

# **Отчёт по лабораторной работе 12**

**Дисциплина: Операционные системы**

Волчок Кристина Александровна

# Содержание

1	Цель работы	4
2	Задания	5
3	Ход работы:	7
4	Контрольные вопросы	12
5	Вывод:	15

## Список иллюстраций

3.1	Создаю файл . . . . .	7
3.2	Пишу скрипт . . . . .	8
3.3	Программа . . . . .	8
3.4	Создаю файл . . . . .	9
3.5	Написала скрипт . . . . .	9
3.6	Программа . . . . .	9
3.7	Программа . . . . .	10
3.8	Программа . . . . .	10
3.9	Создаю текстовый файл . . . . .	11
3.10	Пишу скрипт . . . . .	11
3.11	Программа . . . . .	11

# 1 Цель работы

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

## 2 Задания

1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени  $t_1$  дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени  $t_2 < t_1$ , 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 в фоновом, а в привилегированном режиме. Доработать программу так, чтобы имела возможность взаимодействия трёх и более процессов.
2. Реализовать команду `man` с помощью командного файла. Изучите содержимое каталога `/usr/share/man/man1`. В нём 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 псевдослучайные числа в диапазоне от 0 до 32767.

### 3 Ход работы:

1. Написала командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени  $t_1$  дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени  $t_2 < t_1$ , также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запустила 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 имела возможность взаимодействия трёх и более процессов.(рис. 3.1)(рис. 3.2)(рис. 3.3)

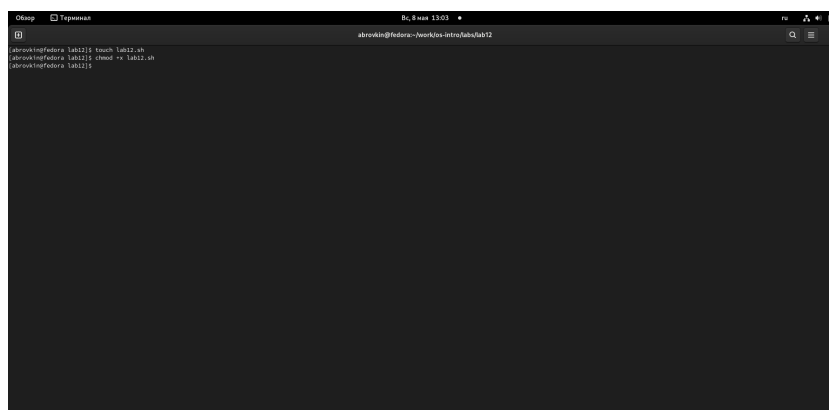


Рис. 3.1: Создаю файл

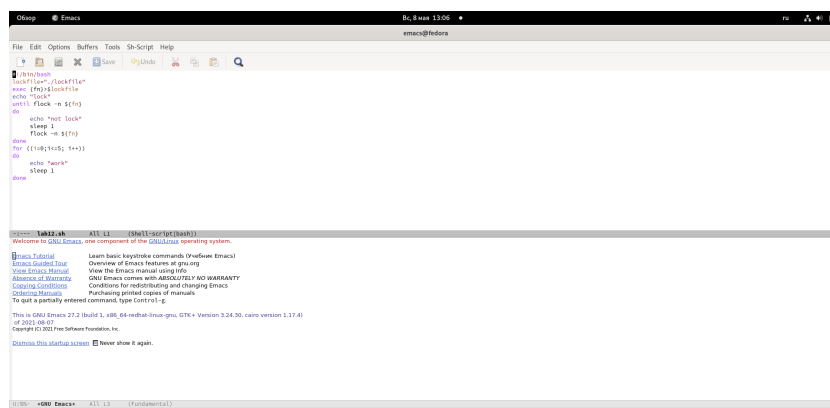


Рис. 3.2: Пишу скрипт

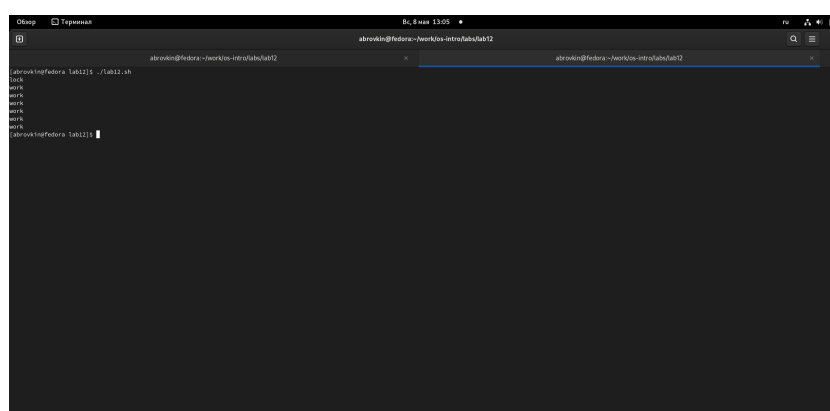


Рис. 3.3: Программа

2. Реализовала команду man с помощью командного файла. Изучила 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. 3.4)(рис. 3.5)(рис. 3.6)(рис. 3.7)(рис. 3.8)



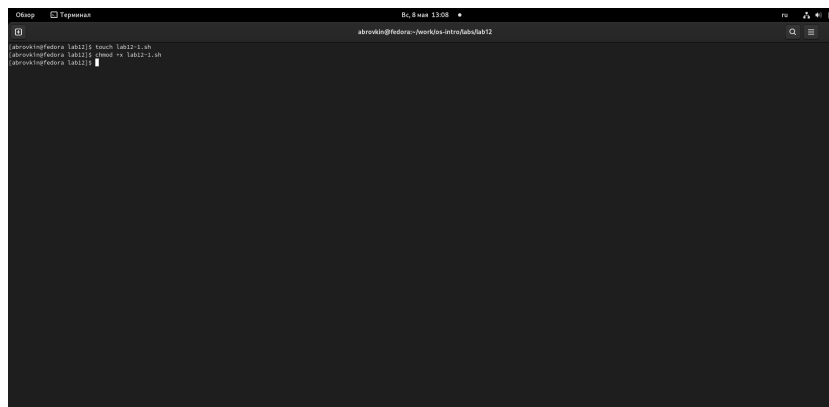


Рис. 3.4: Создаю файл

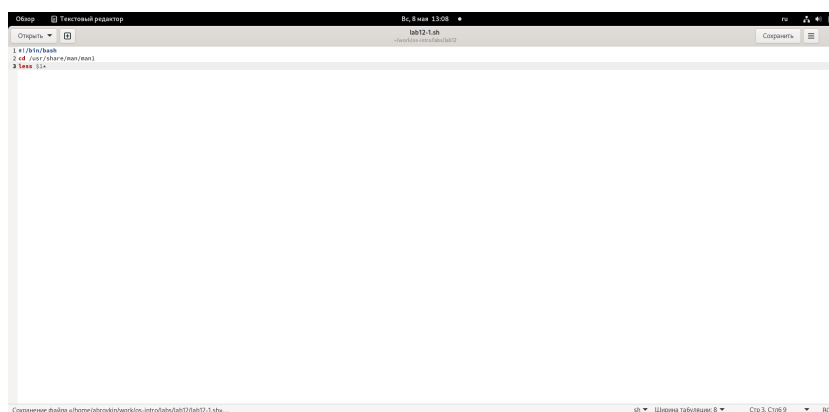


Рис. 3.5: Написала скрипт

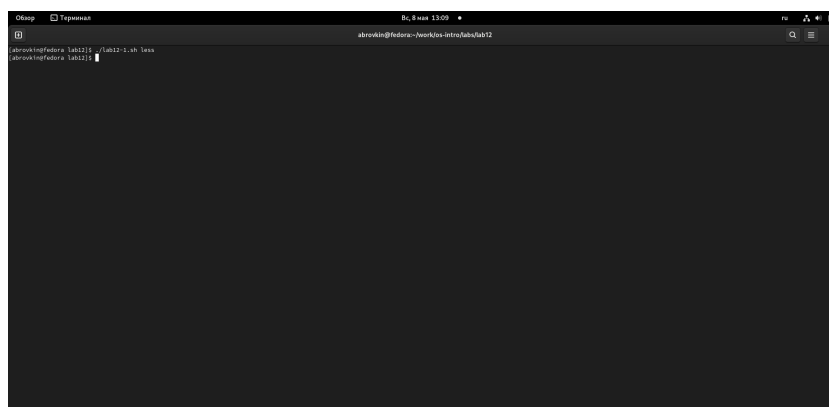
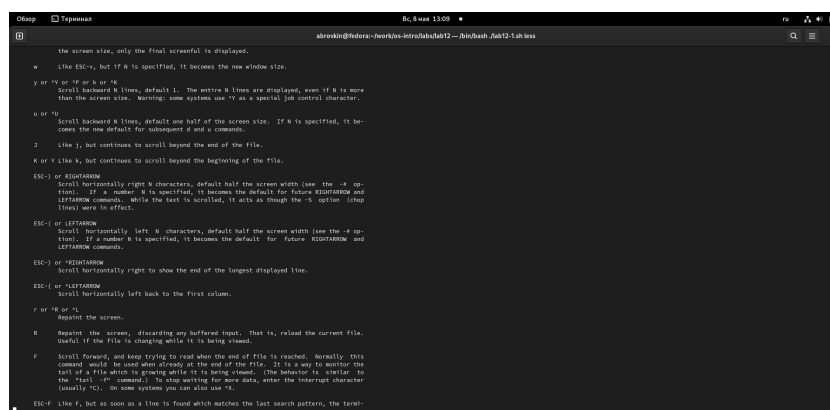
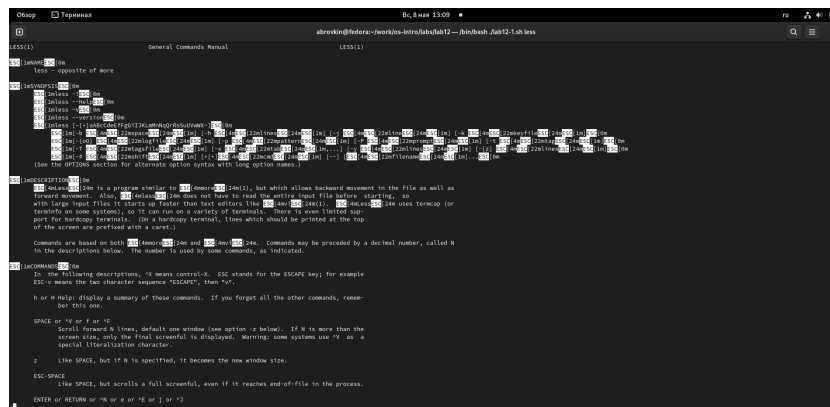


Рис. 3.6: Программа



3. Используя встроенную переменную \$RANDOM, написала командный файл, генерирующий случайную последовательность букв латинского алфавита. Учла, что \$RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767. (рис. 3.9)(рис. 3.10)(рис. 3.11)

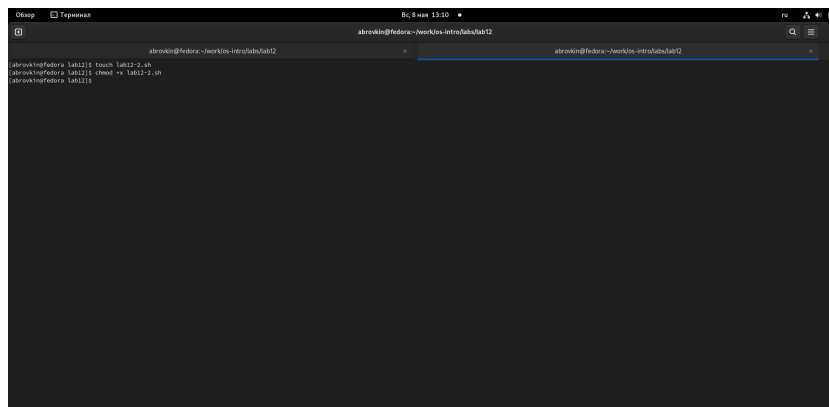


Рис. 3.9: Создаю текстовый файл

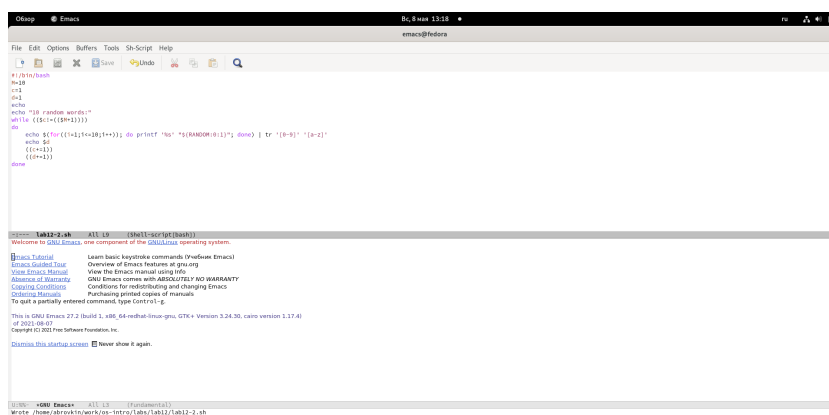


Рис. 3.10: Пишу скрипт

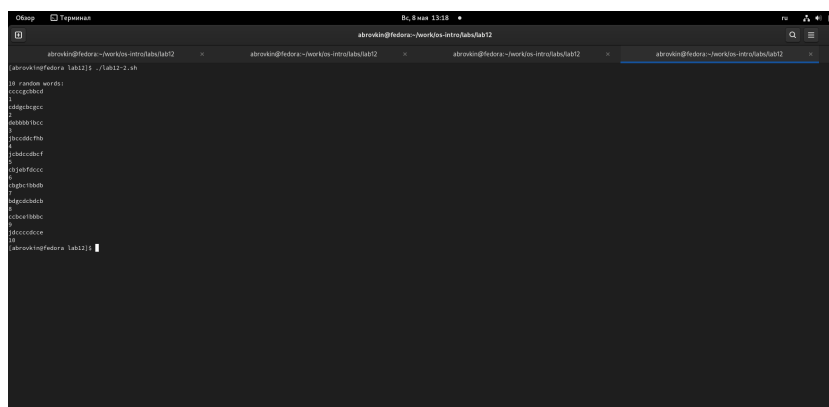


Рис. 3.11: Программа

## 4 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 `$((10/3))` будет 3, потому что это целочисленное деление без остатка.
- 5) Отличия командной оболочки `zsh` от `bash`:
- В `zsh` более быстрое автодополнение для `cd` с помощью `Tab`
  - В `zsh` существует калькулятор `zcalc`, способный выполнять вычисления внутри терминала
  - В `zsh` поддерживаются числа с плавающей запятой
  - В `zsh` поддерживаются структуры данных «хэш»
  - В `zsh` поддерживается раскрытие полного пути на основе неполных данных
  - В `zsh` поддерживается замена части пути
  - В `zsh` есть возможность отображать разделенный экран, такой же как разделенный экран `vim`
- 6) `for ((a=1; a <= LIMIT; a++))` синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать `$` 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 скорости выполнения этого скрипта
  - Скрипты, написанные на bash, нельзя запустить на других операционных системах без дополнительных действий

## **5 Вывод:**

В ходе проделанной лабораторной работы я изучила основы программирования в оболочке ОС UNIX, научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.