Лабораторная работа №12

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Бабенко Артём Сергеевич

Содержание

[Цель работы 1](#_Toc104525821)

[Теоретическое введение 1](#_Toc104525822)

[Ход работы 2](#_Toc104525823)

[Выводы 6](#_Toc104525824)

[Контрольные вопросы 6](#_Toc104525825)

# Цель работы

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

# Теоретическое введение

Командные процессоры или оболочки – это программы, позволяющие пользователю взаимодействовать с компьютером. Их можно рассматривать как настоящие интерпретируемые языки, которые воспринимают команды пользователя и обрабатывают их. Поэтому командные процессоры также называют интерпретаторами команд. На языках оболочек можно писать программы и выполнять их подобно любым другим программам. UNIX обладает большим количеством оболочек. Наиболее популярными являются следующие четыре оболочки:

– оболочка Борна – первоначальная командная оболочка UNIX: базовый, но полный набор функций;

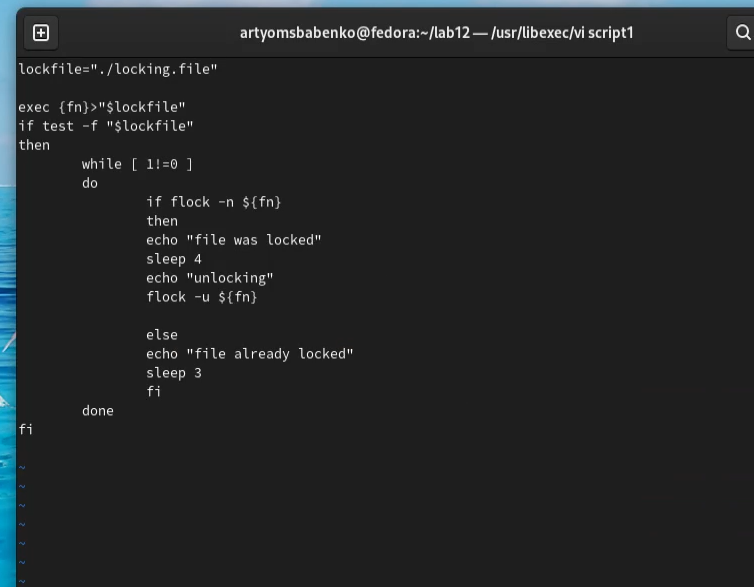
– С-оболочка – добавка университета Беркли к коллекции оболочек: она надстраивается над оболочкой Борна, используя С-подобный синтаксис команд, и сохраняет историю выполненных команд;

– оболочка Корна – напоминает оболочку С, но операторы управления программой совместимы с операторами оболочки Борна;

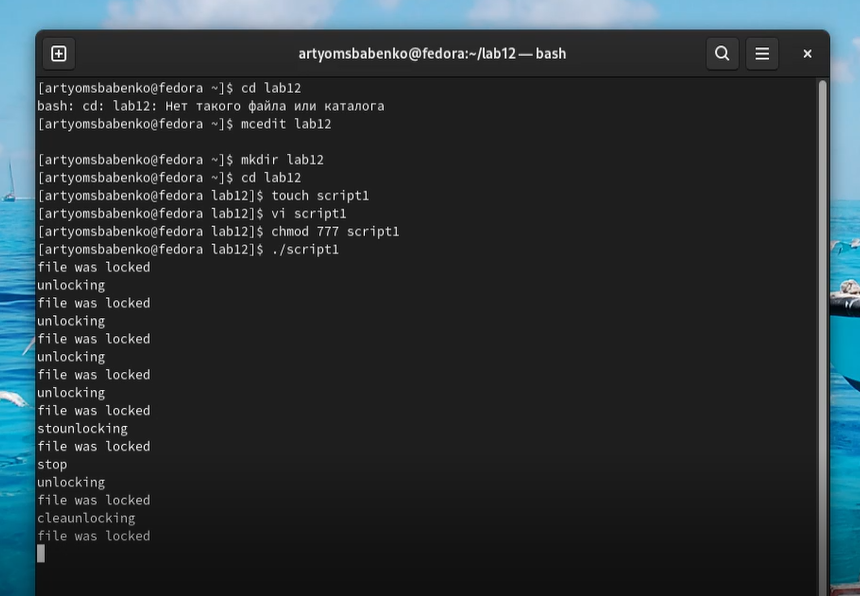
– BASH – сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек С и Корна (разработка компании Free Software Foundation).

# Ход работы

1. Написал 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, в котором также запущен этот файл, но не фоновом, а в привилегированном режиме (рис.1,2).

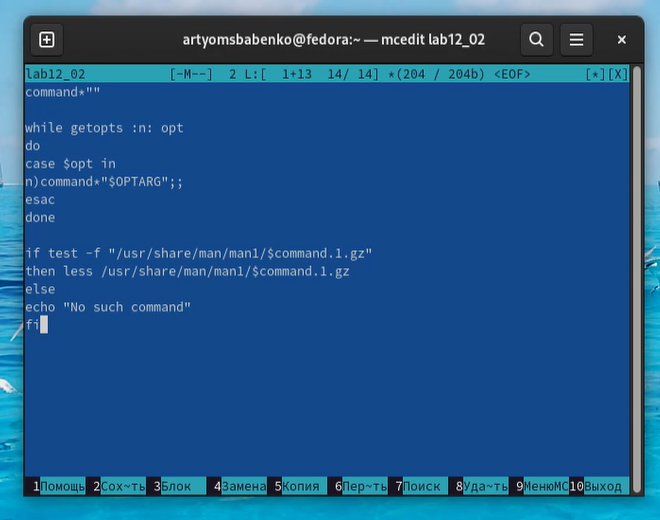


*Рис.1. Листинг № 1*

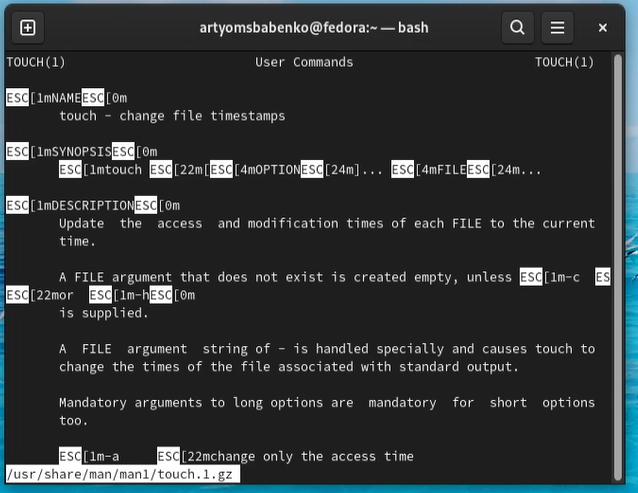


*Рис.2. Результат выполнения программы*

1. Реализовал команду man с помощью командного файла. Изучил содержимое каталога /usr/share/man/man1. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд. 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).

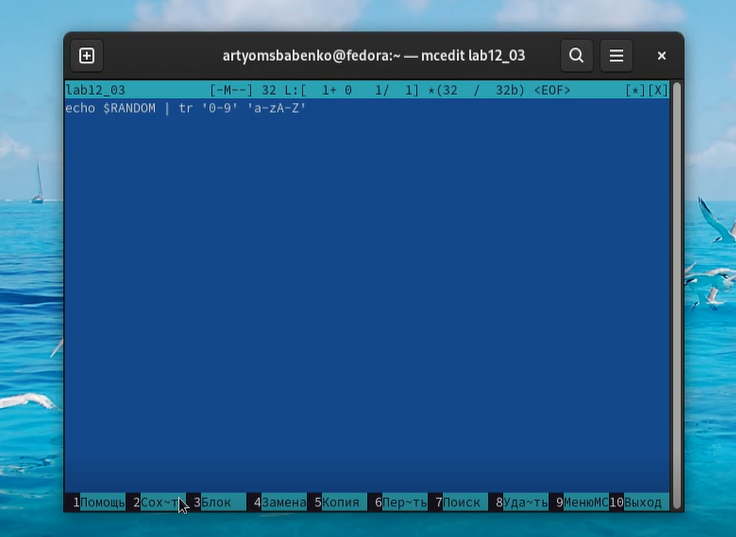


*Рис.3. Листинг № 2*

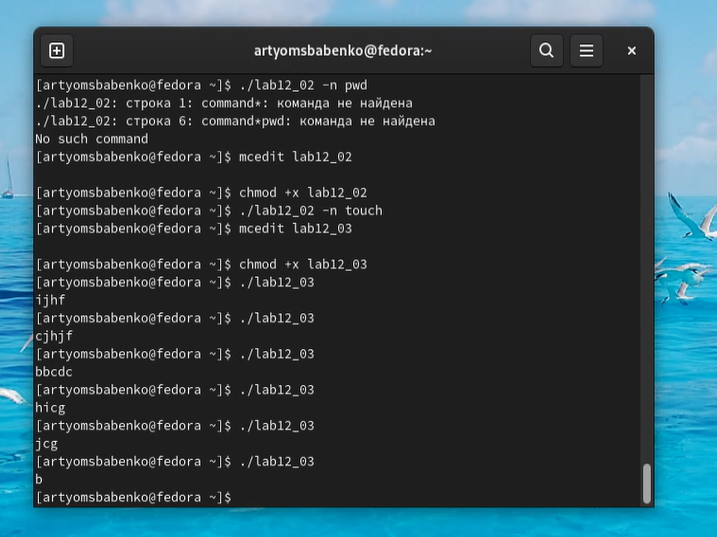


*Рис.4. Результат выполнения программы*

1. Используя встроенную переменную $RANDOM, написал командный файл, генерирующий случайную последовательность букв латинского алфавита (рис.3,4).



*Рис.5. Листинг № 3*



*Рис.6. Результат выполнения программы*

# Выводы

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

# Контрольные вопросы

1. Путём добавления одной строки в конец другой.
2. Команда seq в Linux используется для генерации чисел от первого до последнего шага increment. Это очень полезная команда, в которой нам пришлось генерировать список чисел в цикле while, for, before.
3. 3
4. У ZSH много функций, часть из них — лишь незначительные улучшения в Bash, но вот некоторые из основных:

автоматический cd: просто введите имя каталога;

рекурсивное расширение пути: например, «/u/lo/b» заменяется на «/usr/local/bin»;

исправление орфографии и приблизительное завершение: если вы допустили незначительную ошибку при вводе имени каталога, ZSH исправит её за вас;

поддержка плагинов и тем: ZSH включает множество различных фреймворков плагинов;

поддержка плагинов и тем, вероятно, самая крутая функция ZSH.