Отчёт по лабораторной работе №12

Программирование в командномпроцессоре ОС UNIX. Расширенное программирование

Федюшина Ярослава Андреевна

Содержание

# 1 Цель работы

Изучить основы программирования в оболочке ОС UNIX. Научиться писать болеесложные командные файлы с использованиемлогических управляющих конструкцийи циклов

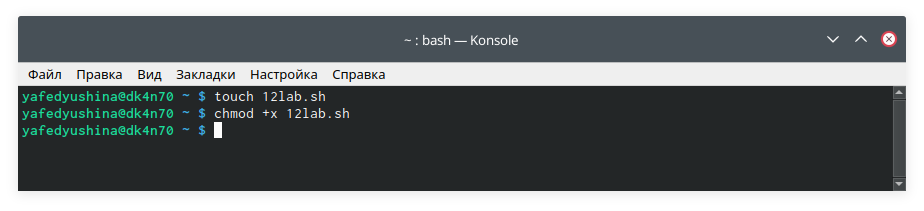
# 2 Задание

1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен втечение 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).Запустить командный файл в одном виртуальном терминале в фоновом режиме,перенаправив его вывод в другой (> /dev/tty#,где # —номер терминала куда перенаправляется вывод),в котором также запущен этот файл,но не фоновом,а в привилегированном режиме.Доработать программу так,чтобы имелась возможность взаимодействия трёх и более процессов.
2. Реализовать команду man с помощью командного файла.Изучите 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.
3. Используя встроенную переменную $RANDOM,напишите командный файл,генерирующий случайную последовательность букв латинского алфавита.Учтите,что $RANDOM выдаёт псевдо случайные числа в диапазоне от 0 до 32767.

# 3 Выполнение лабораторной работы

Создаю файл для выполнения работы и написания кода

## 3.1

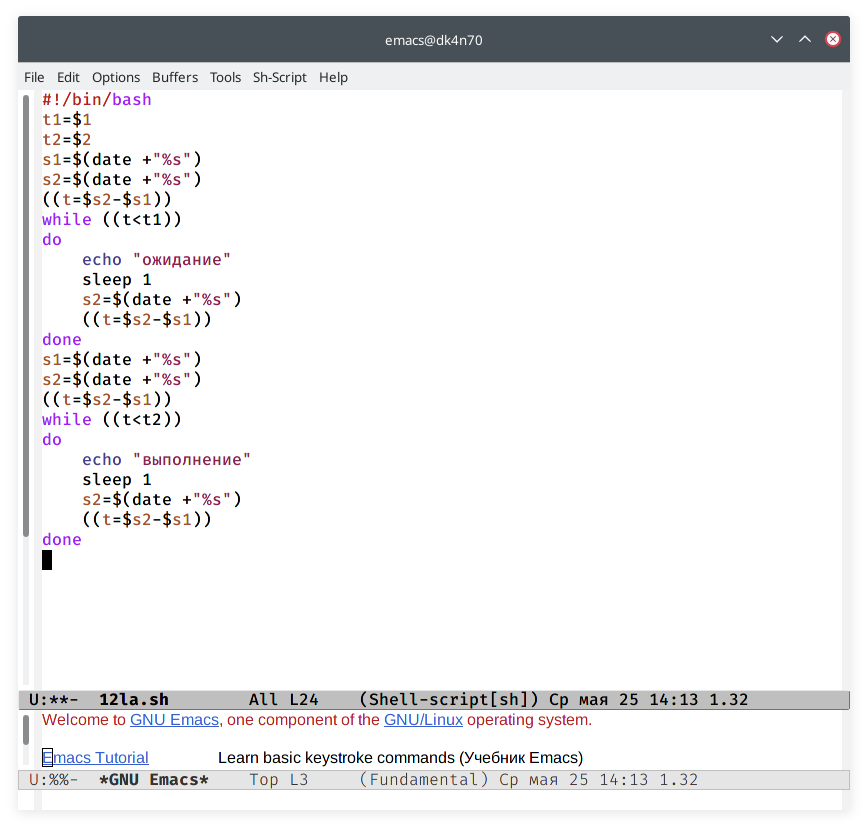


создание файла

## 3.2

написание кода по заданию в emacs

## 3.3

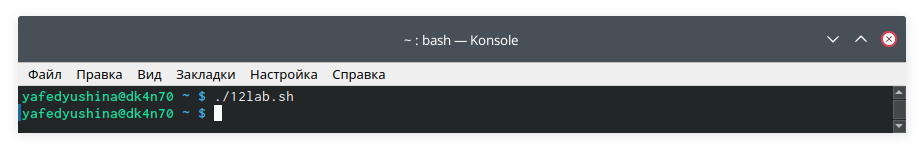


код

## 3.4

проверяю написанную мной программу

## 3.5

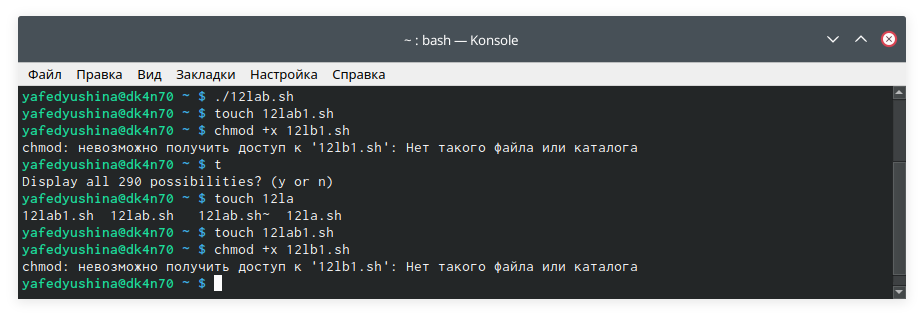


проверка программы

## 3.6

создаю новый файл для реализации команды man

## 3.7

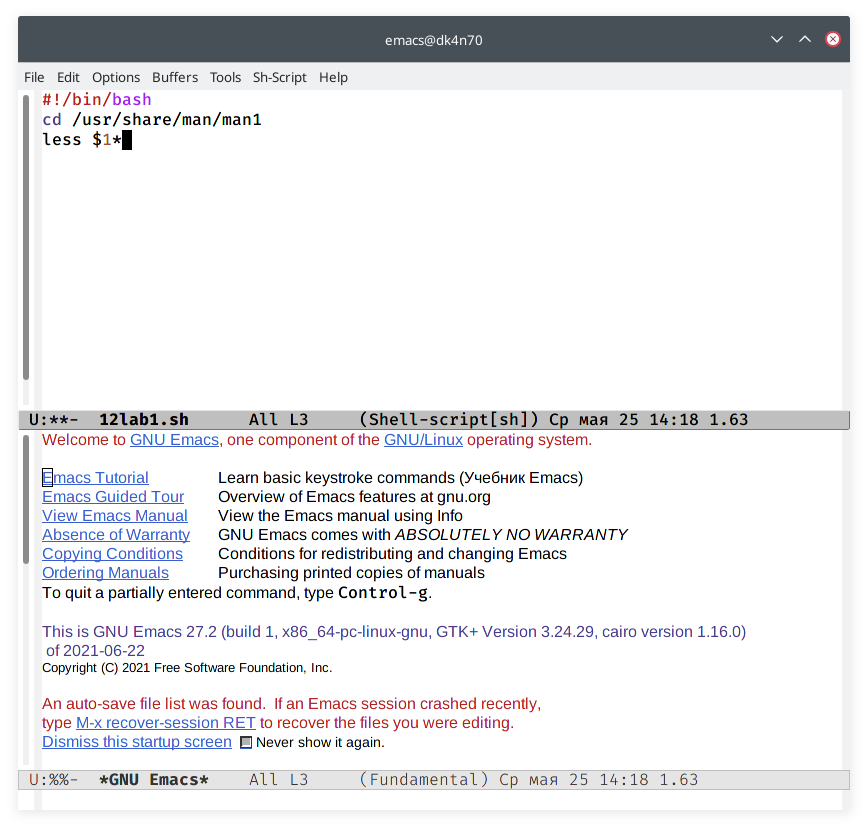


создание нового файла

## 3.8

пишу сам код для реализации задумки

## 3.9

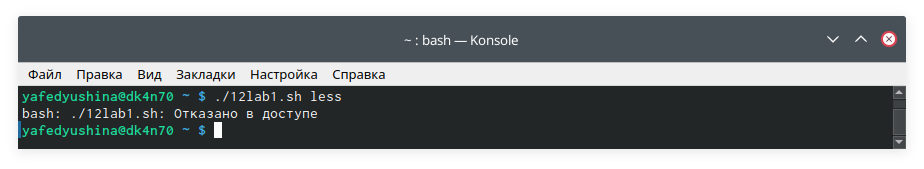


код для man

## 3.10

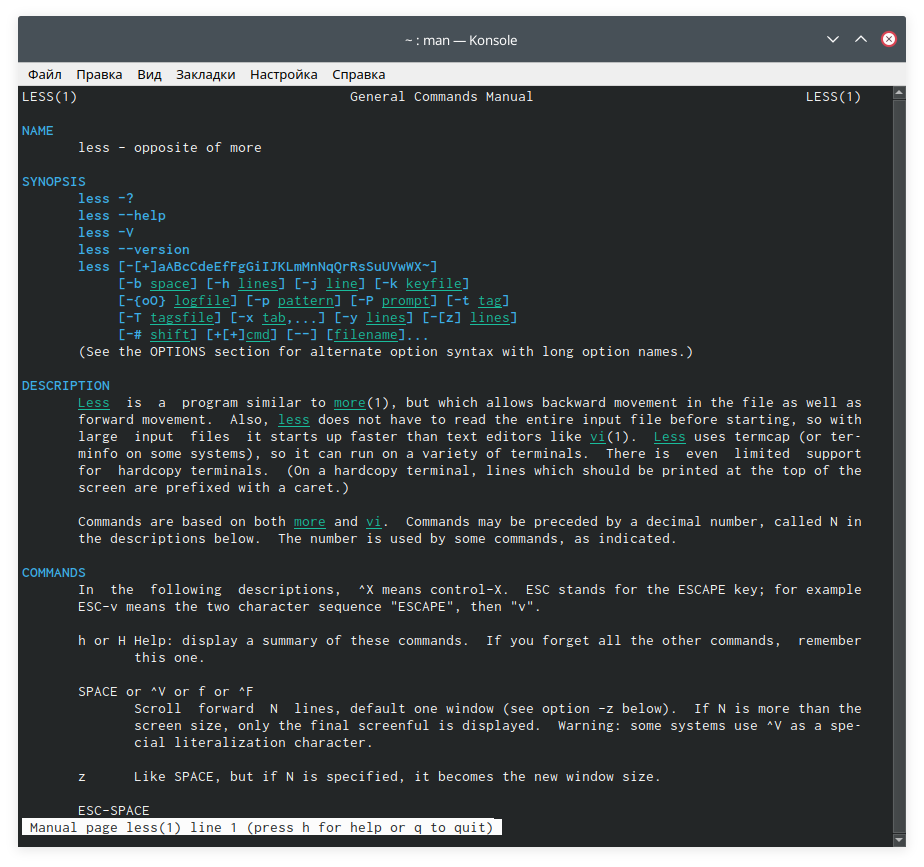
проверяю новую программу на работу

## 3.11



проверка кода

## 3.12

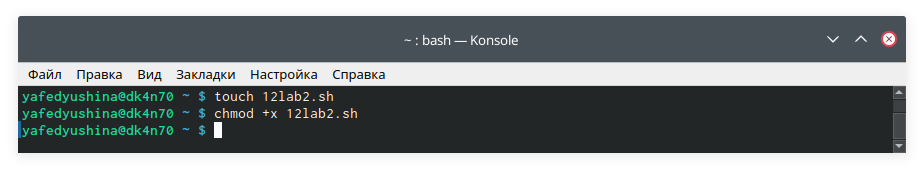


man less

## 3.13

создаю новый текстовый файл и делаю его исполняемым

## 3.14

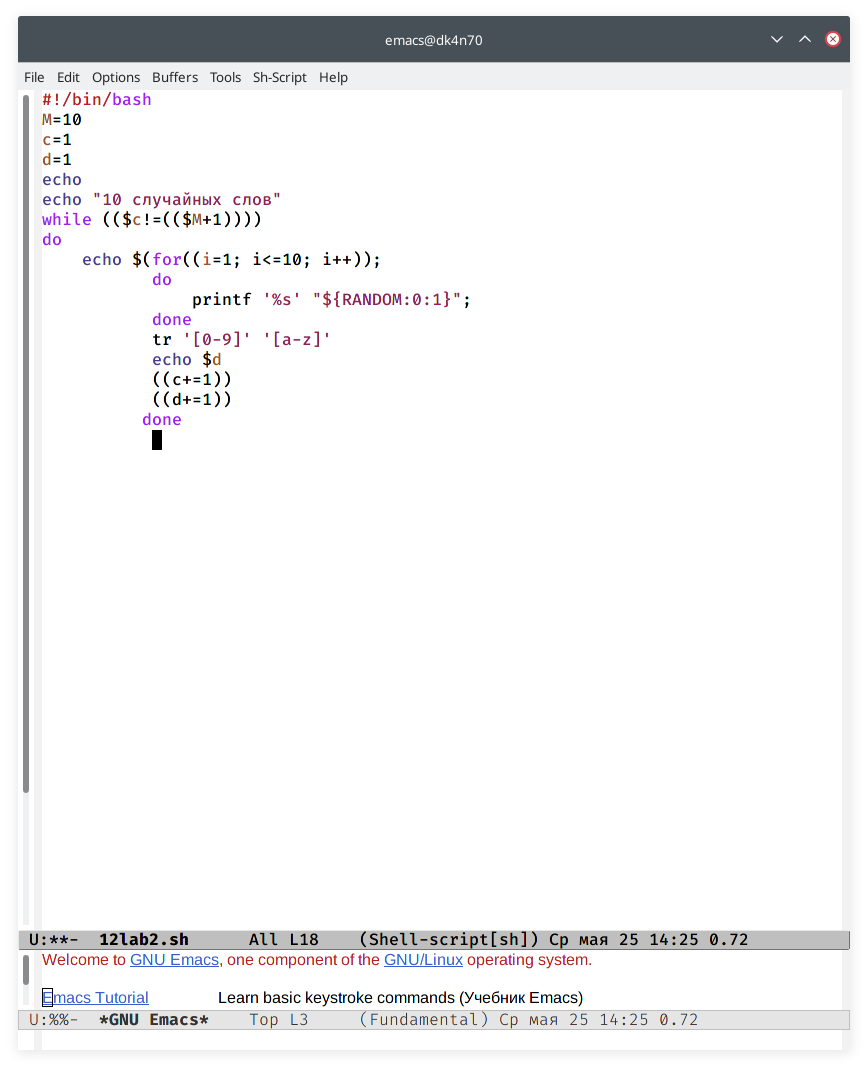


создание нового файла

## 3.15

пишу код, генерирующий случайную последовательность букв латинского алфавита

## 3.16

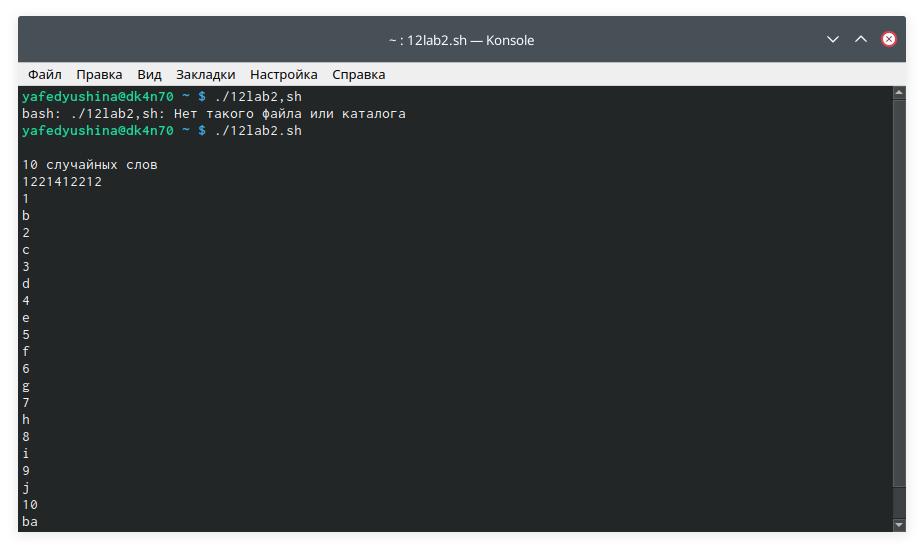


код random

## 3.17

проверяю программу на работу

## 3.18



проверка кода с random

## 3.19

# 4 Выводы

В ходе выполнения лабораторной работы №12 я изучила основы программирования в оболочке ОС Linux, а так же научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов

# 5 Клнтрольные 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