Лабораторная работа №13

Дисциплина: Операционные системы

Егорова Александра

Содержание

# Цель работы

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

# Задание

1. Написать 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). 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 коман- де 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.

# Выполнение лабораторной работы

1. Cоздала файл: pr1.sh и написала соответствующий скрипт. Написала 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). (рис. -fig. 1) (рис. -fig. 2)

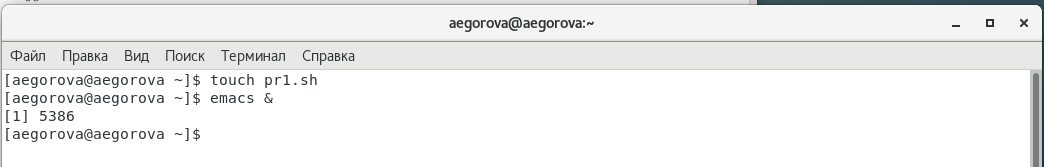


Figure 1: Создаем файл

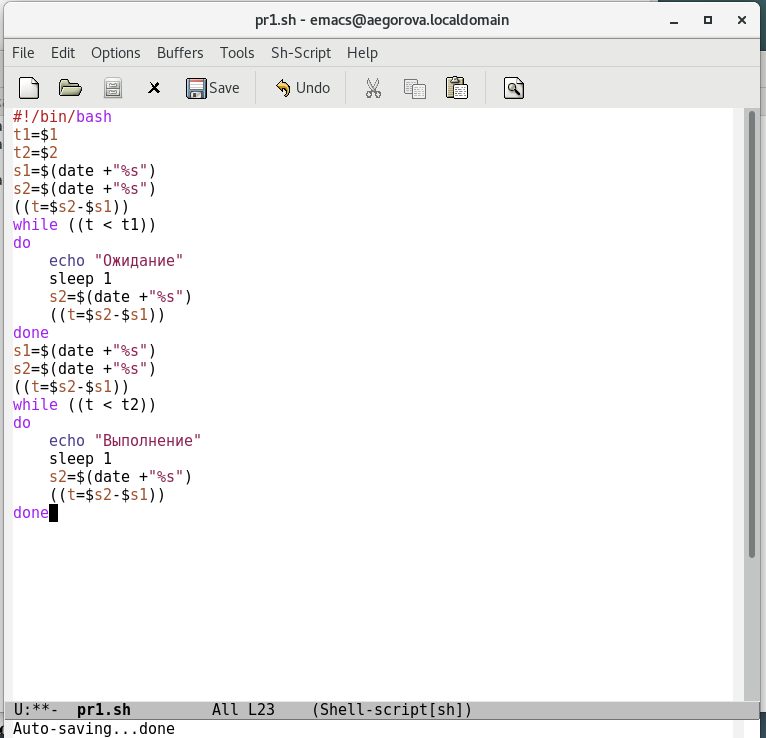


Figure 2: Первый скрипт

Проверяю работу написанного скрипта (команда «./pr1.sh 3 6»), предварительно добавив право на исполнение файла («chmod +x pr1.sh»). Скрипт работает корректно. (рис. -fig. 3)

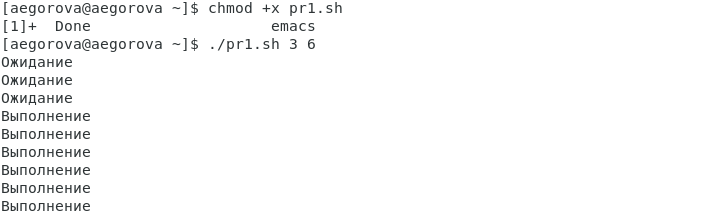


Figure 3: Проверка скрипта

После этого я изменила скрипт так, чтобы его можно было выполнять в нескольких терминалах и проверила его работу («./pr1.sh 2 3 Ожидание > dev/tty# , где # — номер терминала куда перенаправляется вывод»). Ни одна из команд не сработала (выводит сообщение “Отказано в доступе”). При этом скрипт работает корректно. (рис. -fig. 4) (рис. -fig. 5) (рис. -fig. 6)

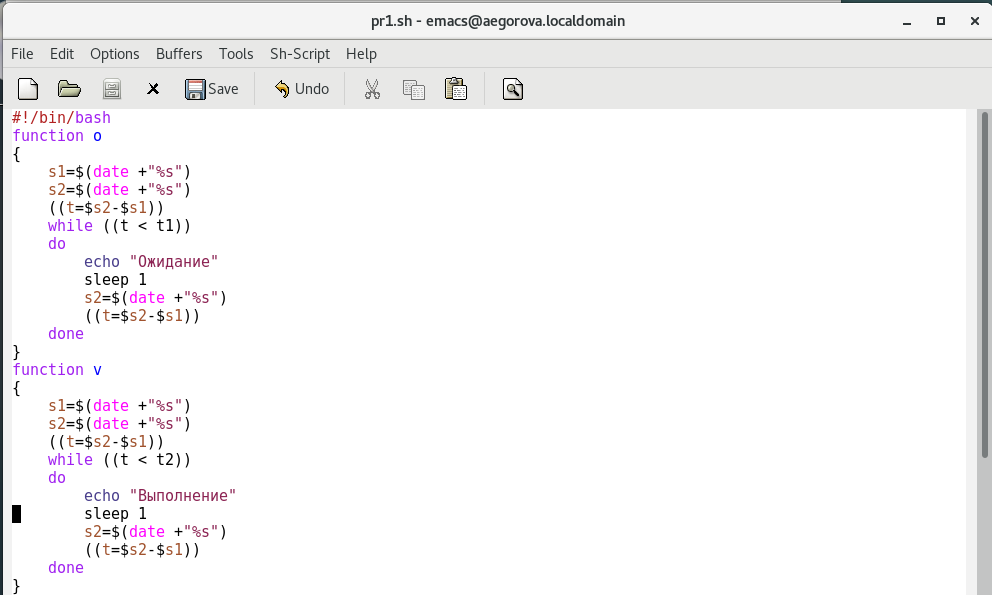


Figure 4: Изменяем первый скрипт

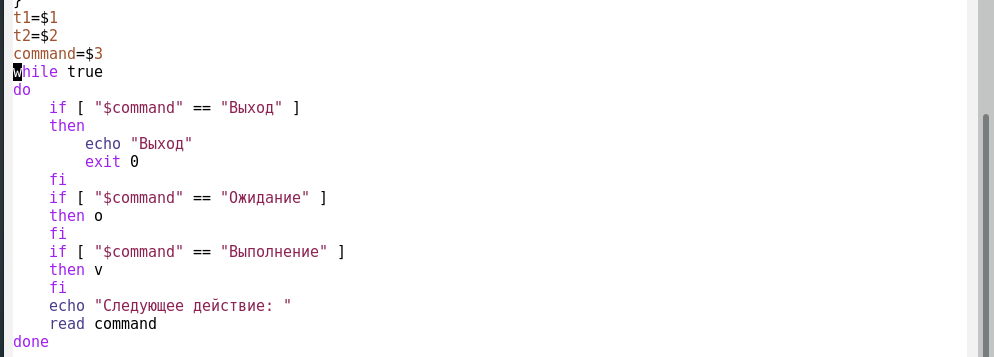


Figure 5: Изменяем первый скрипт

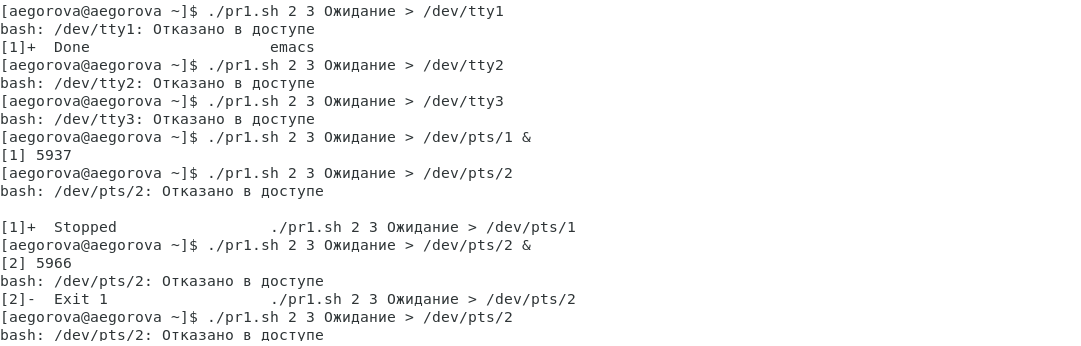


Figure 6: Проверка скрипта

1. Реализуем 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 отсутствиисправки, если соответствующего файла нет в каталоге man1. (рис. -fig. 7)

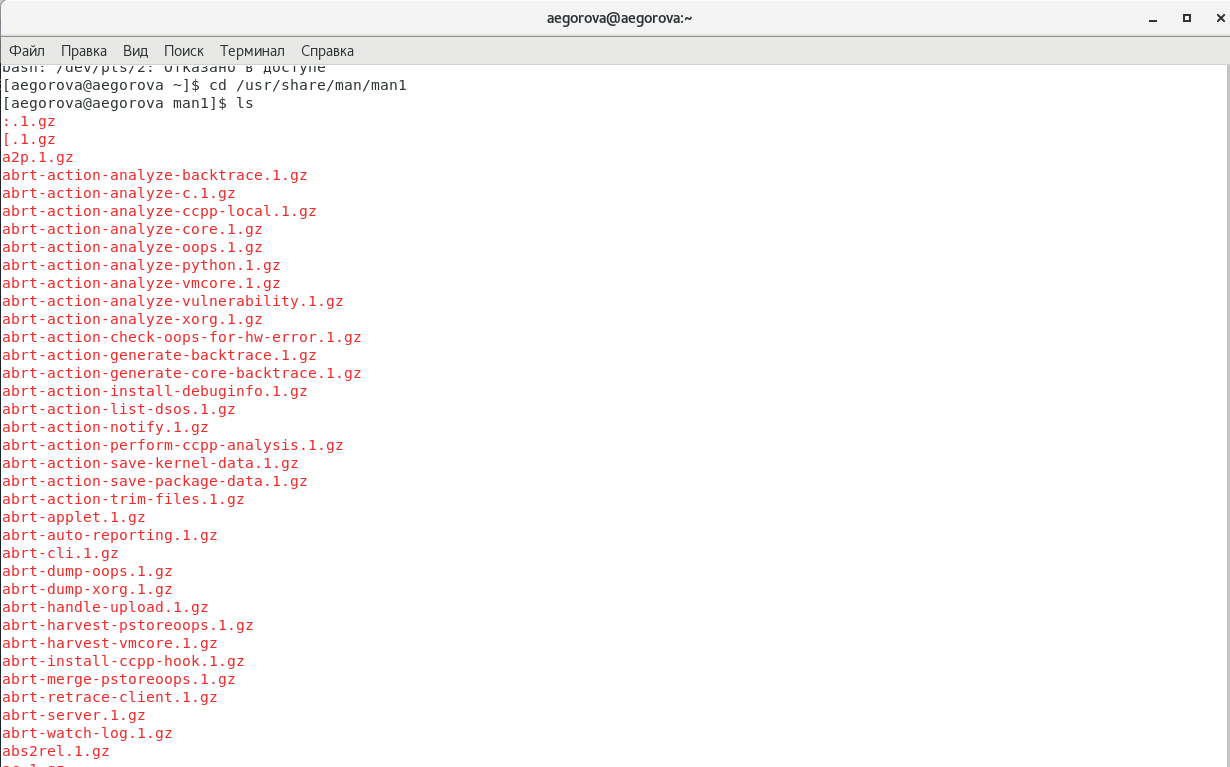


Figure 7: Содержимое каталога

Создала файл: pr2.sh и написала соответствующий скрипт. Далее я проверила работу написанного скрипта («./pr2.sh ls» и «./pr2.sh mkdir»), предварительно добавив право на исполнение файла («chmod +x pr2.sh»). Скрипт работает корректно. (рис. -fig. 8) (рис. -fig. 9) (рис. -fig. 10) (рис. -fig. 11)

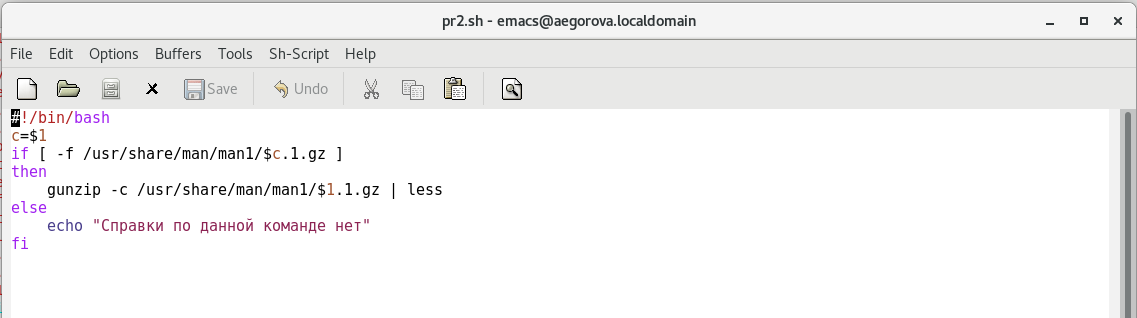


Figure 8: Второй скрипт

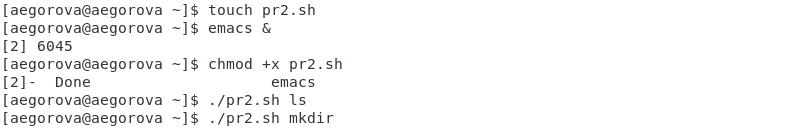


Figure 9: Проверка скрипта

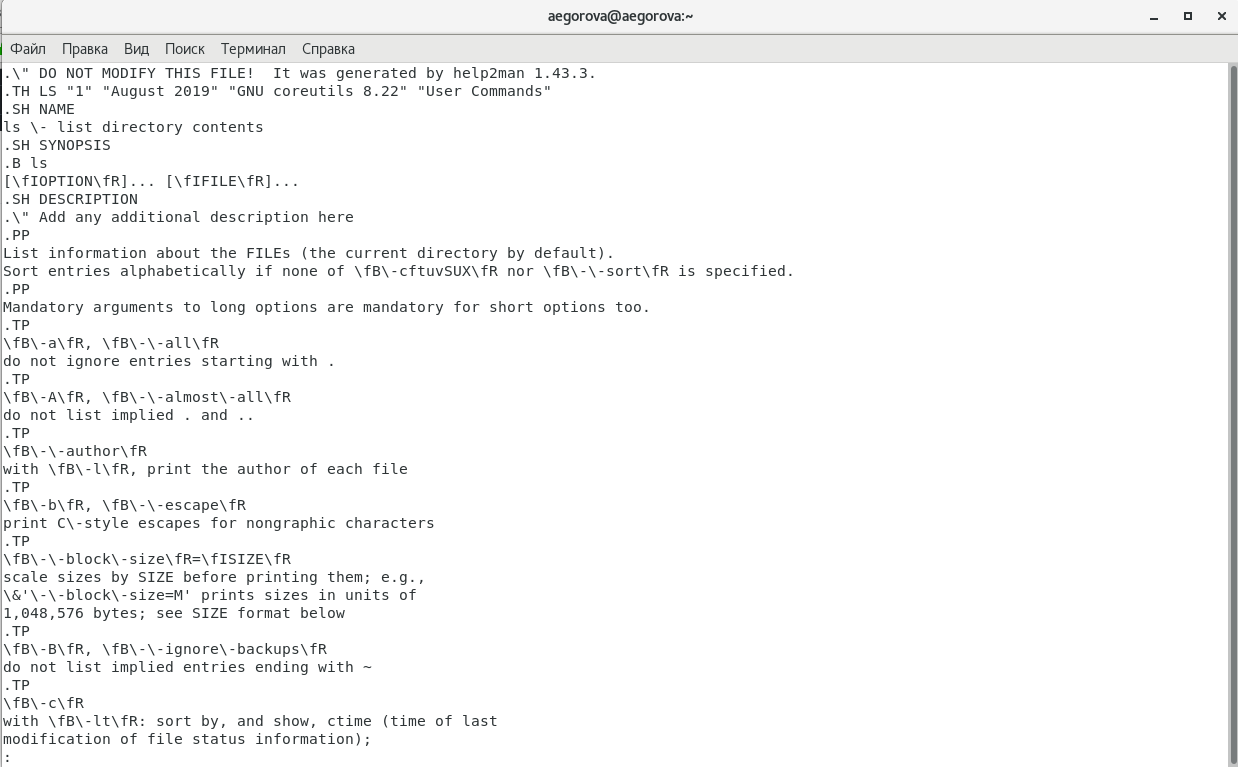


Figure 10: Проверка скрипта

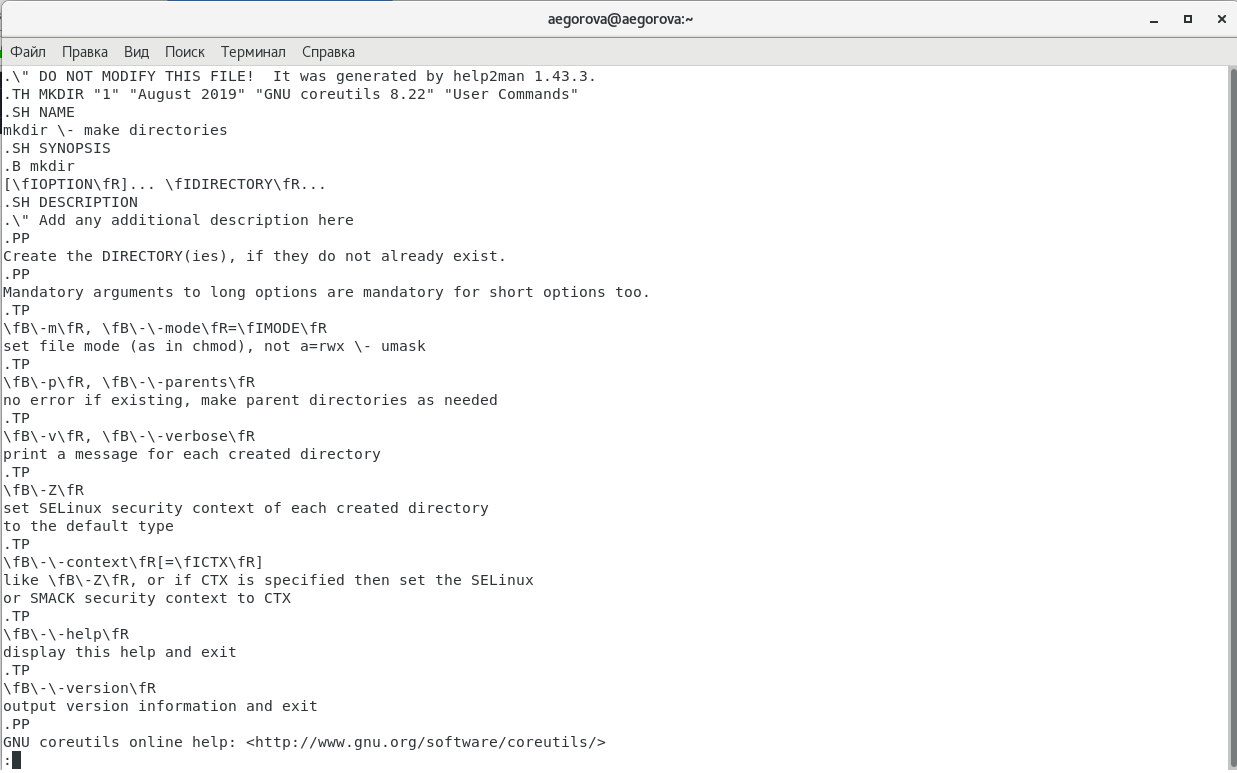


Figure 11: Проверка скрипта

1. Используя встроенную переменную $RANDOM, написала командный файл, генерирующий случайную последовательность букв латинского алфавита. Для данной задачи я создала файл: pr3.sh и написала соответствующий скрипт. (рис. -fig. 12)

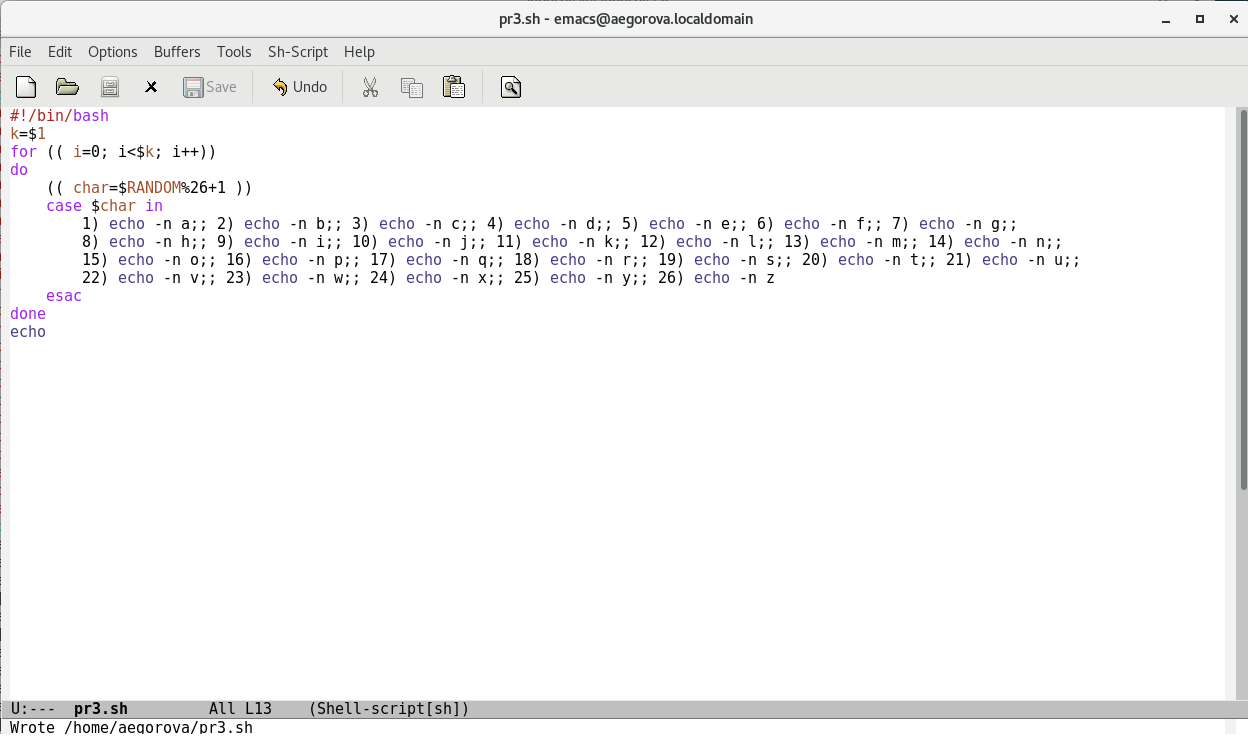


Figure 12: Третий скрипт

Далее я проверила работу написанного скрипта («./pr3.sh 5» и «./pr3.sh 13»), предварительно добавив право на исполнение файла («chmod +x pr3.sh»). Скрипт работает корректно. (рис. -fig. 13)



Figure 13: Проверка скрипта

# Выводы

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

# Контрольные вопросы

1. while [$1 != “exit”]. В данной строчке допущены следующие ошибки: 1) не хватает пробелов после первой скобки [ и перед второй скобкой ]; 2) 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 от ПЕРВОГО до ПОСЛЕДНЕГО шага INCREMENT. чиселПараметры: 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 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
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 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 другихоперационных системах без дополнительных действий.