Архитектура компьютеров и операционные системы | Операционные системы

Лабораторная работа № 11. Программирование в командном процессоре ОС UNIX. Ветвления и циклы

Мугари Абдеррахим - НКАбд-03-22

Содержание

# 1 Цель работы

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

# 2 Выполнение лабораторной работы :

* Прежде всего, я написал программу, которая ищет шаблон в файле и отображает каждую строку, содержащую его, с указанием ее номера (рис. [1](#fig:001))

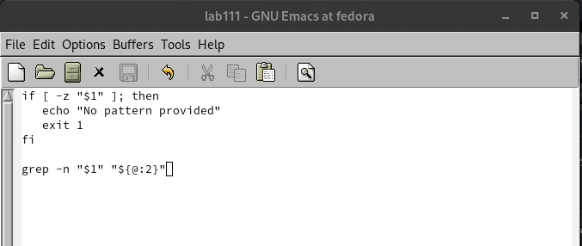


Figure 1: Написание первой программы

* после этого мне пришлось изменить права доступа к файлу, содержащему код, и добавить к нему выполнение, и после этого я запустил код, и он заработал (рис. [2](#fig:002))

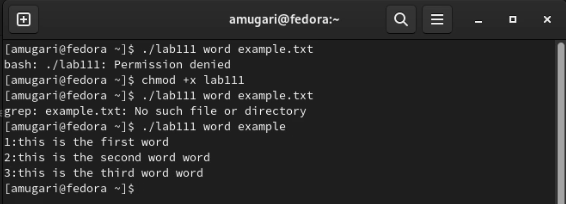


Figure 2: изменение прав доступа и выполнение программы

* затем здесь мне пришлось написать программу на C, которая вводит число и определяет, больше ли оно нуля, меньше нуля или равно нулю. Затем программа завершается с помощью функции exit(n), передавая информацию о коде завершения в

оболочку.(рис. [3](#fig:003))

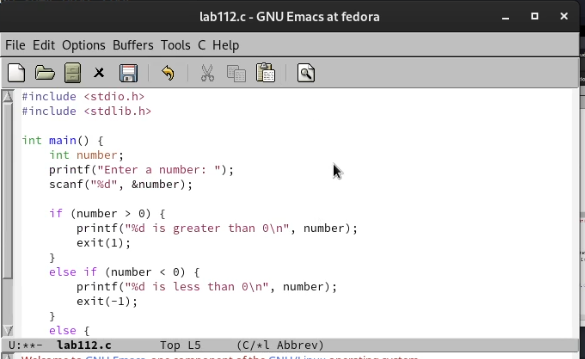


Figure 3: Написание второй программы

* после написания кода колледжа мне пришлось скомпилировать его перед выполнением (рис. [4](#fig:004))

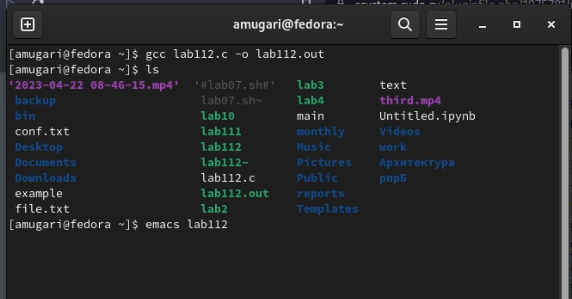


Figure 4: компиляция кода, написанного на C

* затем я написал командный файл, который должен вызвать эту программу и, проанализировав ее с помощью команды **$?**, выдать сообщение о том, какое число было введено. (рис. [5](#fig:005))

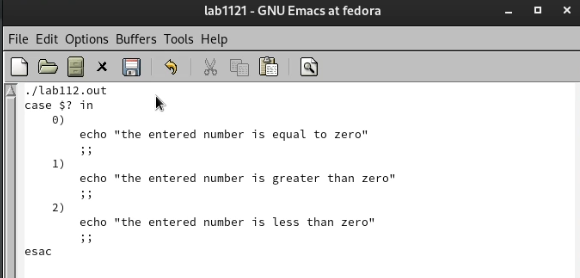


Figure 5: Написание командного файла

* здесь я изменил право доступа к командному файлу, а затем выполнил его для вызова другой программы, написанной на C, где я протестировал три возможные ситуации, и это сработало (рис. [6](#fig:006))

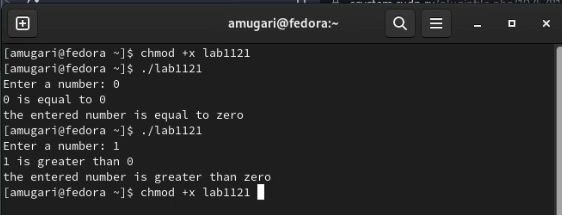


Figure 6: изменение прав доступа и выполнение программы

* затем я написал код, который создает указанное количество файлов, пронумерованных последовательно от 1 до N. Количество файлов , которые должны быть созданы, передается в качестве аргументов командной строки. Один и тот же командный

файл должен иметь возможность удалять все созданные им файлы (если они существуют). (рис. [7](#fig:007))

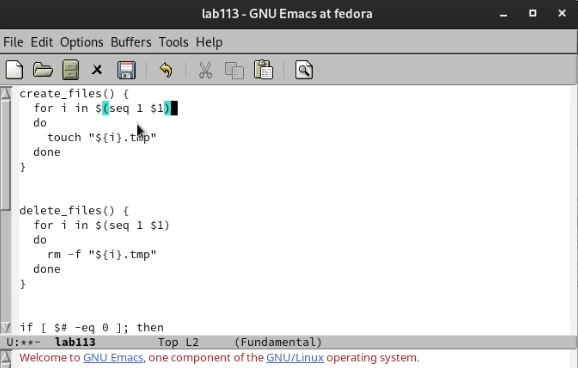


Figure 7: написание третьего программы

* затем мне пришлось изменить права доступа к коду и выполнить программу, и все работало нормально (рис. [8](#fig:008))

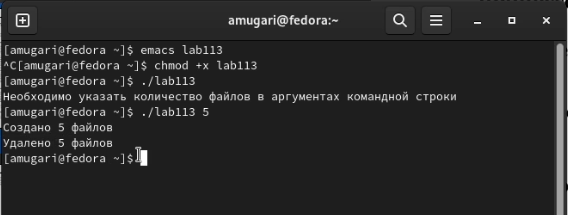


Figure 8: изменение прав доступа и выполнение программы

* здесь я написал код, который использует команду **tar** для архивирования всех файлов в указанном каталоге. Меняем его таким образом, чтобы упаковывались только те файлы, которые были изменены менее недели назад (с помощью команды

find) (рис. [9](#fig:009))

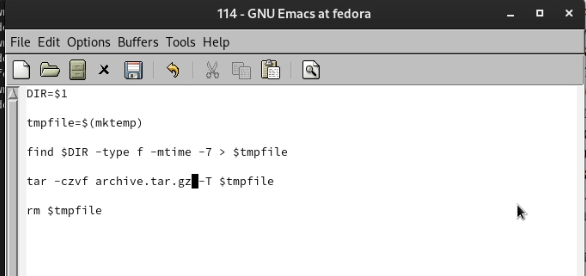


Figure 9: написание четвертой программы

* наконец, я изменил права доступа и выполнил файл, и все заработало, как и должно быть (рис. [10](#fig:010))

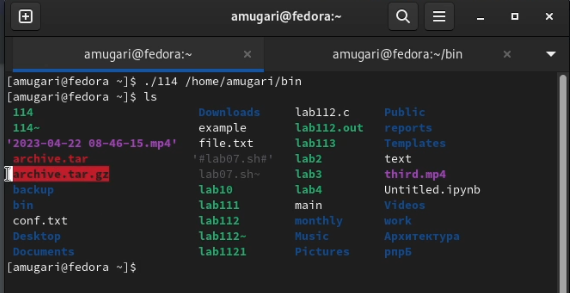


Figure 10: изменение прав доступа и выполнение программы

## 2.1 Контрольные вопросы:

1. Команда getopts в UNIX-подобных операционных системах используется для анализа аргументов командной строки, переданных в скрипт. Она обрабатывает короткие опции, заданные после символа “-” и длинные опции, заданные после символа “–”.
2. Эта команда позволяет программисту легко определять и обрабатывать опции и аргументы, переданные в командной строке. Метасимволы в UNIX используются для генерации имен файлов и для манипуляции с файловой системой. Например, символ звездочки (\*) используется для обозначения любого количества любых символов в имени файла, а символ вопросительного знака (?) используется для обозначения любого одного символа в имени файла.
3. Операторы управления действиями в UNIX-подобных системах включают в себя операторы условного выполнения (if, case), операторы циклов (for, while, until) и операторы перенаправления ввода-вывода (>, >>, <, <<).
4. Для прерывания цикла в UNIX используются операторы break и continue. Оператор break прерывает выполнение цикла и переходит к следующей инструкции после цикла, а оператор continue прерывает текущую итерацию цикла и переходит к следующей итерации.
5. Команда false возвращает ненулевое значение и используется для проверки скриптов на ошибки, а команда true возвращает нулевое значение и используется для явной инициализации переменных и для создания бесконечных циклов.
6. Данная строка проверяет наличие файла в директории man$s/ с именем, состоящим из переменной $i и переменной $s, и если файл существует, то скрипт продолжает выполнение.
7. Конструкция while выполняет цикл, пока условие истинно, а конструкция until выполняет цикл, пока условие ложно. То есть, пока условие в while истинно, цикл будет выполняться, а в until, пока условие ложно. Также, в while цикл будет выполняться, если условие истинно с самого начала, а в until - если условие ложно.

## 2.2 выводы по результатам выполнения заданий:

* В ходе этой лабораторной работы у меня была возможность научиться программировать в операционной системе UNIX и приобрести практические навыки написания сложного кода с использованием логических управляющих структур и циклов.

# 3 Выводы, согласованные с целью работы:

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