Лабораторная работа №12

Дисциплина: Операционные системы

Королев Федор Константинович

Содержание

[Цель работы 1](#_Toc104503931)

[Ход работы 1](#_Toc104503932)

[Контрольные вопросы 5](#_Toc104503933)

[Вывод 6](#_Toc104503934)

# Цель работы

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

# Ход работы

1. Написал командный файл, реализующий упрощенный 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 некторого времени t2< >t1, также выдавая информацию о том, что ресурс используется соответствующим командным файлом. Для данной задачи создадим файд sem.sh(Рис. 1) и напишем соответствующий скрипт(Рис. 2):

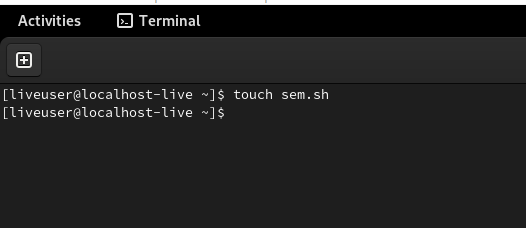


Рис. 1 создание файла

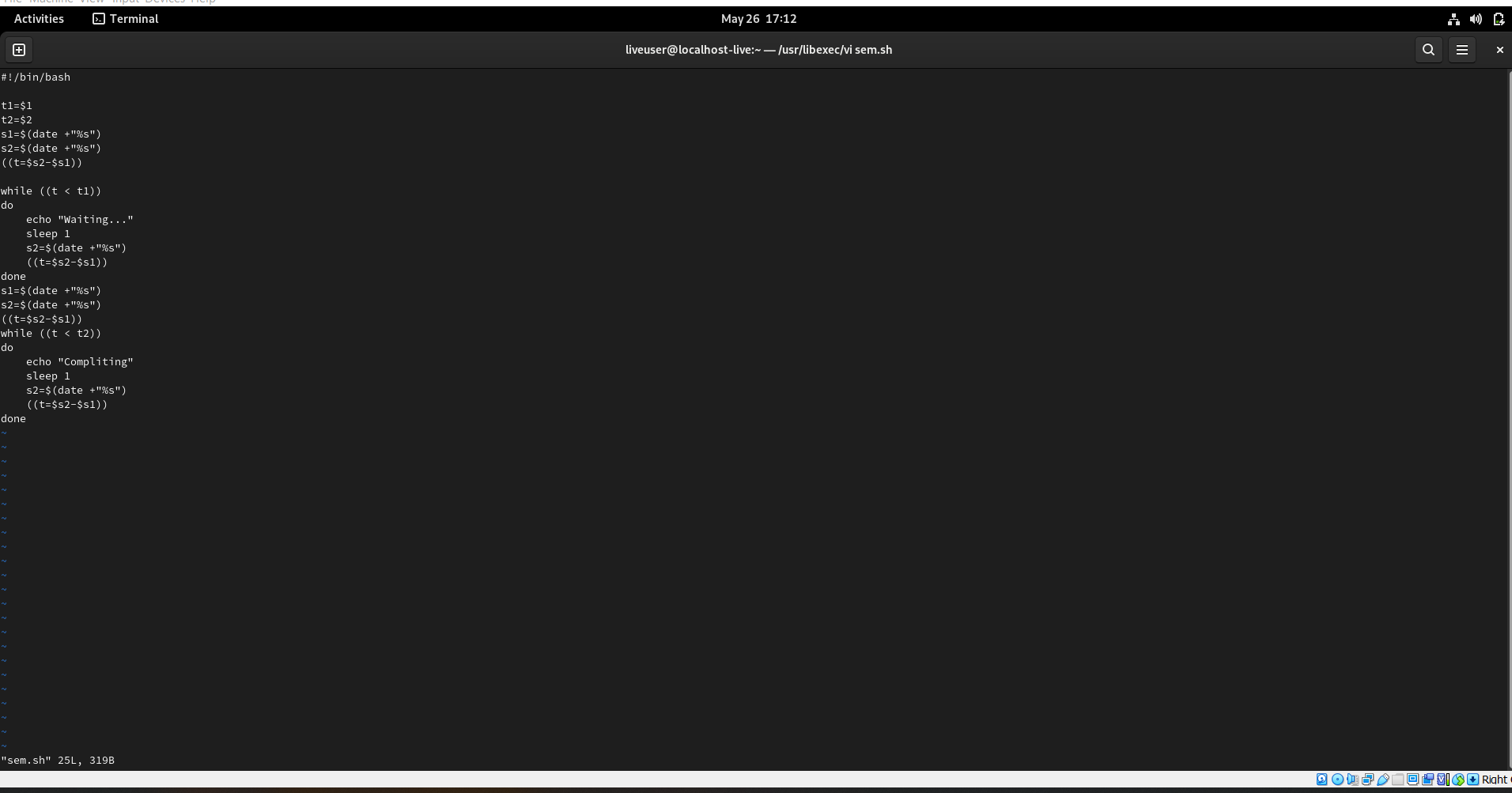


Рис. 2 написание скрипта

Дадим право на исполнение и проверим работу(Рис. 3):

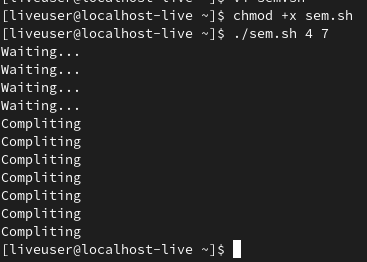


Рис. 3 проверка

1. Реализуем команду man с помощью командного файла. Изучим содержимое каталога /usr/share/man/man1 (Рис. 7, 8). 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.

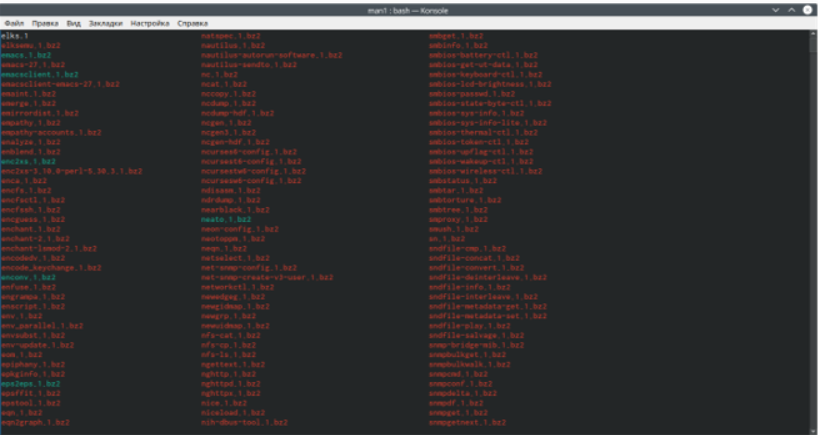


Рис. 4 реализация команды man

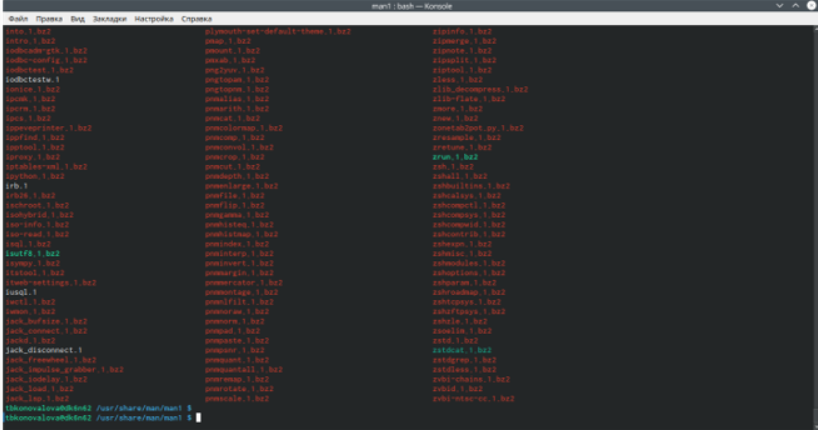


Рис. 5 реализация команды man

Для реализации создадим файл man.sh(Рис. 6, 7):

Рис. 6 создание файла

Рис. 6 создание файла

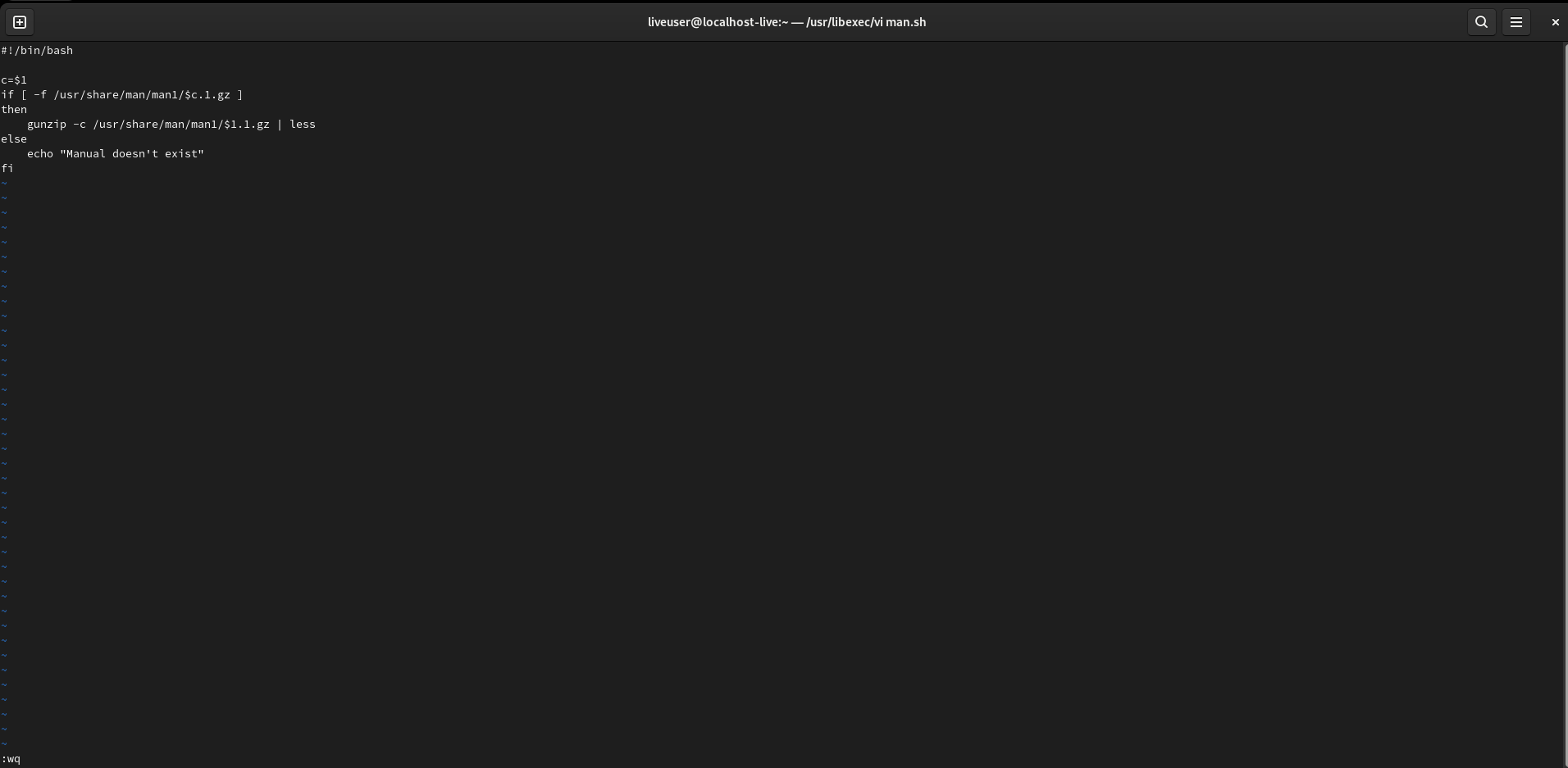


Рис. 7 реализация скрипта

Дадим права доступа и проверим(Рис. 8):

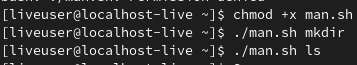


Рис. 8 проверка

1. Используя встроенную переменную $RANDOM, написал командный файл, генерирующий случайную последовательность букв латинского алфавита. Для данной задачи я создал файл: random.sh(Рис. 9) и написал соответствующий скрипт(Рис. 10):

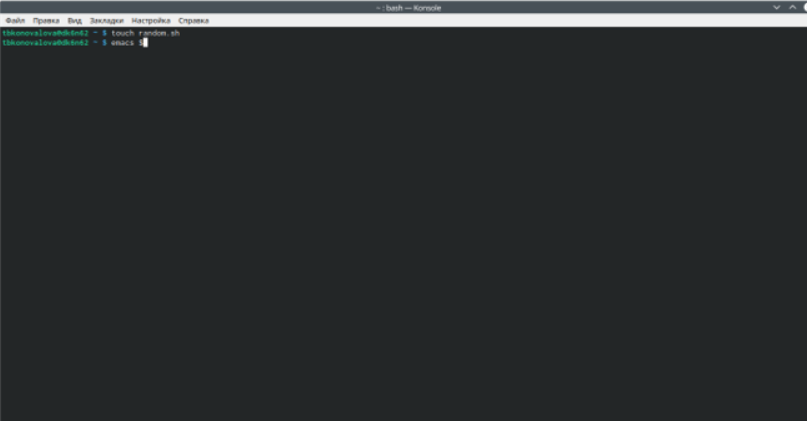


Рис. 9 создание файла



Рис. 10 скрипт

Проверим работу скрипта, предварительно дав права доступа(Рис. 11):



Рис. 11 проверка

# 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 [“$1”!= “exit”] 2). Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами: Первый: VAR1="Hello, “VAR2=” World" VAR3=“VAR1VAR2” echo “$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 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 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 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.

# Вывод

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