

Лабораторная работа №12

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

Чемоданова Ангелина Александровна

Содержание

1	Цель работы	4
2	Задание	5
3	Теоретическое введение	7
4	Выполнение лабораторной работы	8
5	Выводы	13
6	Контрольные вопросы	14

Список иллюстраций

4.1	Скрипт 1	9
4.2	Выполнение скрипта 1	9
4.3	Скрипт 2	10
4.4	Выполнение скрипта 2	10
4.5	Результат выполнения скрипта 2	11
4.6	Скрипт 3	11
4.7	Выполнение скрипта 3	12

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. Командный файл должен в течение некоторого времени t_1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени $t_2 < t_1$, 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 команде 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.

3 Теоретическое введение

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: – оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций; – C-оболочка (или csh) — надстройка на оболочкой Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд; – оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна; – BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation). POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ.

4 Выполнение лабораторной работы

1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t_1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени $t_2 < t_1$, 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.

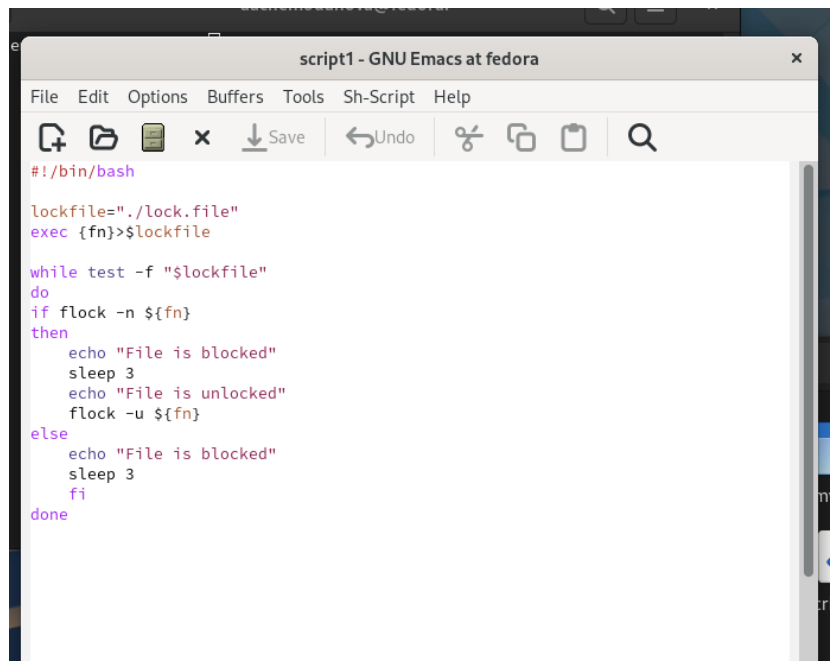


Рис. 4.1: Скрипт 1

Выполнение скрипта 1(рис. 4.2).

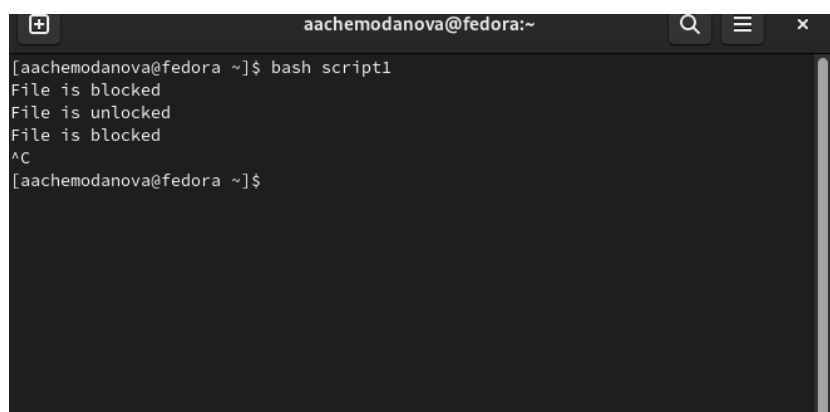
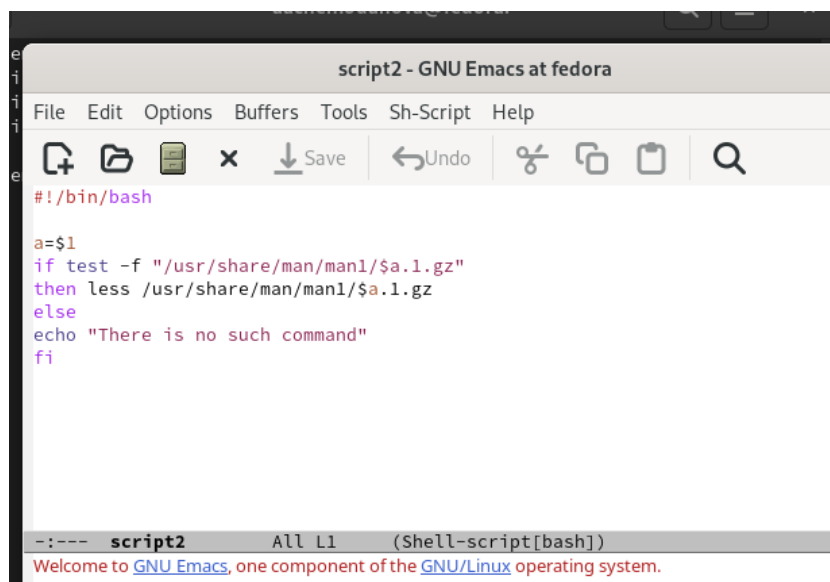


Рис. 4.2: Выполнение скрипта 1

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 команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге man1.

Скрипт 2 (рис. 4.3).

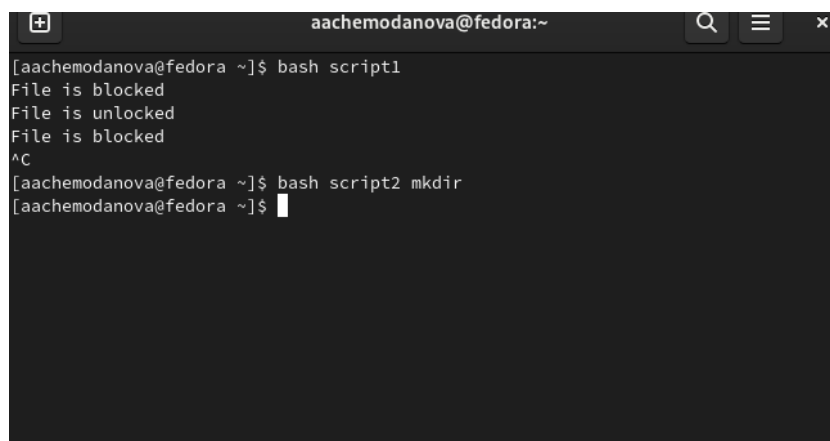


```
#!/bin/bash

a=$1
if test -f "/usr/share/man/man1/$a.1.gz"
then less /usr/share/man/man1/$a.1.gz
else
echo "There is no such command"
fi
```

Рис. 4.3: Скрипт 2

Выполнение скрипта 2 (рис. 4.4).



```
[aachemodanova@fedora ~]$ bash script1
File is blocked
File is unlocked
File is blocked
^C
[aachemodanova@fedora ~]$ bash script2 mkdir
[aachemodanova@fedora ~]$
```

Рис. 4.4: Выполнение скрипта 2

Результат выполнение скрипта 2(рис. 4.5).

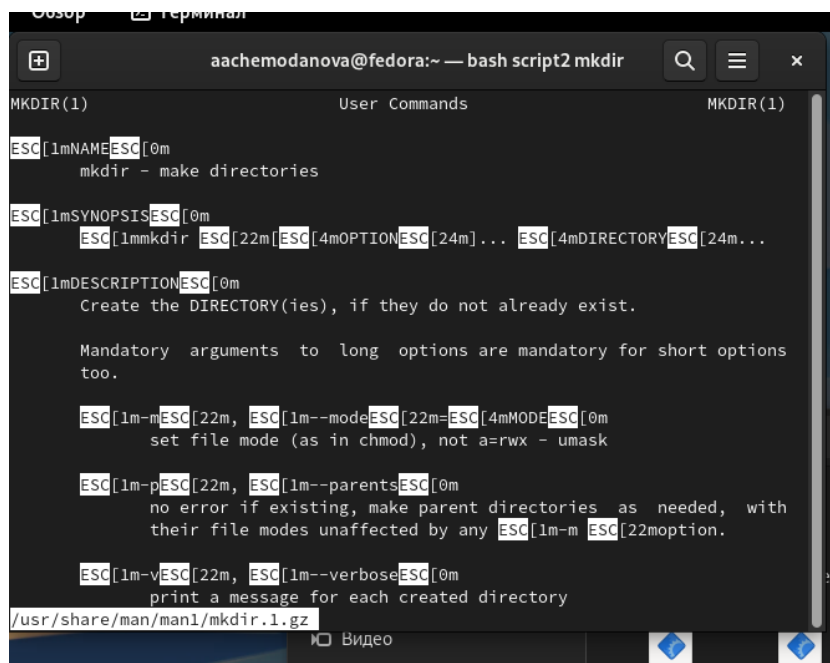


Рис. 4.5: Результат выполнение скрипта 2

- 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.

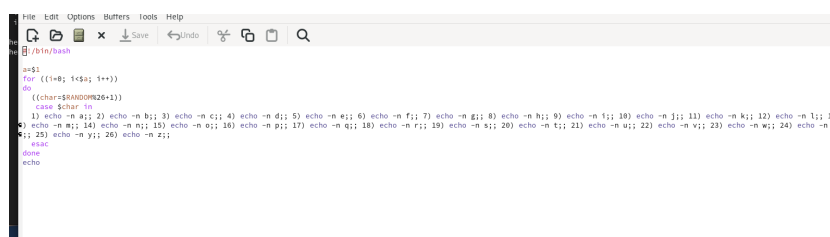
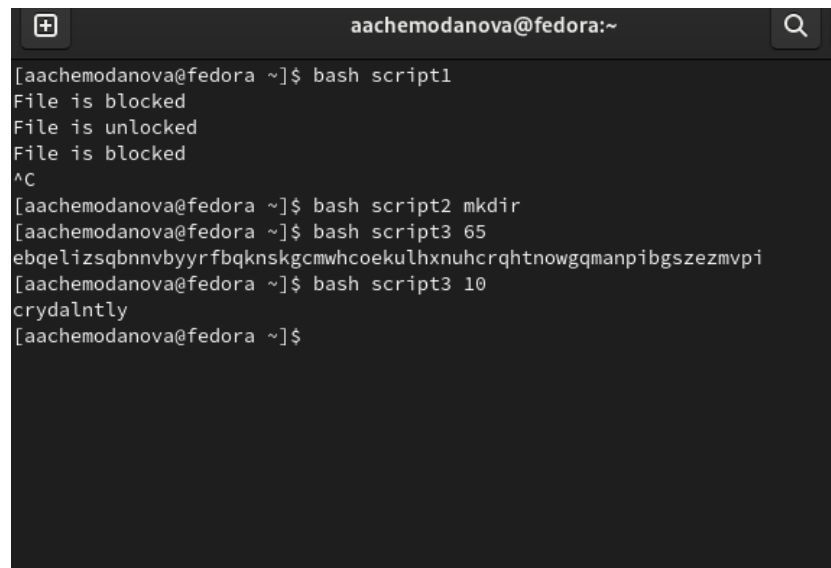


Рис. 4.6: Скрипт 3

Выполнение скрипта 3(рис. 4.7).

A terminal window titled 'aachemodanova@fedora:~' with a search icon in the top right. The terminal shows the following commands and output:

```
[aachemodanova@fedora ~]$ bash script1
File is blocked
File is unlocked
File is blocked
^C
[aachemodanova@fedora ~]$ bash script2 mkdir
[aachemodanova@fedora ~]$ bash script3 65
ebqelizsqbnnvbyrfbqknskkgcmwhcoekulhxnuhcrqhtnowgqmanpibgszezmvpi
[aachemodanova@fedora ~]$ bash script3 10
cryda!ntly
[aachemodanova@fedora ~]$
```

Рис. 4.7: Выполнение скрипта 3

5 Выводы

Мы изучили основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

6 Контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке: `while [$1 != "exit"]`
Между выражением и квадратными скобками должны быть пробелы.
2. Как объединить (конкатенация) несколько строк в одну? С помощью `cat` и `|`.
3. Найдите информацию об утилите `seq`. Какими иными способами можно реализовать её функционал при программировании на `bash`? `seq` - утилита, способная сгенерировать последовательность чисел. Реализовать эту же функцию можно с помощью цикла `for`.
4. Какой результат даст вычисление выражения `$((10/3))`? Результат: 3
5. Укажите кратко основные отличия командной оболочки `zsh` от `bash`. Оболочка `zsh` больше подходит для, например, работы с файлами, так как она сильно упрощает работу. Она имеет в некоторых местах отличающийся синтаксис с обязательными правилами (например, пробел перед `for`). Если нужно написать скрипт, эффективно работающий от множества пользователей, то лучше использовать `bash`.
6. Проверьте, верен ли синтаксис данной конструкции `for ((a=1; a <= LIMIT; a++))` Да
7. Сравните язык `bash` с какими-либо языками программирования. Какие преимущества у `bash` по сравнению с ними? Какие недостатки? `Bash` позволяет работать с файловой системой без лишних конструкций, однако его возможности не так велики, как у остальных языков программирования.