке OC UNIX и научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.