

Лабораторная работа №10

**Программирование в командном процессоре ОС UNIX. Расширенное
программирование**

Мальсагов Акрамат Абу-Бакарович

Содержание

1	Цель работы	5
2	Выполнение лабораторной работы	6
3	Выводы	10
4	Контрольные вопросы	11

Список иллюстраций

2.1	Код 1 скрипта	6
2.2	Результат работы 1 скрипта	7
2.3	Код 2 скрипта	8
2.4	Работа скрипта	8
2.5	Код 3 скрипта	9
2.6	Запуск скрипта	9

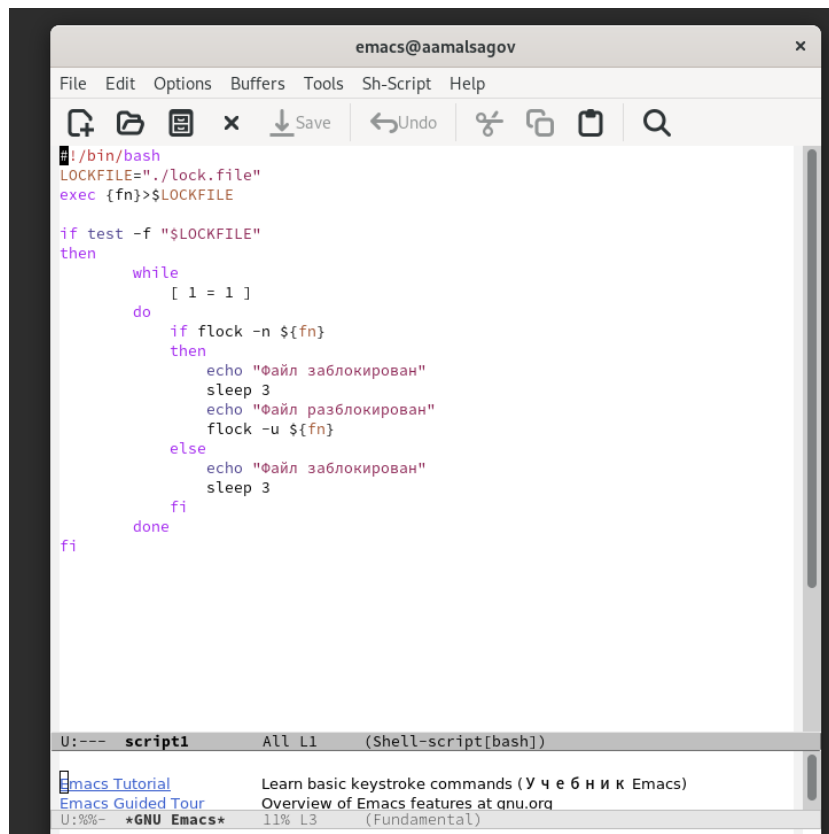
Список таблиц

1 Цель работы

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

2 Выполнение лабораторной работы

1. Создал script1 и открыл его в emacs. Написал программу, реализующая упрощённый механизм семафоров. Командный файл в течение некоторого времени t1 дожидается освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использует его в течение некоторого времени t2<>t1, также выдавая информацию о том, что ресурс используется соответствующим командным файлом.(рис. 2.1)

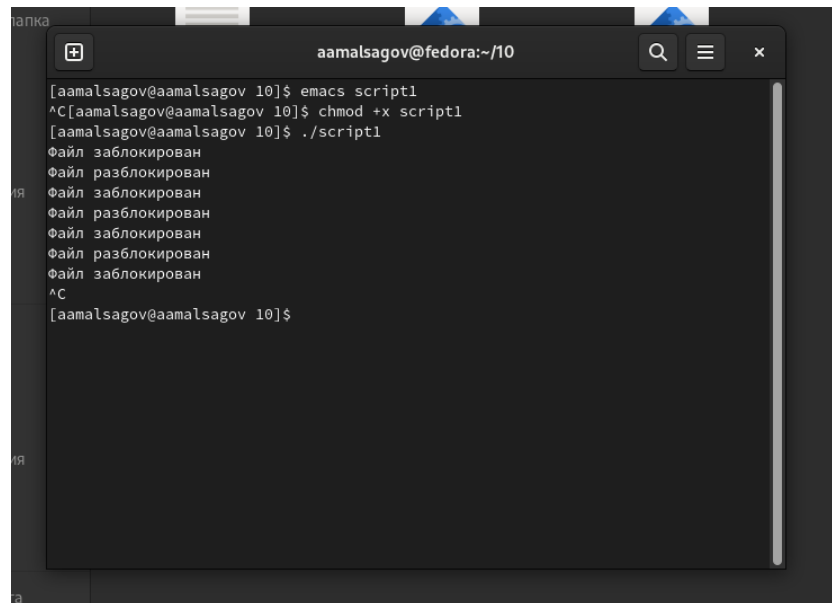


```
#!/bin/bash
LOCKFILE="./lock.file"
exec {fn}>$LOCKFILE

if test -f "$LOCKFILE"
then
  while
    [ 1 = 1 ]
  do
    if flock -n ${fn}
    then
      echo "Файл заблокирован"
      sleep 3
      echo "Файл разблокирован"
      flock -u ${fn}
    else
      echo "Файл заблокирован"
      sleep 3
    fi
  done
fi
```

Рис. 2.1: Код 1 скрипта

2. Проверил его работу. (рис. 2.2)

A terminal window titled 'aamalsagov@fedora:~/10' with search, menu, and close icons. The terminal shows the following commands and output:

```
[aamalsagov@aamalsagov 10]$ emacs script1
^C[aamalsagov@aamalsagov 10]$ chmod +x script1
[aamalsagov@aamalsagov 10]$ ./script1
Файл заблокирован
Файл разблокирован
Файл заблокирован
Файл разблокирован
Файл заблокирован
Файл разблокирован
Файл заблокирован
^C
[aamalsagov@aamalsagov 10]$
```

Рис. 2.2: Результат работы 1 скрипта

3. Написал командный файл, который получает в виде аргумента командной строки название команды и в виде результата выдает справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге man1. (рис. 2.3)

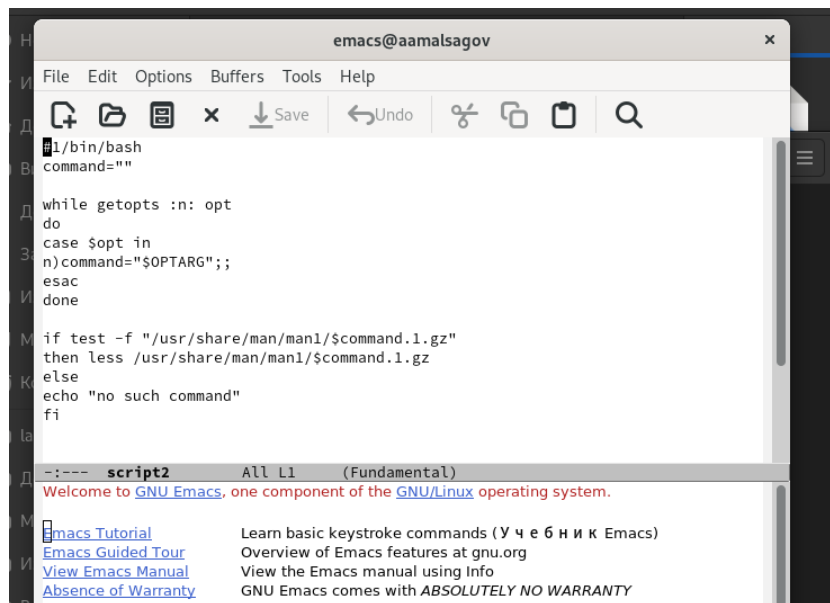


Рис. 2.3: Код 2 скрипта

4. Запустил скрипт.(рис. 2.4)

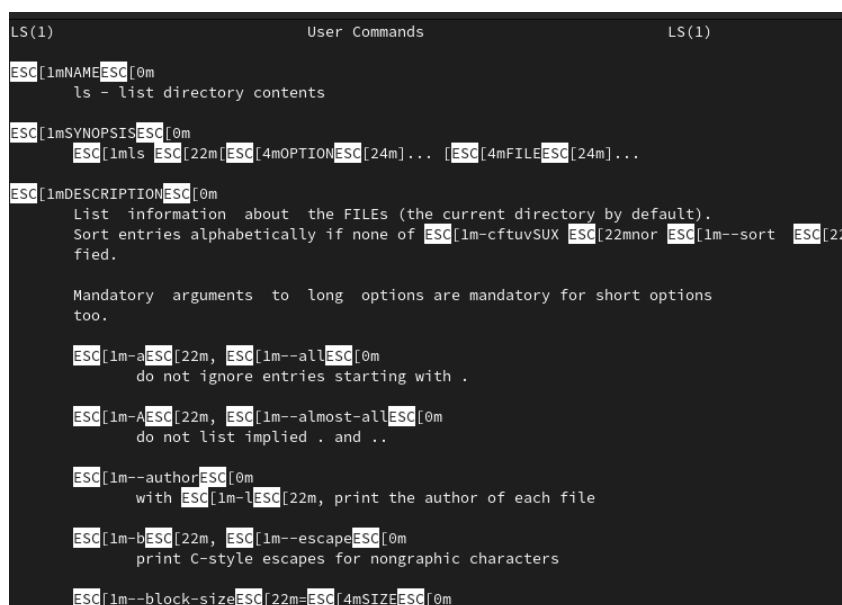


Рис. 2.4: Работа скрипта

5. Используя встроенную переменную \$RANDOM, написал командный файл, генерирующий случайную последовательность букв латинского алфавита.(рис. 2.5)

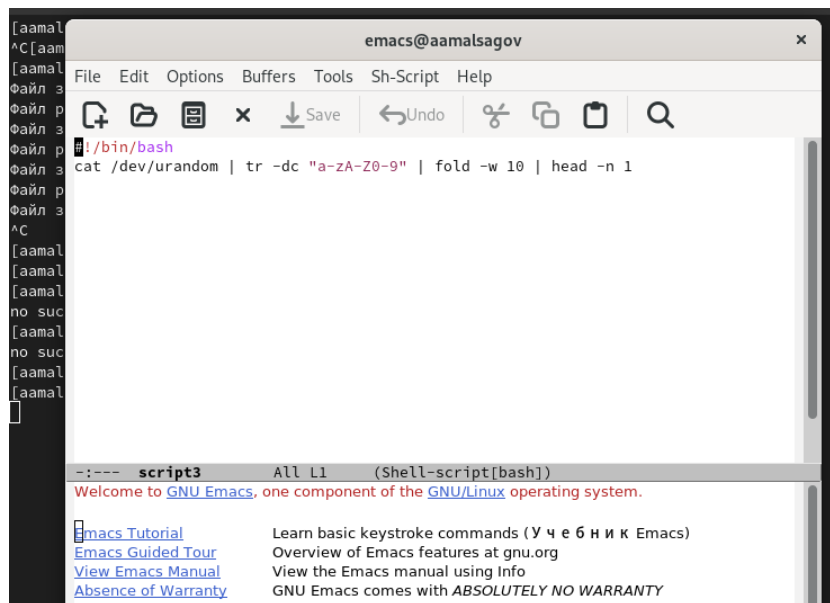


Рис. 2.5: Код 3 скрипта

6. Проверил работу скрипта. (рис. 2.6)

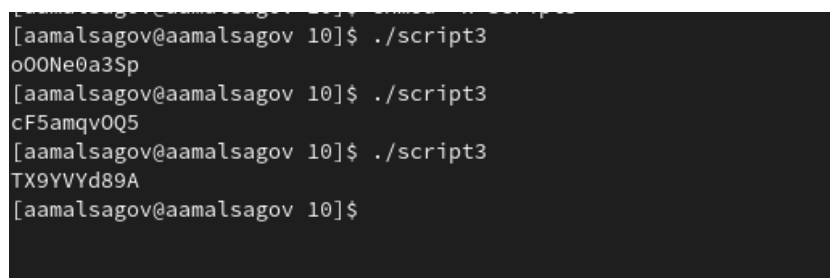


Рис. 2.6: Запуск скрипта

3 Выводы

Мы научились писать более сложные команды файлы.

4 Контрольные вопросы

1: Найдите синтаксическую ошибку в следующей строке: `while [$1 != "exit"]`

\$1.

Так же между скобками должны быть пробелы. В противном случае скобки и рядом стоящие символы будут восприниматься как одно целое

2: Как объединить (конкатенация) несколько строк в одну?

```
cat file.txt | xargs | sed -e 's/\. /\n/g'
```

3: Найдите информацию об утилите `seq`. Какими иными способами можно реализовать её функционал при программировании на `bash`?

`seq` - выдает последовательность чисел. Реализовать ее функционал можно командой `for n in {1..5} do done`

4: Какой результат даст вычисление выражения `$((10/3))`?

3

5: Укажите кратко основные отличия командной оболочки `zsh` от `bash`.

`Zsh` очень сильно упрощает работу. Но существуют различия. Например, в `zsh` после `for` обязательно вставлять пробел, нумерация массивов в `zsh` начинается с 1 (что не особо удобно на самом деле). Если вы собираетесь писать скрипт, который легко будет запускать множество разработчиков, то я рекомендую `Bash`. Если скрипты вам не нужны - `Zsh` (более простая работа с файлами, например)

6: Проверьте, верен ли синтаксис данной конструкции `for ((a=1; a <= LIMIT; a++))`

Верен

7: Сравните язык `bash` с какими-либо языками программирования. Какие преимущества у `bash` по сравнению с ними? Какие недостатки?

Bash позволяет очень легко работать с файловой системой без лишних конструкций (в отличие от обычного языка программирования). Но относительно обычных языков программирования `bash` очень сжат. Тот же Си имеет гораздо более широкие возможности для разработчика.