Отчет по лабораторной работе №13

Дисциплина: Операционные системы

Пермякова Елизавета Евгеньевна

Содержание

# Цель работы

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

# Задачи

1. Познакомиться с логическими управляющими конструкций и циклов.
2. В ходе работы написать 3 командных файла.
3. Выполнить отчет.

# Выполнение лабораторной работы

1. Написала 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 создала файл prog1.sh и написала соответствующий скрипт. (рис. 1)

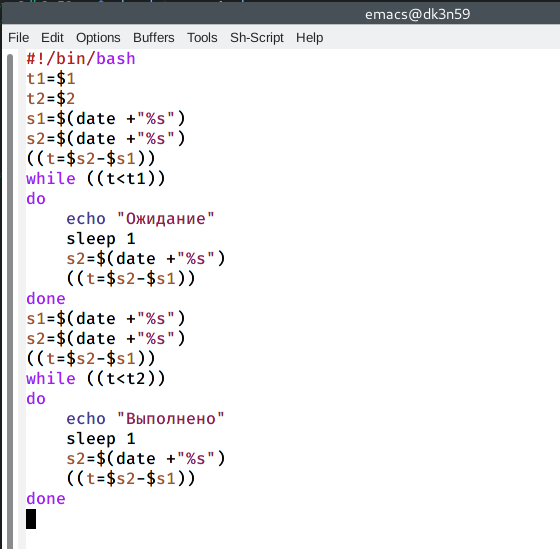


Figure 1: Первый скрипт

Далее я проверила работу написанного скрипта (команда«./prog1.sh 2 6»), предварительно добавив право на исполнение файла (команда «chmod +x prog1.sh»). Скрипт работает корректно. (рис. 2):

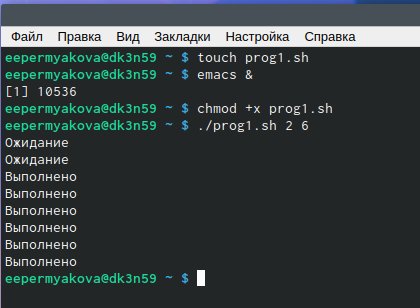


Figure 2: Проверка работы скрипта

После этого я изменила скрипт так, чтобы его можно было выполнять в нескольких терминалах и проверила его работу (например, команда «./prog1.sh 2 5 Выполнение > /dev/pts/2 &» или команда «./prog1.sh 2 5 Выполнение > /dev/tty2» ). Но ни одна команда не работала, выводя сообщение “Отказано в доступе”. При этом скрипт работает корректно (команда «./prog1.sh 3 6»). (рис. 3) (рис. 4)



Figure 3: Измененный скрипт

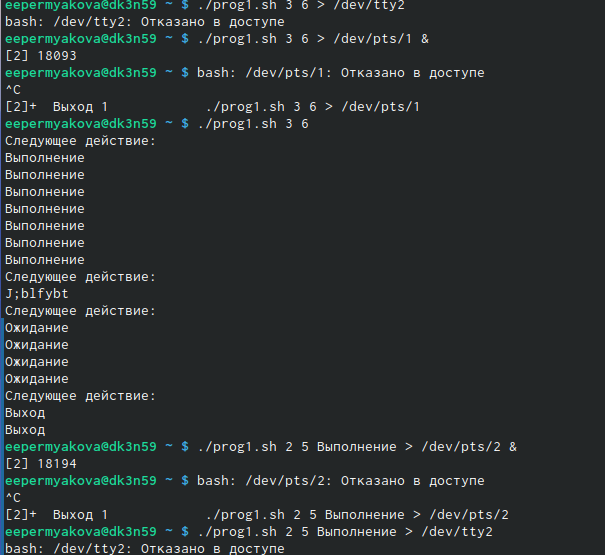


Figure 4: Проверка работы скрипта

1. Реализовалакоманду manс помощью командного файла. Изучила 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. 5) (рис. 6)

Figure 5: Каталог /usr/share/man/man1

Figure 5: Каталог /usr/share/man/man1

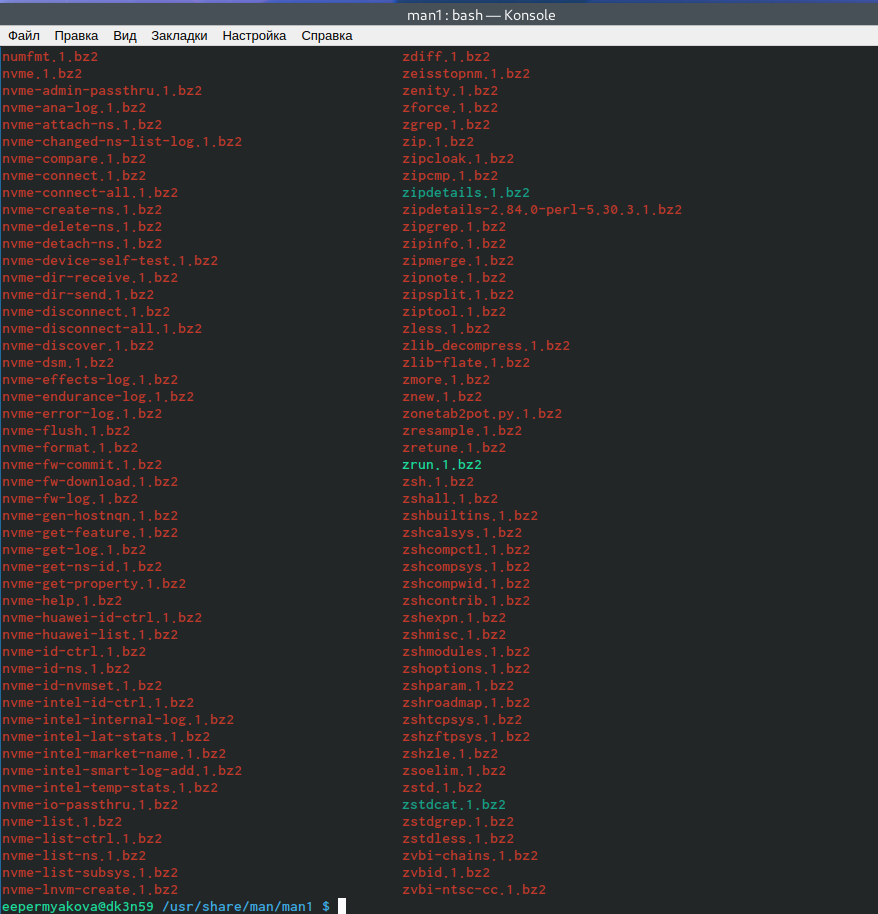


Figure 6: Cодержимое каталога

Для данной задачи я создала файл prog2.sh и написала соответствующий скрипт. (рис. 7)

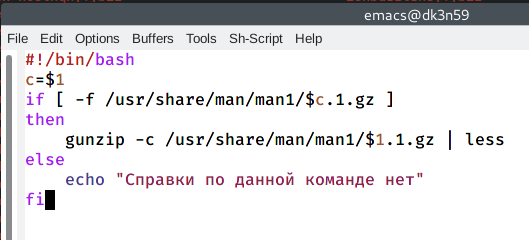


Figure 7: Второй скрипт

Далее я проверила работу написанного скрипта(команды «./prog2.sh ls», «./prog2.sh mkdir» и т. д.), предварительно добавив право на исполнение файла (команда «chmod +x prog2.sh»). Cкрипт сработал и вывел, что по данным командам справок нет. (рис. 8)

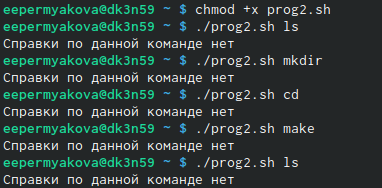


Figure 8: Проверка работы скрипта

1. Используя встроенную переменную $RANDOM, написала командный файл, генерирующий случайную последовательность букв латинского алфавита.

Для данной задачи я создала файл prog3.sh и написала соответствующий скрипт. (рис. 9)

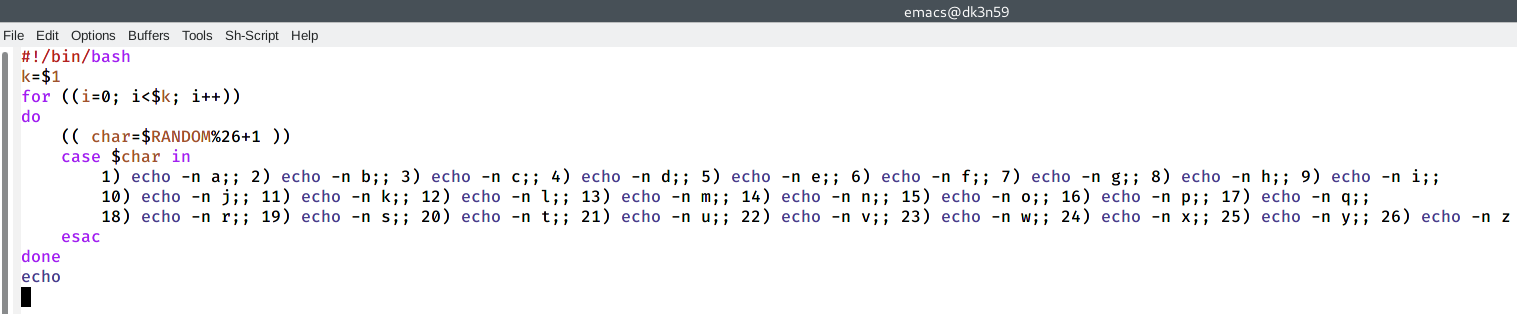


Figure 9: Третий скрипт

Далее я проверила работу написанного скрипта (команды «./prog3.sh 4», «./prog3.sh 23» и «./prog3.sh 26»), предварительно добавив право на исполнение файла (команда «chmod +x prog3.sh»). Скрипт работает корректно.(рис. 10)

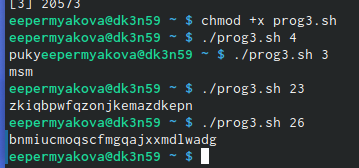


Figure 10: Проверка работы скрипта

# 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 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

1. 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.

# Выводы

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

# Библиография

1. https://esystem.rudn.ru/pluginfile.php/1142096/mod\_resource/content/2/010-lab\_shell\_prog\_3.pdf
2. Кулябов Д.С. Операционные системы: лабораторные работы: учебное пособие / Д.С. Кулябов, М.Н. Геворкян, А.В. Королькова, А.В. Демидова. — М. : Изд-во РУДН, 2016. — 117 с. — ISBN 978-5-209-07626-1 : 139.13; То же [Электронный ресурс]. — URL: http://lib.rudn.ru/MegaPro2/Download/MObject/6118.
3. Робачевский А.М. Операционная система UNIХ [текст] : Учебное пособие / А.М. Робачевский, С.А. Немнюгин, О.Л. Стесик. — 2-е изд., перераб. и доп. — СПб. : БХВ-Петербург, 2005, 2010. — 656 с. : ил. — ISBN 5-94157-538-6 : 164.56. (ЕТ 60)
4. Таненбаум Эндрю. Современные операционные системы [Текст] / Э. Таненбаум. — 2-е изд. — СПб. : Питер, 2006. — 1038 с. : ил. — (Классика Computer Science). — ISBN 5-318-00299-4 : 446.05. (ЕТ 50)