Отчёт по лабораторной работе № 12

*дисциплина:*Операционные системы

Андрианова Марина Георгиевна

Содержание

# 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.Создала файл s1.sh(рис.1) и написала соответствующий скрипт(рис.2): 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).

Создание файла s1.sh

Рис.1: Создание файла s1.sh

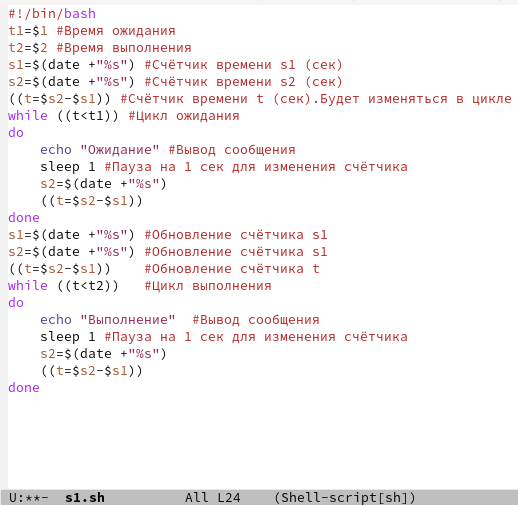


Рис.2: Пишем командный файл s1.sh

Проверяем работу написанного скрипта (команда “./s1.sh 2 5”), предварительно добавив право на исполнение файла(команда “chmod +x s1.sh”). Скрипт работает корректно(рис.3).

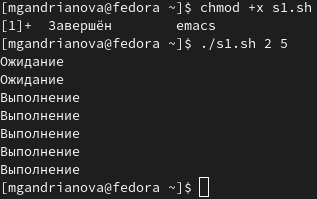


Рис.3: Проверка скрипта

После этого изменяем скрипт так, чтобы его можно было выполнять в нескольких терминалах(рис.4). Проверила его работу(команда “./s1.sh 3 5 Ожидание > /dev/pts/2 &” и команда “./s1.sh 3 5 Ожидание > /dev/tty2”). При этом ни одна из команд не сработала, выводя сообщение “Отказано в доступе”. При этом скрипт работает корректно(рис.5).

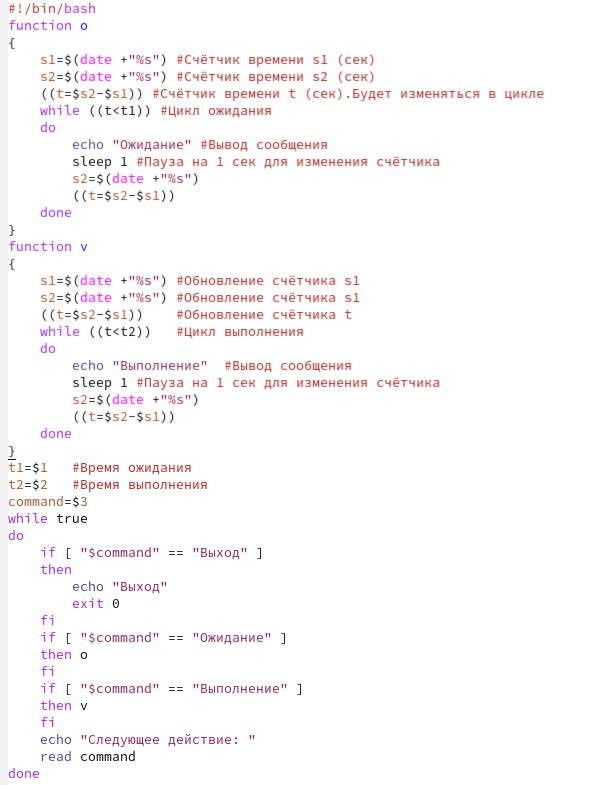


Рис.4: Изменение командного файла s1.sh

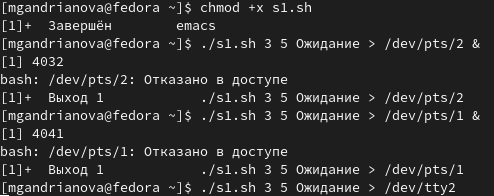


Рис.5: Проверка скрипта

1. Реализовала команду man с помощью командного файла. Изучила содержимое каталога /usr/share/man/man1:сначала перешла в него командой “cd /usr/share/man/man1”,а затем вывела его содержимое(команда “ls”)(рис.6). 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.

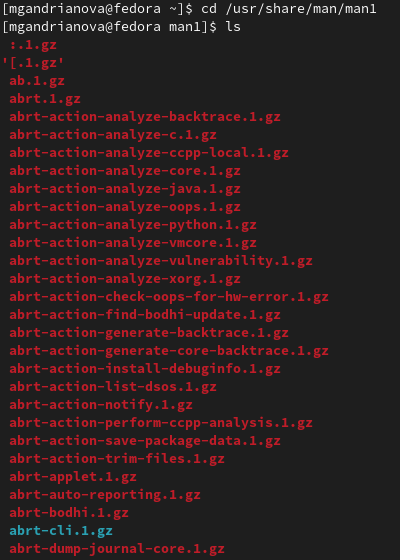


Рис.6: Переход в каталог и вывод его содержимого

Для данной задачи я создала файл s2.sh(рис.7) и написала соответствующий скрипт(рис.8).

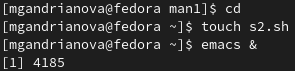


Рис.7: Создание файла s2.sh

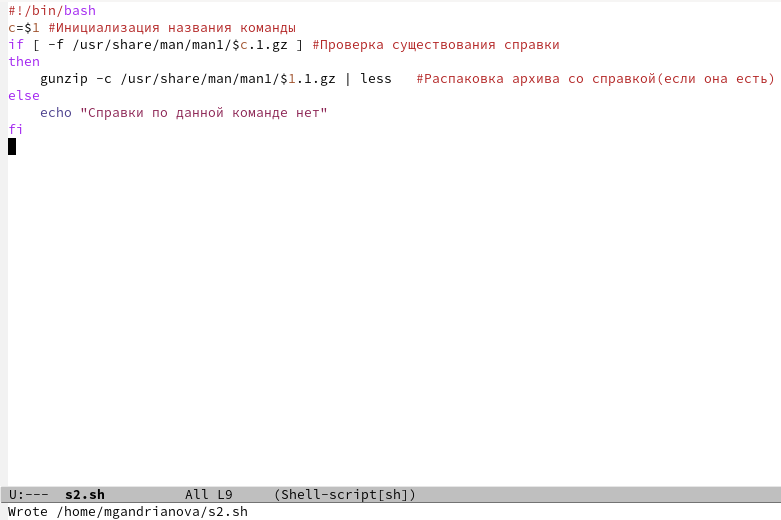


Рис.8: Пишем командный файл s2.sh

Проверяем работу написанного скрипта (команды “./s2.sh touch” и “./s2.sh rm”), предварительно добавив право на исполнение файла(команда “chmod +x s2.sh”)(рис.9). Скрипт работает корректно(рис.10,рис.11).

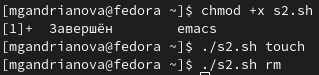


Рис.9: Предоставление права на исполнение и проверка скрипта

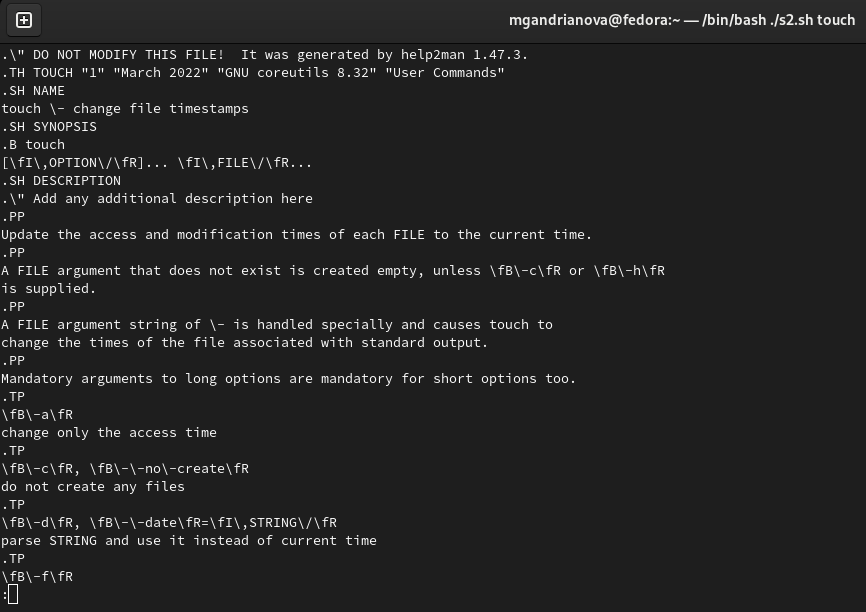


Рис.10: Результат после введения команды “./s2.sh touch”

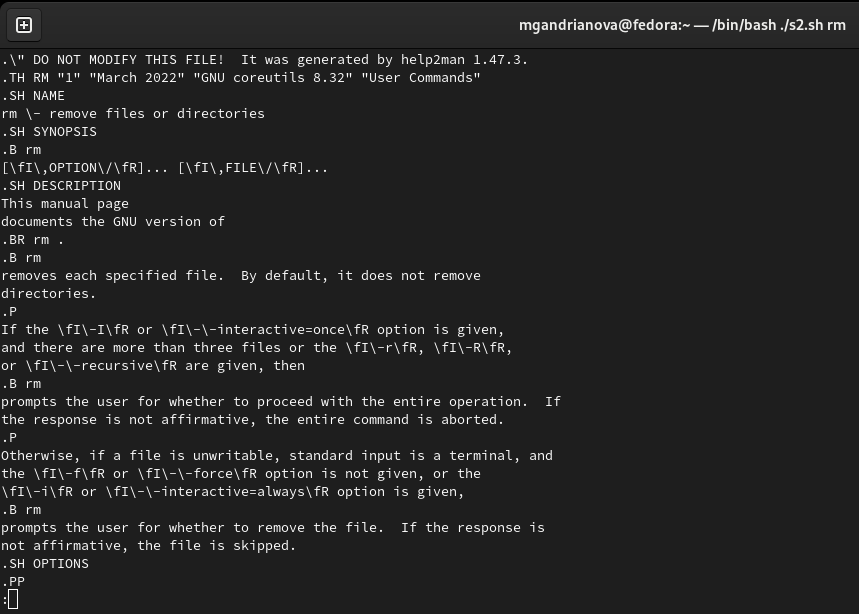


Рис.11: Результат после введения команды “./s2.sh rm”

1. Создала файл s3.sh(рис.12) и написала соответствующий скрипт(рис.13). Используя встроенную переменную $RANDOM, написала командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтём, что $RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767. Добавила право на исполнение файла(команда “chmod +x s3.sh”)(рис.14). Скрипт работает корректно(рис.14).

Создание файла s3.sh

Рис.12: Создание файла s3.sh

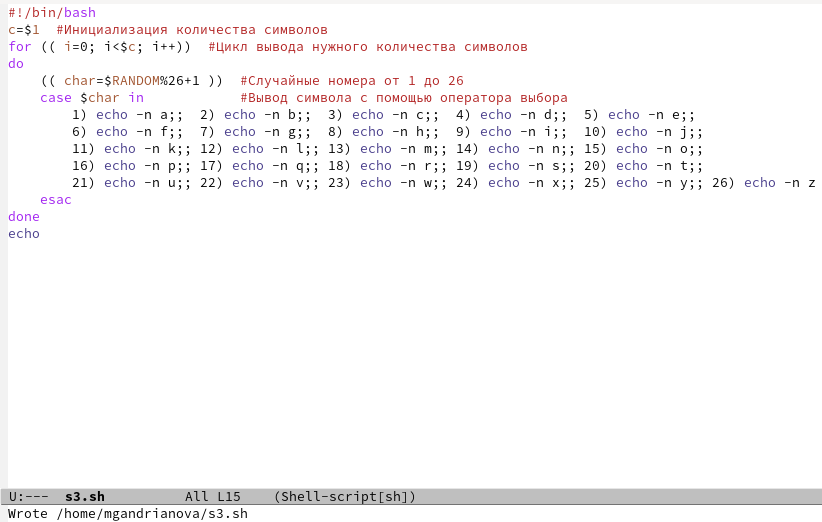


Рис.13: Пишем командный файл s3.sh

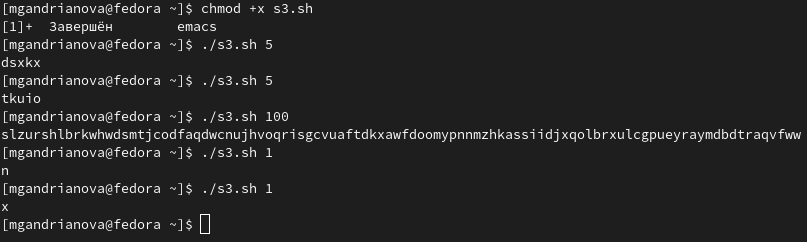


Рис.14: Проверка скрипта

# Выводы

Я изучила основы программирования в оболочке ОС UNIX и научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

# 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=“$VAR1$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 zsh от 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 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 языком 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.