РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Лабораторная работа No 12. Программирование в командном процессоре ОС UNIX. Расширенное программирование

Абдуллина Ляйсан Раисовна НПИбд-01-21

Содержание

# 1 Цель работы

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

# 2 Теоретическое введение

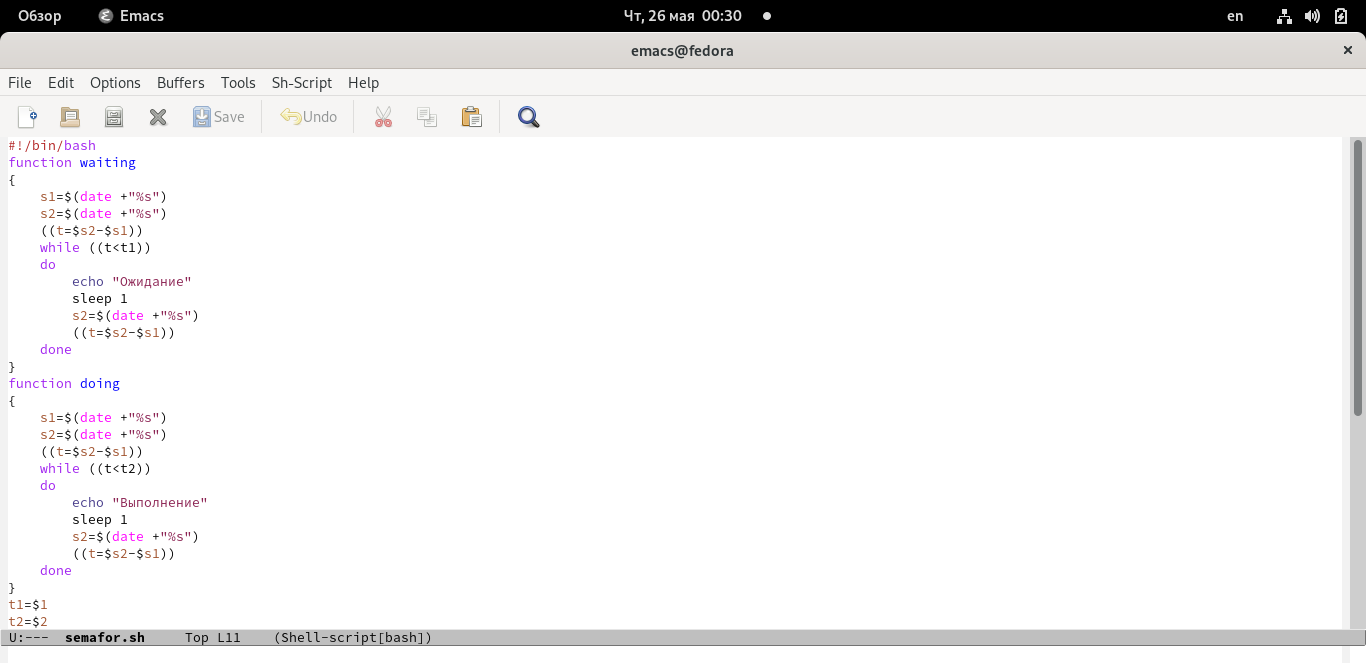
*Командный процессор* (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера.В операционных системахтипа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: - оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux,содержащая базовый,но при этом полный набор функций; - С-оболочка (или csh)—надстройка на оболочкой Борна,использующая С-подобный синтаксис команд с возможностью сохранения истории выполнения команд; - оболочка Корна (или ksh)—напоминает оболочку С,но операторы управления программой совместимы с операторами оболочки Борна; - BASH — сокращение от Bourne Again Shell (опять оболочка Борна),в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation). POSIX (Portable Operating System Interface for Computer Environments)— набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна.

# 3 Выполнение лабораторной работы

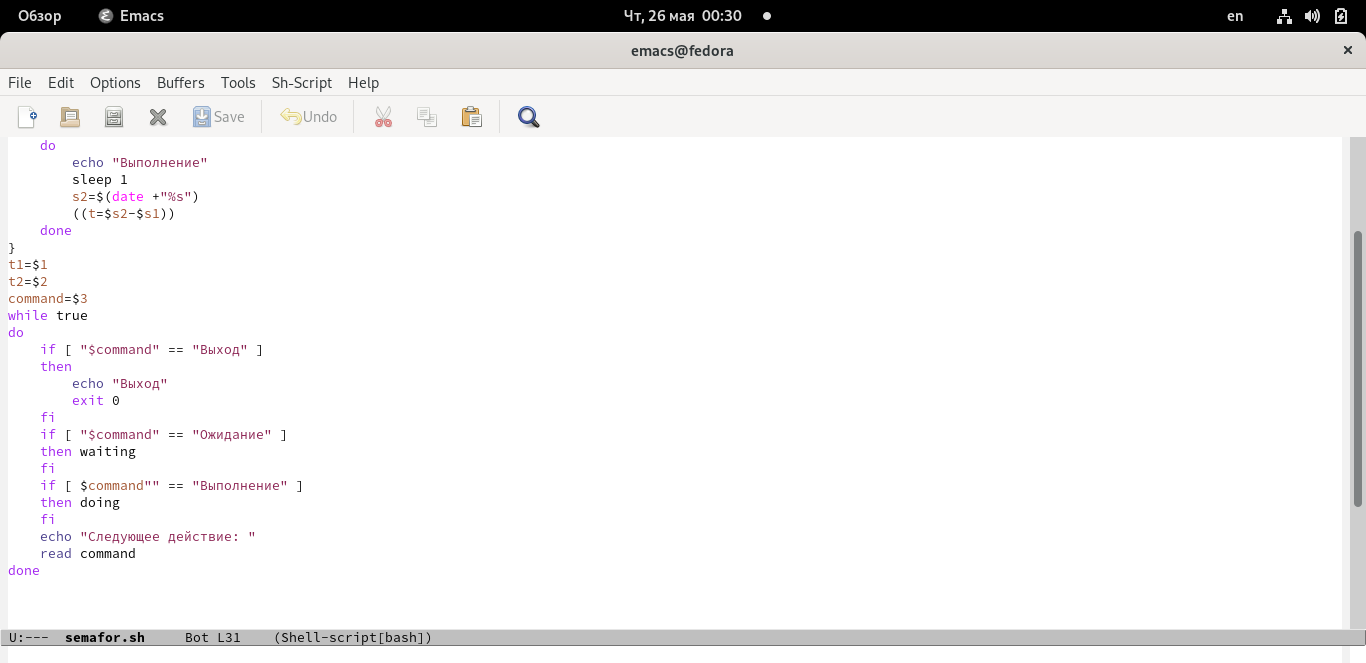
##1

Напишем командный файл, реализующий упрощённый механизм семафоров. Командный файл в течение некоторого времени t1 будет дожидаться освобожденияресурса,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 виртуальномтерминале в фоновом режиме,перенаправив его вывод в другой (> /dev/tty#,где # —номертерминала куда перенаправляется вывод),в которомтакже запущен этот файл,но не фоновом,а в привилегированном режиме. Доработать программутак,чтобы имелась возможность взаимодействия трёх и более процессов.

Для этого мы создаем файл semafor.sh и пишем программу (скриншоты 1, 2)

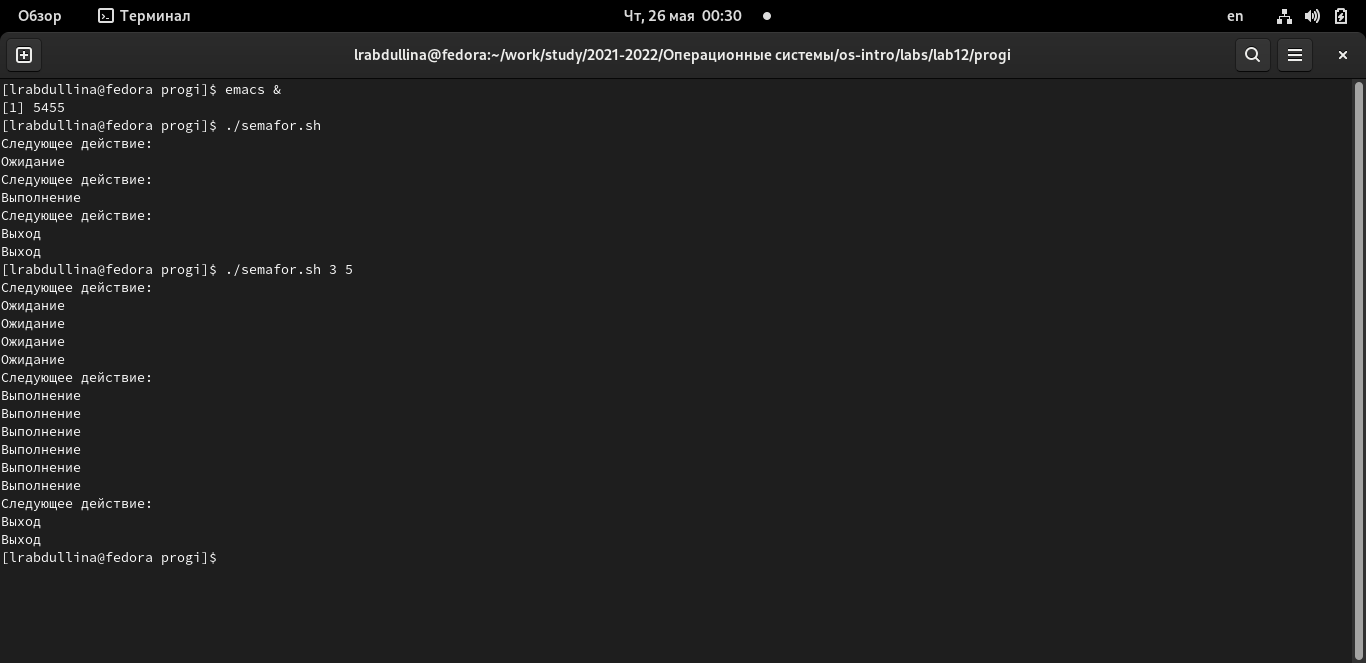


Скриншот 1: Написанная программа



Скриншот 2: Написанная программа

Не забудем сделать наш файл с программой исполняемым через команду chmod +x semafor.sh. После этого вводим команды в консоли ./semafor и ./semafor 3 5 и смотрим на результат. (скриншот 3)

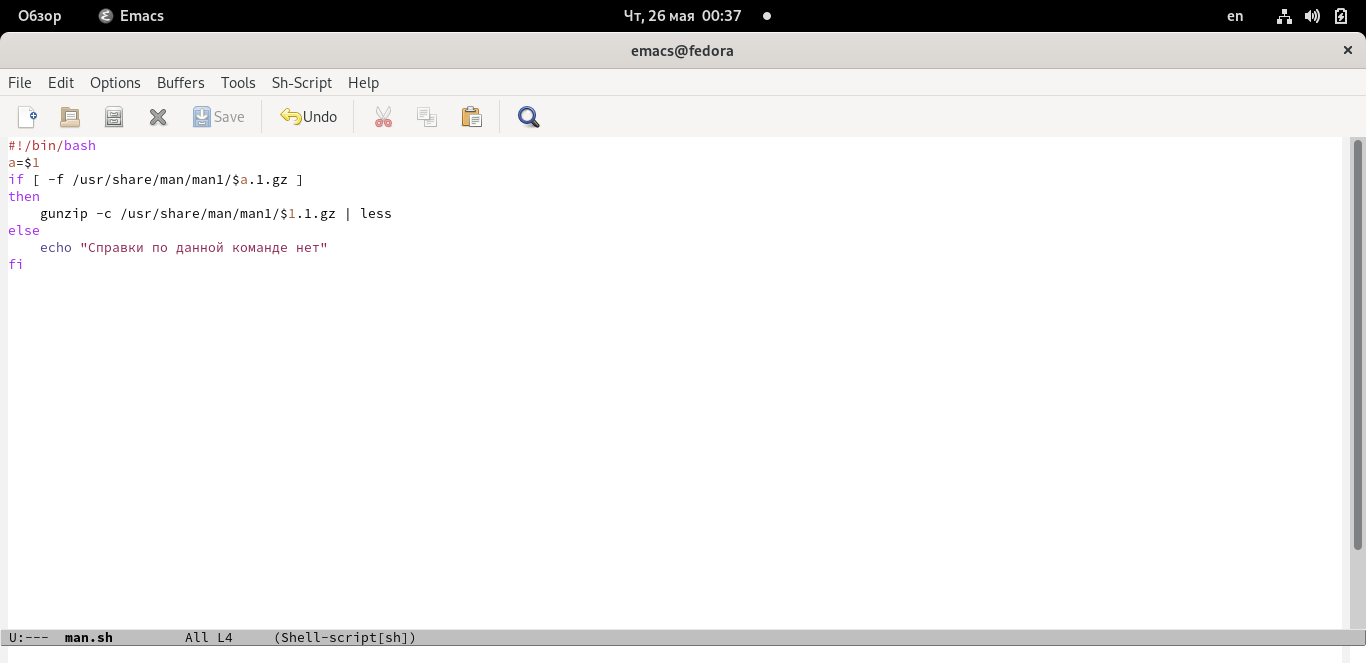


Скриншот 3: Проверка - работает успешно

## 3.1 2

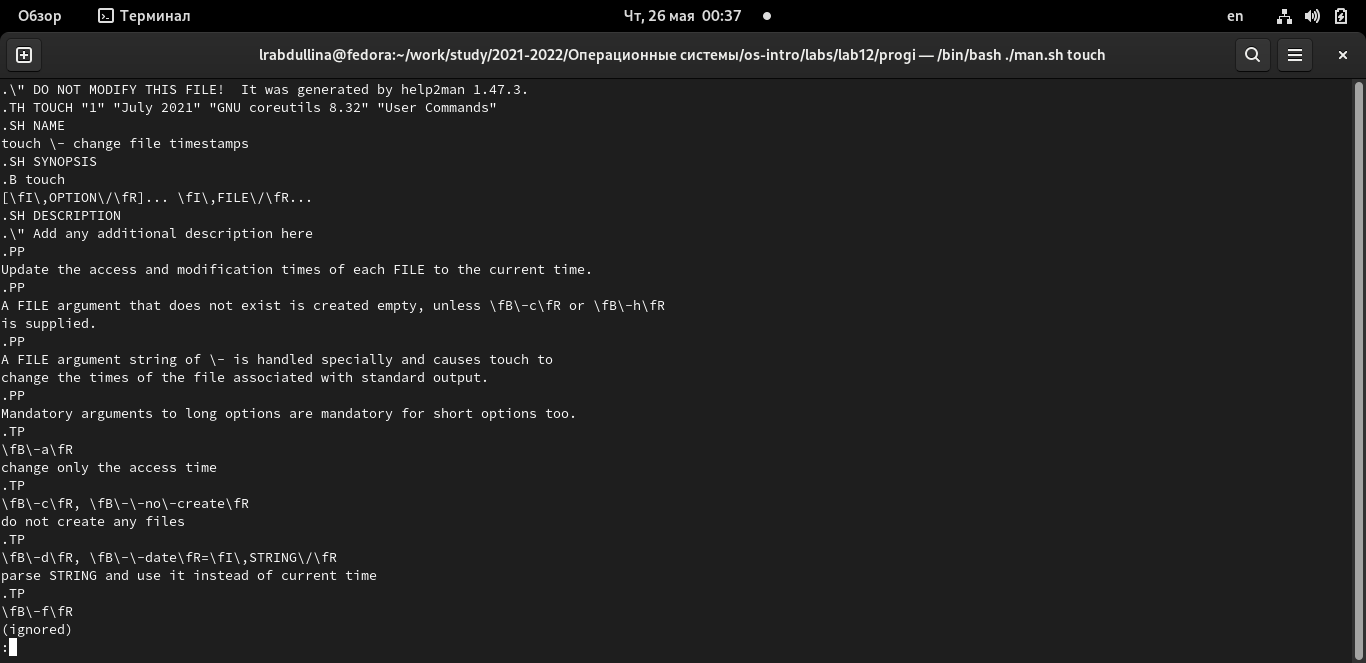
Реализуем команду man с помощью командного файла.Изучим содержимое каталога /usr/share/man/man1. В нем находятся архивытекстовых файлов,содержащих справку по большинству установленных в системе программ и команд.Каждый архив можно открытькомандой less сразу же просмотрев содержимое справки.Командный файл должен получать в виде аргумента командной строки название команды и ввиде результата выдавать справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге man1

Для этого мы создаем файл man.sh и пишем код (скриншот 4)

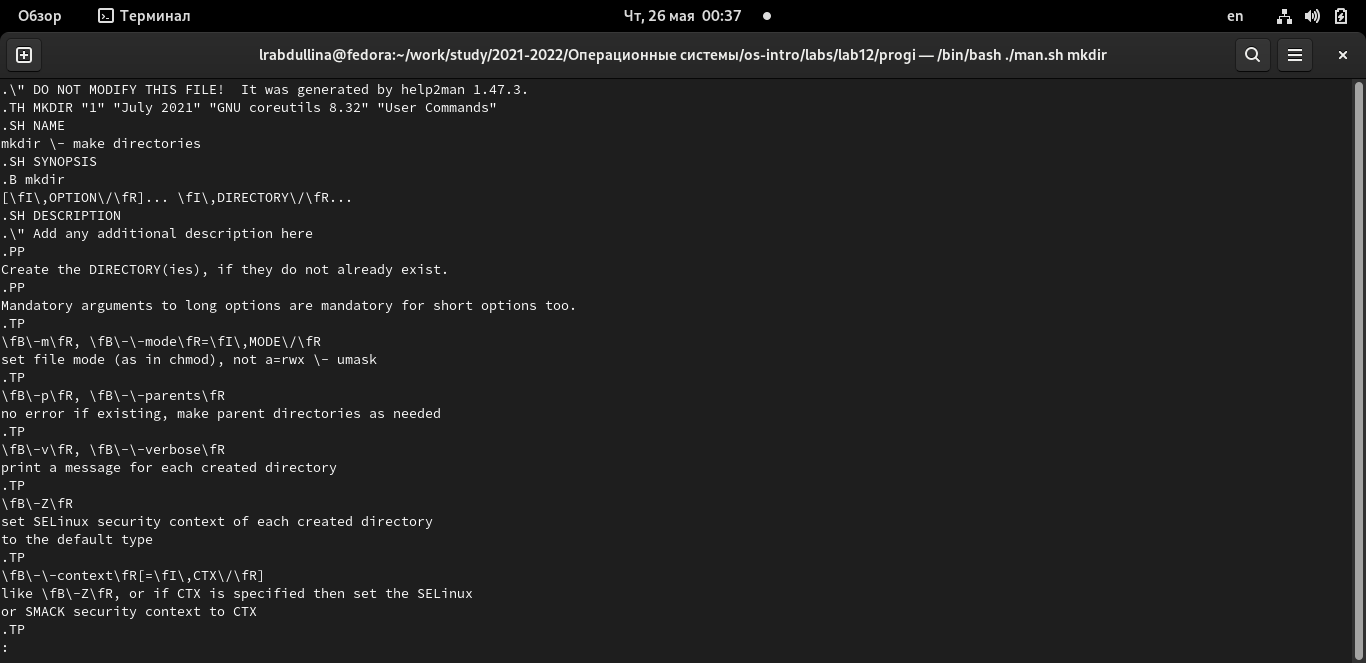


Скриншот 4: Написанная программа

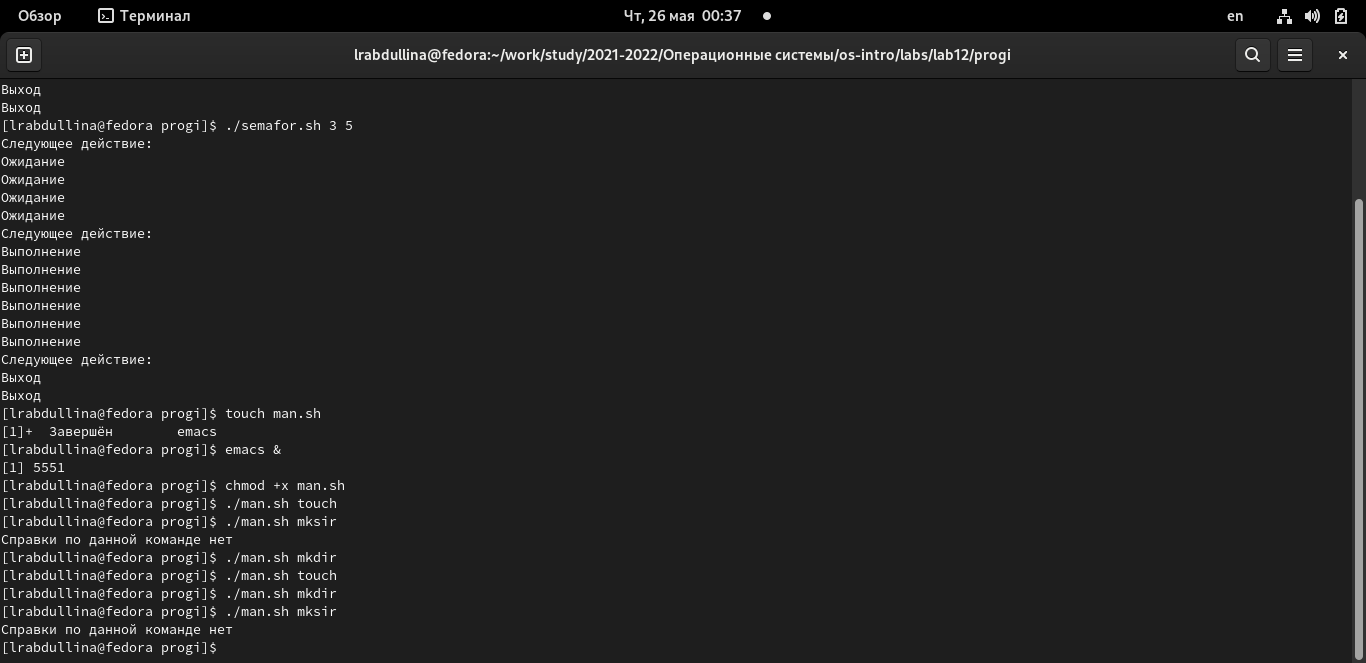
Сделаем файл исполняемым через команду chmod +x man.sh. И проверим его работу. Пишем команды ./man.sh touch ./man.sh mkdir ./man.sh mksir (скриншоты 5, 6, 7)



Скриншот 5: Проверка - работает успешно



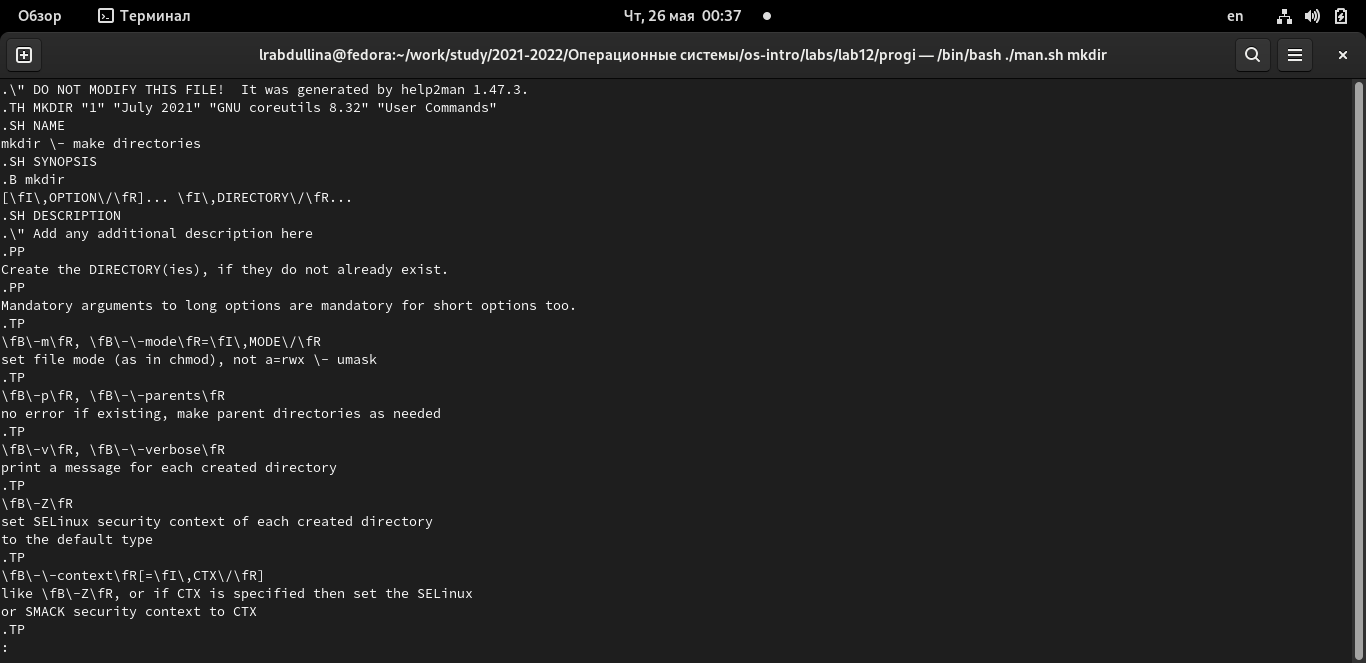
Скриншот 6: Проверка - работает успешно



Скриншот 7: Проверка - работает успешно

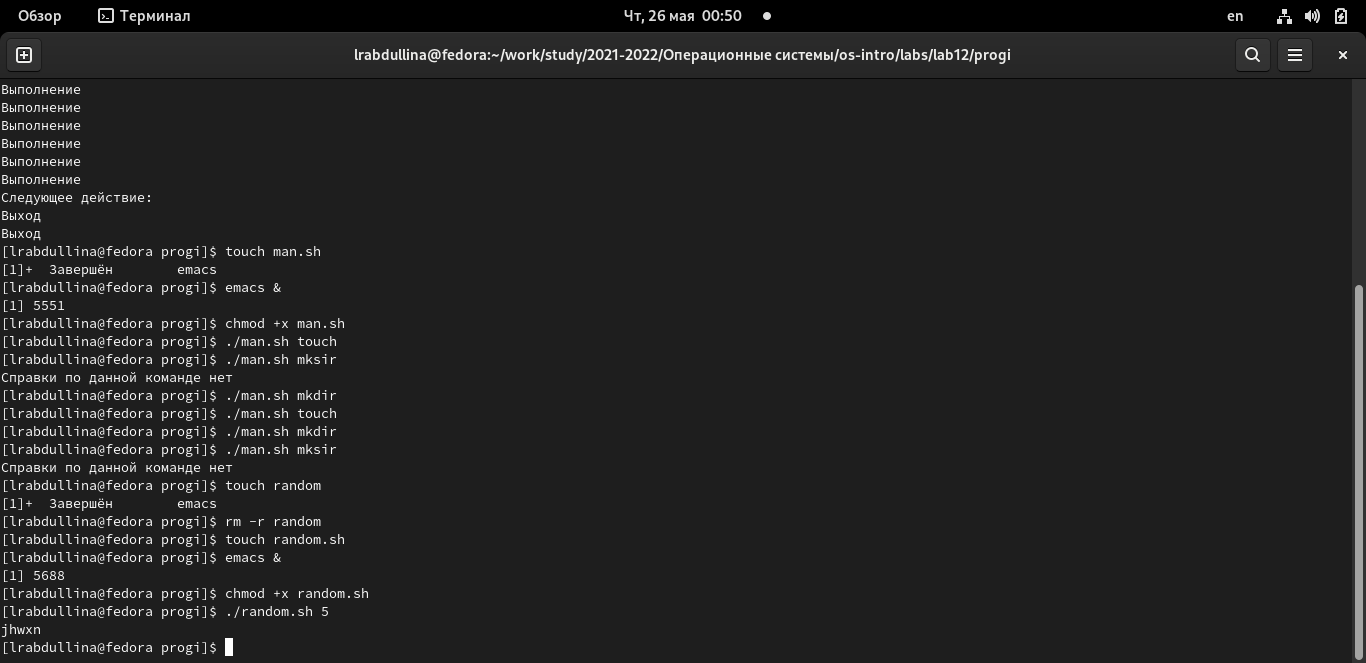
1. Используя встроенную переменную $RANDOM,напишем командный файл,генерирующий случайную последовательность букв латинского алфавита.Учтем,что $RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767

Для начала созданим новый файл random.sh и напишем программу. (скриншот 8)



Скриншот 8: Написанная программа

Сделаем файл исполняемым через команду chmod +x random.sh. И проверим его работу. Напишем случайное число и посмотрим на результат. (скриншот 9)



Скриншот 9: Проверка - работает успешно

# 4 Контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке:

* 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. Как объединить (конкатенация) несколько строк в одну?

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. Найдите информацию об утилите seq.Какими иными способами можно реализовать её функционал при программировании на bash?

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. Какой результатдаствычисление выражения $((10/3))?

Результатом данного выражения $((10/3)) будет 3, потому что это целочисленное деление без остатка.

1. Скажите кратко основные отличия командной оболочки zsh от bash

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. Проверьте,верен ли синтаксис данной конструкции

* for ((a=1; a <= LIMIT; a++))

Синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать $ перед переменными ().

1. Сравните язык bash с какими-либо языками программирования.Какие преимущества у bash по сравнению с ними? Какие недостатки?

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

# 5 Выводы

В ходе лабораторной работы мы изучили основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованиемлогических управляющих конструкций и циклов.

# 6 Список литературы

https://esystem.rudn.ru/course/view.php?id=5790 :::