Отчет по лабораторной работе №12

Дисциплина: Операционные системы

Шишук Владислав Олегович

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. Создаем файл 1s.sh и пишем соответствующий скрипт. (рис. -@fig:001) Пишем 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).(рис. -@fig:002)

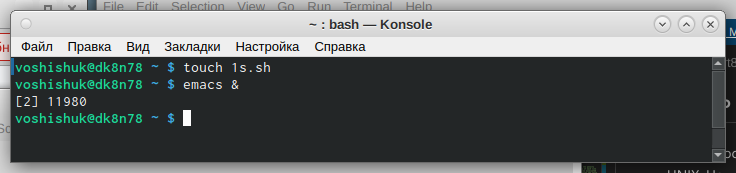


рис.1

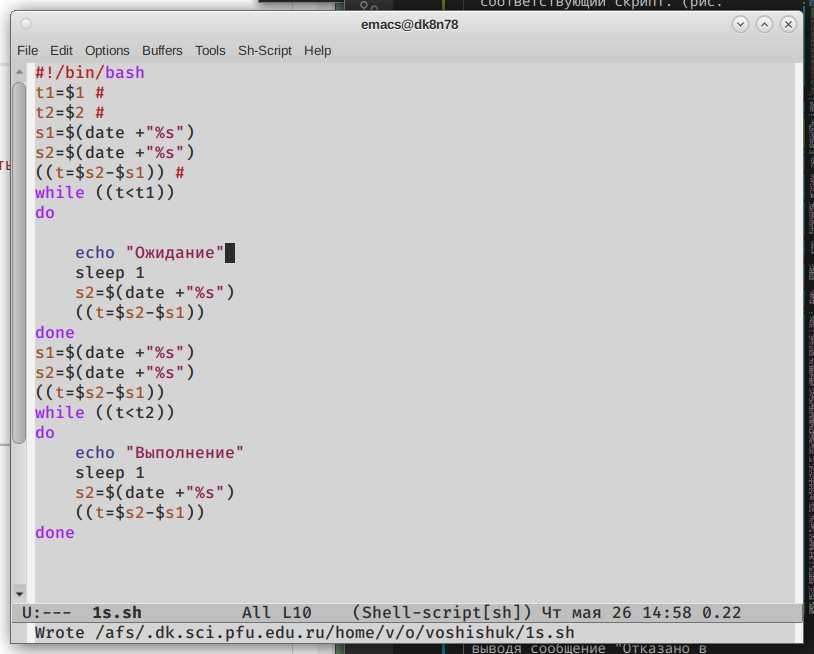


рис.2

Проверяем работу написанного скрипта (команда «./1s.sh 3 5»), предварительно добавив право на исполнение файла (команда «chmod +x 1s.sh»). Скрипт работает корректно.(рис. -@fig:003)

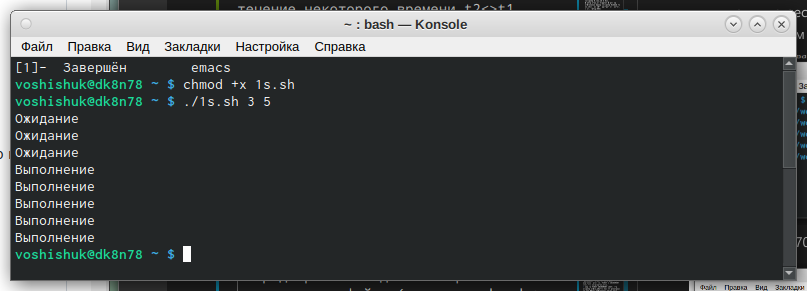


рис.3

После этого изменяем скрипт так, чтобы его можно было выполнять в нескольких терминалах и проверим его работу (команда «./1s.sh 2 5 Ожидание > /dev/pts/2 &» и команда «./1s.sh 2 5 Ожидание > /dev/tty2 »). При этом ни одна из команд не сработала, выводя сообщение “Отказано в доступе”. При этом скрипт работает корректно. (рис. -@fig:004) (рис. -@fig:005) 

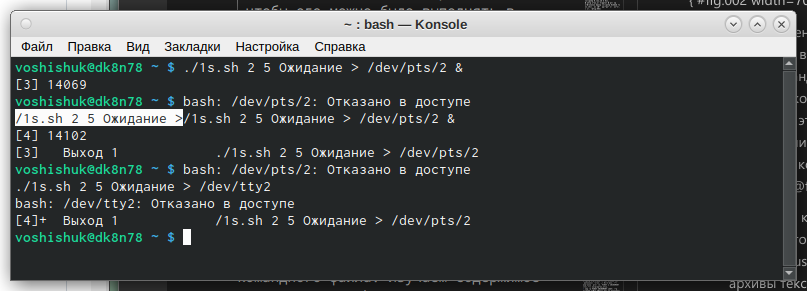


рис.5

1. Реализуем команду man с помощью командного файла. Изучаем 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. -@fig:009)

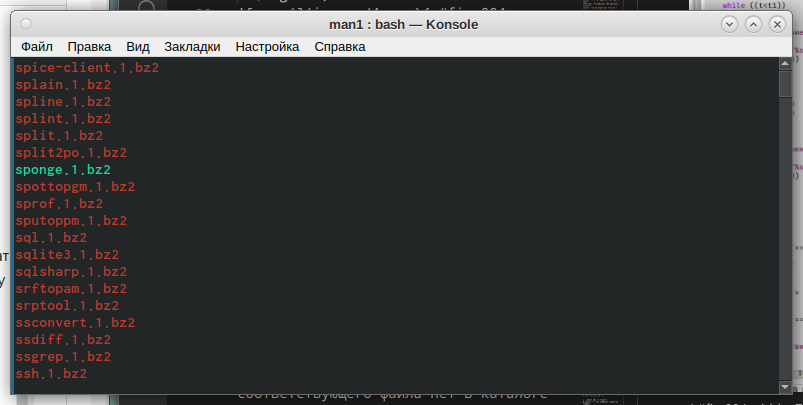


рис.6

Создаем файл 2s.sh и пишем соответствующие скрипт. (рис. -@fig:007)

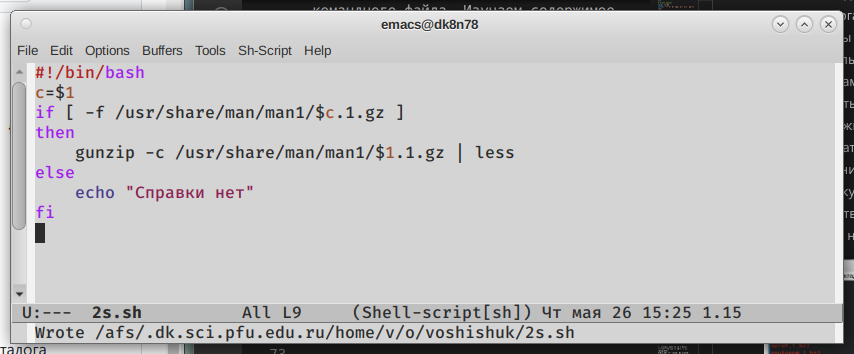


рис.6.1

Проверяем работу написанного скрипта (команды «./2s.sh mkdir» и «./2s.sh rm»), предварительно добавив право на исполнение файла (команда «chmod +x 2s.sh»). Скрипт работает корректно. (рис. -@fig:008)

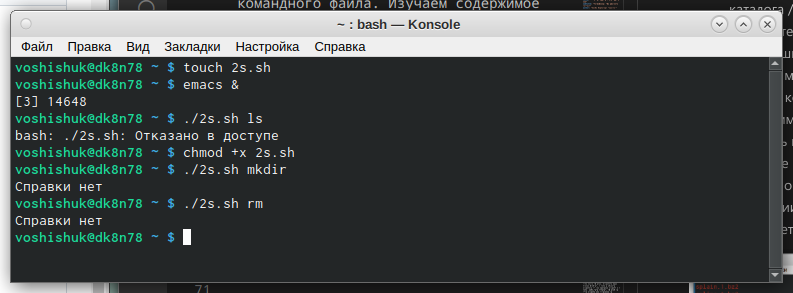


рис.7

1. Создаем файл 3s.sh и пишем соответствующие скрипты.Используя встроенную переменную $RANDOM, пишем командный файл, генерирующий случайную последовательность букв латинского алфавита. (рис. -@fig:009)

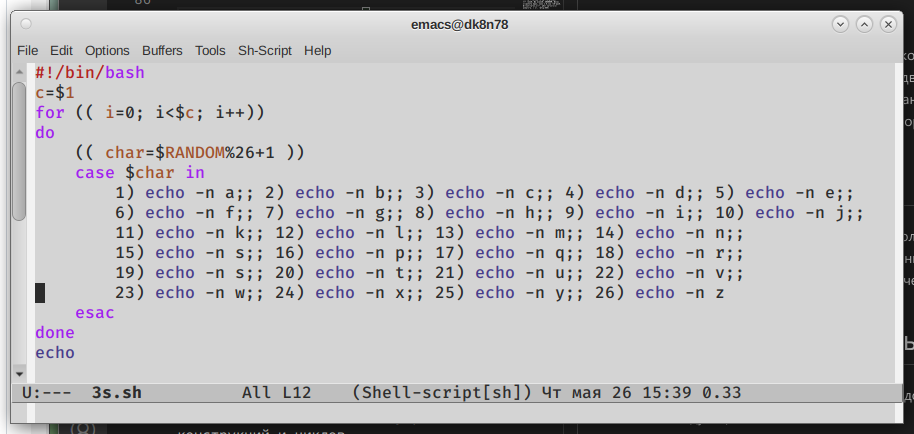


рис.8

Проверяем работу написанного скрипта (команды «./3s.sh 45», «./3s.sh 1000», «./3s.sh 1»), предварительно добавив право на исполнение файла (команда «chmod +x random.sh»). Скрипт работает корректно. (рис. -@fig:010)

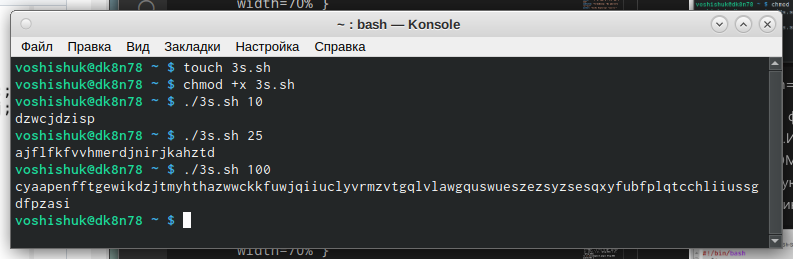


рис.9

# Выводы

Я изучил основы программирования в оболочке ОС UNIX, а также научился 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” ]

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 основенеполных данных
* В zsh поддерживается замена части пути
* 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 ().
2. 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