Отчет к №12 лабараторной работе

Операционные системы

Газизова Регина

Содержание

# 1 Цель работы

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

# 2 Задание

1. Написать командный файл, реализующий упрощённый механизм семафоров. Ко-мандный файл должен втечение некоторого времениt1дожидаться освобожденияресурса,выдавая об этом сообщение,а дождавшись его освобождения,использоватьего в течение некоторого времениt2<>t1, также выдавая информацию о том, чторесурс используется соответствующим командным файлом (процессом).Запуститькомандный файл в одном виртуальномтерминале в фоновом режиме,перенаправивего вывод в другой (> /dev/tty#,где#—номертерминала куда перенаправляетсявывод),в которомтакже запущен этотфайл,но не фоновом,а в привилегированномрежиме.Доработатьпрограммутак,чтобыимеласьвозможностьвзаимодействиятрёхи более процессов.
2. Реализовать командуmanс помощью командного файла.Изучите содержимое ката-лога/usr/share/man/man1.В нем находятся архивытекстовых файлов,содержащихсправку по большинству установленных в системе программ и команд.Каждый архивможнооткрытькомандойlessсразужепросмотревсодержимоесправки.Командныйфайлдолженполучатьввидеаргументакоманднойстрокиназваниекомандыиввидерезультата выдавать справку об этой команде или сообщение об отсутствии справки,если соответствующего файла нет в каталогеman1.
3. Используя встроенную переменнуюRANDOMвыдаёт псевдослучайные числа в диапазоне от 0 до 32767

# 3 Выполнение лабораторной работы

1. Напишем 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).

Для выполнения данной задачи создадим файл semafor.sh и откроем его в emacs (рис.1).

Рис. 1: Создание файла для 1 задания

Рис. 1: Создание файла для 1 задания

В файле напишем соответствующий скрипт (рис.2) и проверим его работу (команда ./semafor.sh 2 4), предварительно добавив права на выполнение (команда chmod +x semafor.sh) (рис.3).

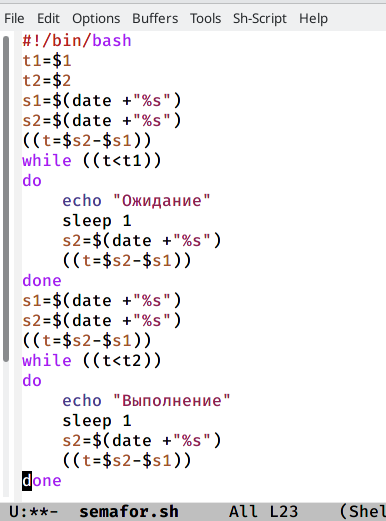


Рис. 2: Скрипт для 1 задания

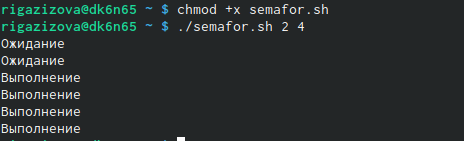


Рис. 3: Выполнение командного файла

Затем изменим скрипт так, чтобы можно было запускать командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой (рис.4-5).

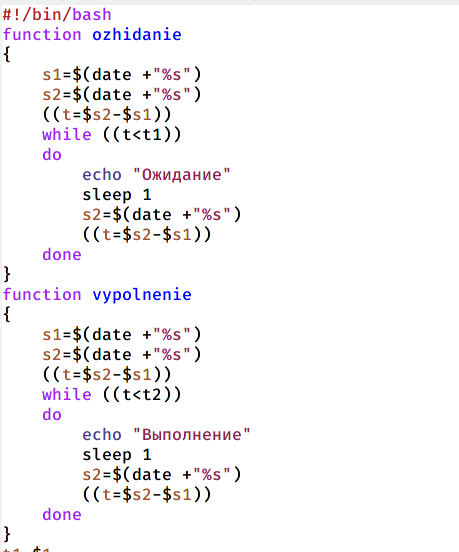


Рис. 4: Измененный скрипт (часть 1)



Рис. 5: Измененный скрипт (часть 2)

Проверим его работу (например, команда ./semafor.sh 2 4 Ожидание > /dev/pts/1) и увидим, что нам отказано в доступе . Но при этом скрипт работает корректно (рис.7) при вводе команды ./semafor.sh 2 4 Ожидание.

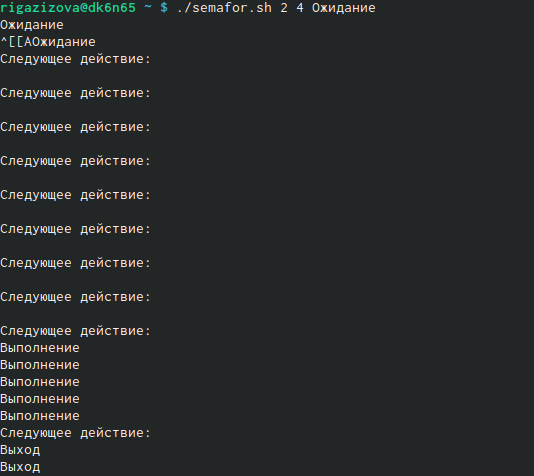


Рис. 6: Выполнение командного файла

1. Перед тем как приступить к выполнению 2 задания, изучим содержимое каталога /usr/share/man/man1 (рис.8). В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд.

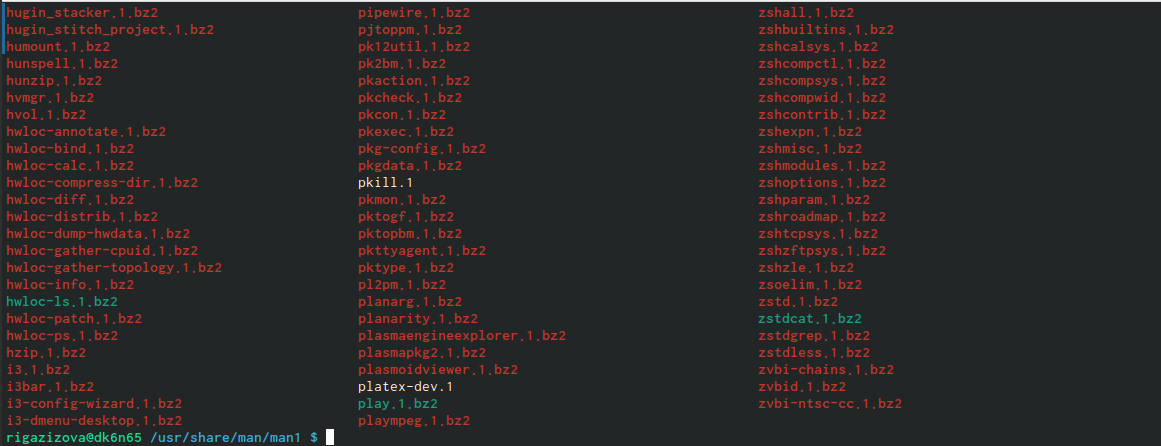


Рис. 7: Содержимое каталога

Реализуем команду man с помощью командного файла. Для этого создадим файл man.sh и откроем его в emacs. Напишем скрипт для выполнения задания (рис.9).

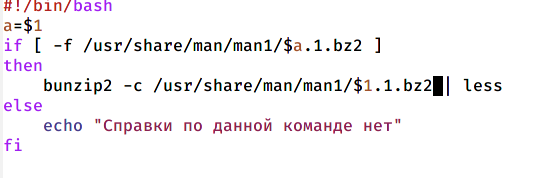


Рис. 8: Скрипт для 2 задания

Проверим его работу (команды ./man.sh mkdir, ./man.sh rm, ./man.sh cat), предварительно дав ему право на выполнение с помощью команды chmod +x man.sh (рис.10). Результаты работы трех команд представлены на рисунках 11-13.

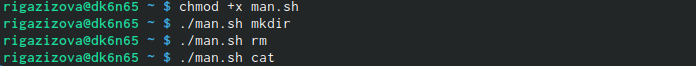


Рис. 9: Выполнение командного файла

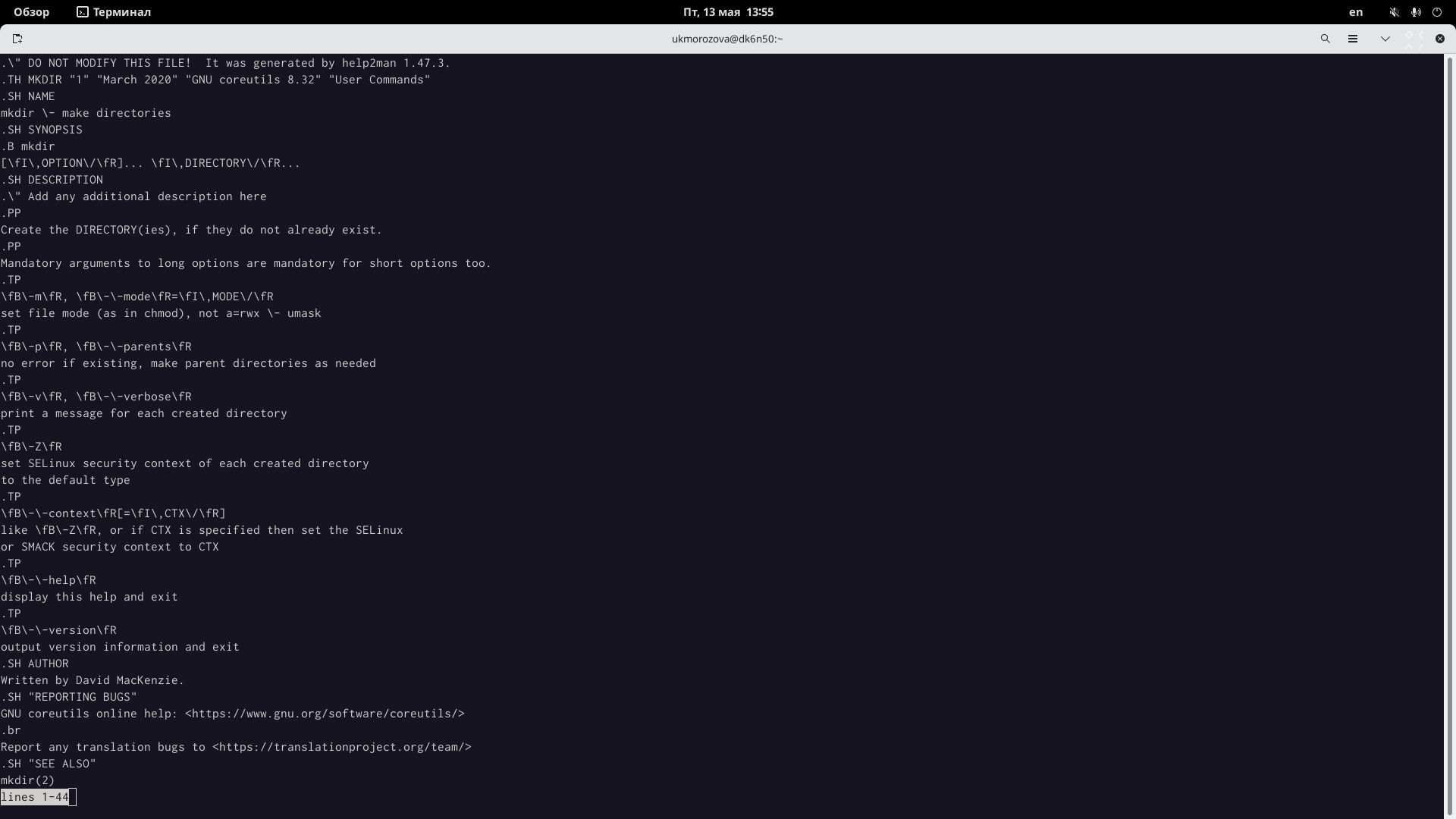


Рис. 10: mkdir

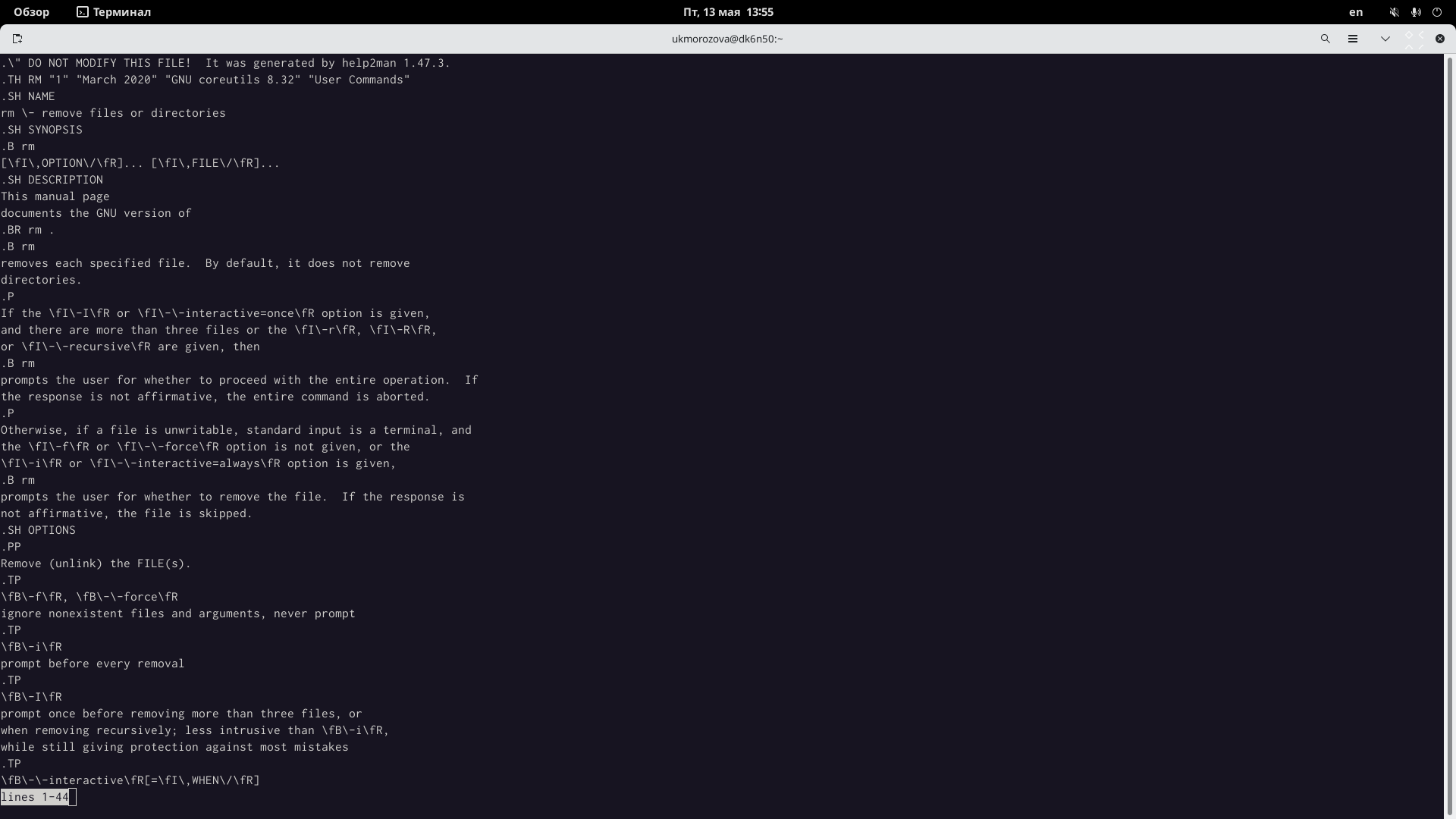


Рис. 11: rm

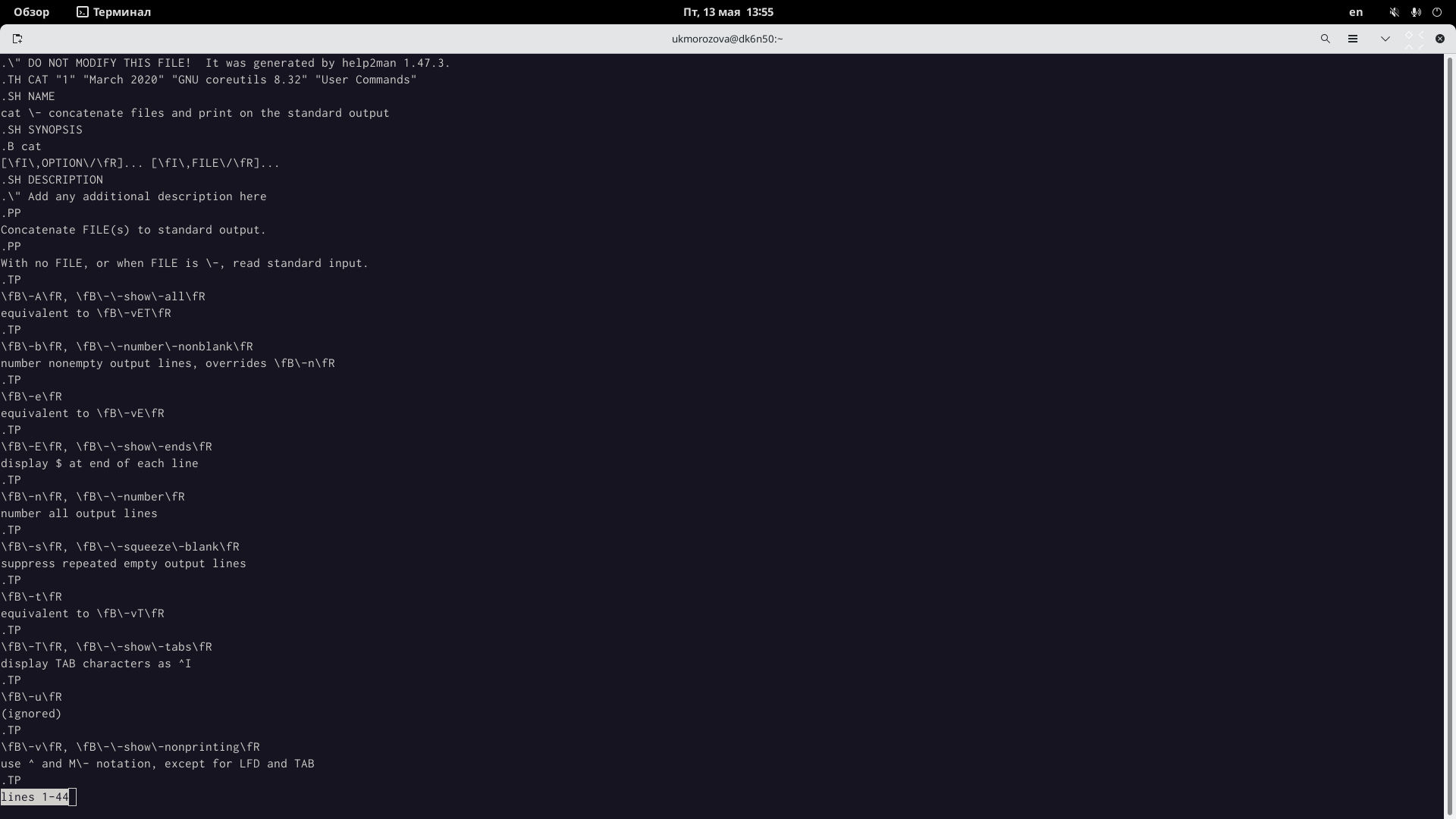


Рис. 12: cat

1. Используя встроенную переменную $RANDOM, напишим командный файл, генерирующий случайную последовательность букв латинского алфавита. Для этого создадим файл random.sh и откроем его в emacs.

Напишем скрипт для выполнения 3 задания (рис.14).

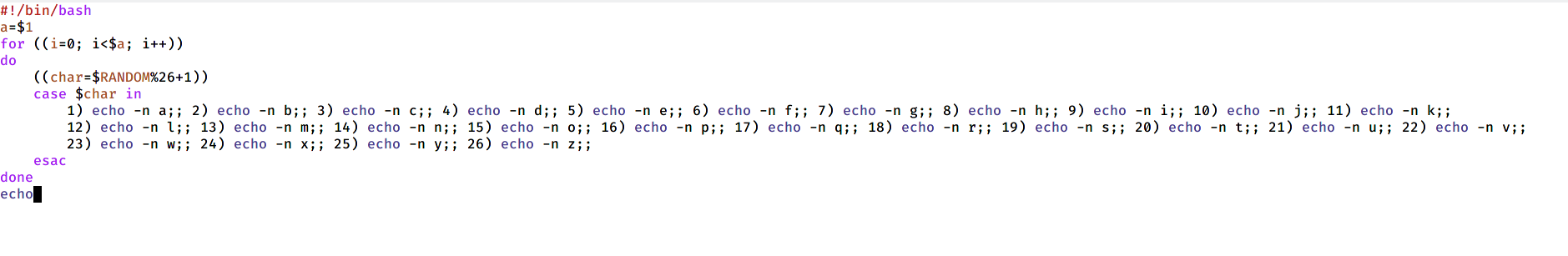


Рис. 13: Скрипт для 3 задания

Проверим его работу (команда ./random.sh 158), предварительно дав ему право на выполнение с помощью команды chmod +x random.sh (рис.15).

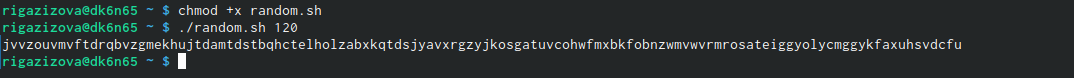


Рис. 14: Выполнение командного файла

# 4 Выводы

Я изучила основы программирования в оболочке ОС UNIX и научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

# 5 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=“VAR2”

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.