

# **Лабораторная работа №12**

**Операционные системы**

Ищенко Ирина Олеговна

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>5</b>
<b>2</b>	<b>Задание</b>	<b>6</b>
<b>3</b>	<b>Теоретическое введение</b>	<b>7</b>
<b>4</b>	<b>Выполнение лабораторной работы</b>	<b>8</b>
<b>5</b>	<b>Вывод</b>	<b>13</b>
<b>6</b>	<b>Выполнение заданий самостоятельной работы</b>	<b>14</b>

## Список иллюстраций

4.1	Скрипт к заданию 1. . . . .	8
4.2	Результат выполнения скрипта 1. . . . .	9
4.3	Просмотр каталога /usr/share/man/man1. . . . .	9
4.4	Скрипт к заданию 2. . . . .	10
4.5	Выполнения скрипта 2. . . . .	10
4.6	Результат выполнения скрипта 2. . . . .	11
4.7	Скрипт к заданию 3. . . . .	12
4.8	Результат выполнения скрипта 3. . . . .	12

## 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 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, 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.

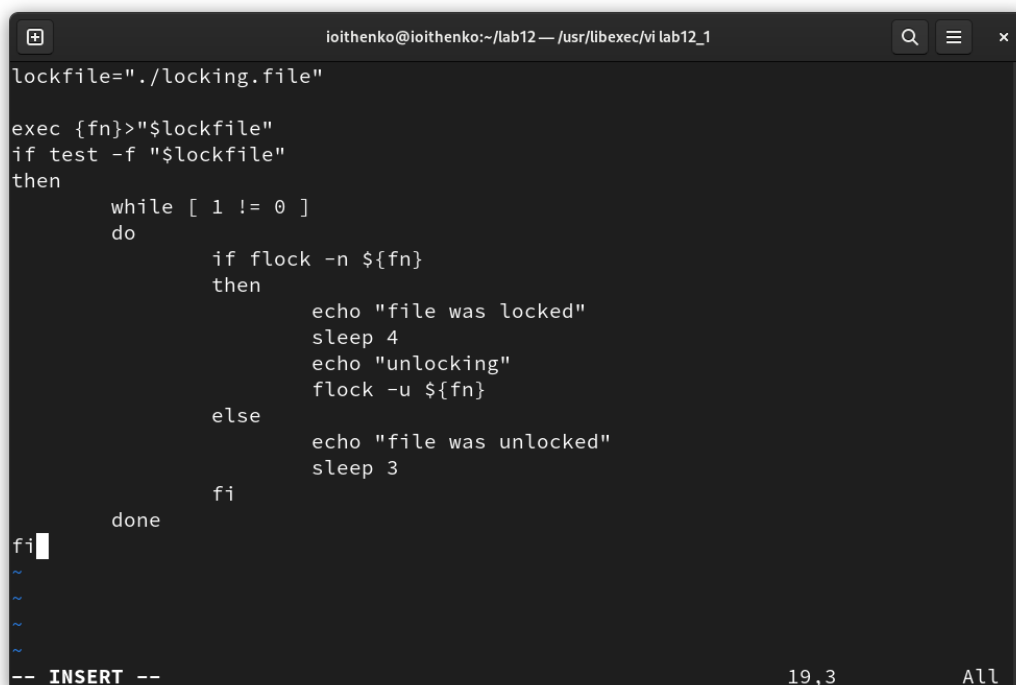
### 3 Теоретическое введение

Преимущества и недостатки Bash:

Многие языки программирования намного удобнее и понятнее для пользователя. Например, Python более быстр, так как компилируется байтами. Однако главное преимущество Bash – его повсеместное распространение. Более того, Bash позволяет очень легко работать с файловой системой без лишних конструкций (в отличие от других языков программирования). Но относительно таких bash очень сжат. То есть, например, C имеет гораздо более широкие возможности для разработчика.

## 4 Выполнение лабораторной работы

1. Я написала командный файл, реализующий упрощенный механизм семафоров (рис. [4.1]).



```
lockfile="./locking.file"

exec {fn}>"$lockfile"
if test -f "$lockfile"
then
    while [ 1 != 0 ]
    do
        if flock -n ${fn}
        then
            echo "file was locked"
            sleep 4
            echo "unlocking"
            flock -u ${fn}

        else
            echo "file was unlocked"
            sleep 3
        fi
    done
fi
```

The screenshot shows a terminal window with a dark background. The title bar indicates the user is 'ioithenko' and the current directory is '/lab12'. The script content is as follows:   
lockfile="./locking.file"  
  
exec {fn}>"\$lockfile"  
if test -f "\$lockfile"  
then  
 while [ 1 != 0 ]  
 do  
 if flock -n \${fn}  
 then  
 echo "file was locked"  
 sleep 4  
 echo "unlocking"  
 flock -u \${fn}  
  
 else  
 echo "file was unlocked"  
 sleep 3  
 fi  
 done  
fi  
The cursor is positioned at the end of the last line 'fi'. At the bottom of the terminal, it shows '-- INSERT --' on the left, '19,3' in the center, and 'All' on the right.

Рис. 4.1: Скрипт к заданию 1.

Затем я добавила право на исполнение файла и выполнила его (рис. [4.2]).



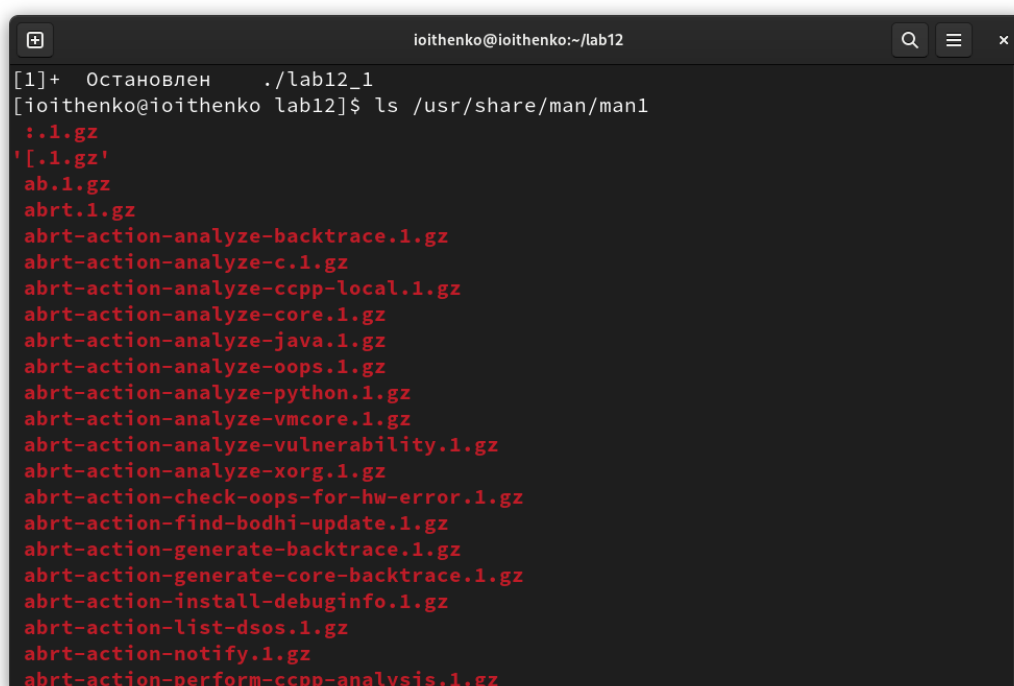
```

[ioithenko@ioithenko lab12]$ vi lab12_1
[ioithenko@ioithenko lab12]$ chmod +x lab12_1
[ioithenko@ioithenko lab12]$ ./lab12_1
file was locked
unlocking
file was locked
unlocking
file was locked

```

Рис. 4.2: Результат выполнения скрипта 1.

2. Я просмотрела содержимое каталога /usr/share/man/man1 (рис. [4.3]).



```

ioithenko@ioithenko:~/lab12
[1]+  Остановлен ./lab12_1
[ioithenko@ioithenko lab12]$ ls /usr/share/man/man1
.:1.gz
'[:1.gz'
abrt.1.gz
abrt.1.gz
abrt-action-analyze-backtrace.1.gz
abrt-action-analyze-c.1.gz
abrt-action-analyze-ccpp-local.1.gz
abrt-action-analyze-core.1.gz
abrt-action-analyze-java.1.gz
abrt-action-analyze-oops.1.gz
abrt-action-analyze-python.1.gz
abrt-action-analyze-vmcore.1.gz
abrt-action-analyze-vulnerability.1.gz
abrt-action-analyze-xorg.1.gz
abrt-action-check-oops-for-hw-error.1.gz
abrt-action-find-bodhi-update.1.gz
abrt-action-generate-backtrace.1.gz
abrt-action-generate-core-backtrace.1.gz
abrt-action-install-debuginfo.1.gz
abrt-action-list-dsos.1.gz
abrt-action-notify.1.gz
abrt-action-perform-ccpp-analysis.1.gz

```

Рис. 4.3: Просмотр каталога /usr/share/man/man1.

Я написала командный файл, позволяющий реализовать команду man с помощью команды less, которая выдает содержимое справки по команде (рис. [4.4]).



```
ioithenko@ioithenko:~/lab12 — bash
PWD(1) User Commands PWD(1)
ESC[1mNAMEESC[0m
    pwd - print name of current/working directory
ESC[1mSYNOPSISESC[0m
    ESC[1mpwd ESC[22m[ESC[4mOPTIONESC[24m]...
ESC[1mDESCRIPTIONESC[0m
    Print the full filename of the current working directory.
    ESC[1m-LESC[22m, ESC[1m--logicalESC[0m
        use PWD from environment, even if it contains symlinks
    ESC[1m-PESC[22m, ESC[1m--physicalESC[0m
        avoid all symlinks
    ESC[1m--help ESC[22mdisplay this help and exit
    ESC[1m--versionESC[0m
        output version information and exit
    If no option is specified, ESC[1m-P ESC[22mis assumed.
/usr/share/man/man1/pwd.1.gz
```

Рис. 4.6: Результат выполнения скрипта 2.

3. Я написала командный файл, который генерировал случайную последовательность букв латинского алфавита (рис. [4.7]).



## 5 Вывод

В ходе выполнения 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.

## 6 Выполнение заданий самостоятельной работы

### Контрольные вопросы

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

Между скобками должны быть пробелы, иначе символы в скобках и сами скобки будут восприняты как один элемент.

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

```
cat file.txt | xargs
```

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

Команда `seq` выводит последовательность целых или действительных чисел, подходящую для передачи в другие программы. Реализовать ее функционал можно командой `for n in {1..5} do done`

4. Какой результат даст вычисление выражения  $\$((10/3))$ ? Вычисление этого выражения даст результат 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. Если вы собираетесь писать скрипт, который будет запускать множество разработчиков, то рекомендуется `Bash`. Если скрипты вам не нужны - `Zsh`.

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

Да, этот синтаксис верен.

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

Многие языки программирования намного удобнее и понятнее для пользователя. Например, Python более быстр, так как компилируется байтами. Однако главное преимущество Bash – его повсеместное распространение. Более того, Bash позволяет очень легко работать с файловой системой без лишних конструкций (в отличие от других языков программирования). Но относительно таких `bash` очень сжат. То есть, например, C имеет гораздо более широкие возможности для разработчика.