

# МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РОССИЙСКОЙ ФЕДЕРАЦИИ

---

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ  
ВЫСШЕГО ОБРАЗОВАНИЯ «РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ»

Факультет физико-математических и естественных наук Кафедра информационных технологий

## ОТЧЕТ

---

по лабораторной работе 12

**ТЕМА** «Программирование в командном процессоре ОС UNIX. Расширенное  
программирование» по дисциплине «Операционные системы»

## Выполнил/ла:

---

Студент/ка группы НПИбд-02-21

Студенческий билет No 1032205421

Студент/ка: Стелина Петрити

## Цель работы

---

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

## Последовательность выполнения 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.

```
emacs@fedora
File Edit Options Buffers Tools Sh-Script Help
[Icons: New, Open, Save, Close, Save As, Undo, Cut, Copy, Paste]

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

if test -f "$lockfile"
then
    while [ 1 = 1 ]
    do
        if flock -n ${fn}
        then
            echo "File locked"
            sleep 4
            echo "File unlocked"
            flock -u ${fn}
        else
            echo "File already locked"
            sleep 4
        fi
    done
else
    echo "File does not exists"
fi
```

~:--- work12.sh All L1 (Shell-script[bash])

### 1.1.code work12.sh

```
[inna@fedora ~]$ emacs work12.sh
[inna@fedora ~]$ chmod +x work12.sh
[inna@fedora ~]$ ./work12.sh
File locked
File unlocked
File locked
File unlocked
File locked
File unlocked
File locked
File unlocked
File locked
File unlocked
File locked
File unlocked
File locked
File unlocked
File locked
File unlocked
File locked
File unlocked
```

### 1.2.1.проверка

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

```
#!/bin/bash

name=""
while getopts "n:" opt
do
    case $opt in
        n) name=$OPTARG;;
    esac
done
if test -f "/usr/share/man/man1/$name.1.gz"
then
    less /usr/share/man/man1/"$name".1.gz
else
    echo "Not found"
fi
```

### 2.1.code work2.sh

```
[inna@fedora ~]$ emacs work2.sh
[inna@fedora ~]$ chmod +x work2.sh
[inna@fedora ~]$ work2.sh
bash: work2.sh: command not found...
[inna@fedora ~]$ chmod +x work2.sh
[inna@fedora ~]$ ./work2.sh
Not found
[inna@fedora ~]$ ./work2.sh -n mkdir
Not found
[inna@fedora ~]$
```

### 2.2.проверка

```
MKDIR(1)                                User Commands

ESC[1mNAMEESC[0m
    mkdir - make directories

ESC[1mSYNOPSISESC[0m
    ESC[1mmkdir ESC[22m[ESC[4mOPTIONESC[24m]... ESC[4mDIREC

ESC[1mDESCRIPTIONESC[0m
    Create the DIRECTORY(ies), if they do not already exist

    Mandatory arguments to long options are mandatory f
    too.

    ESC[1m-mESC[22m, ESC[1m--modeESC[22m=ESC[4mMODEESC[0m
        set file mode (as in chmod), not a=rwx - umask

    ESC[1m-pESC[22m, ESC[1m--parentsESC[0m
```

### 2.3.Результат

3.Используя встроенную переменную \$RANDOM, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита. Учтите, что \$RANDOMвыдаёт псевдослучайные числа в диапазоне от 0 до 32767.

```
File Edit Options Buffers Tools Sh-Script Help
[Icons: New, Open, Save, Undo, Redo, Find, etc.]
#!/bin/bash
echo $RANDOM |tr '[0-32767]' '[a-z]'

U:*** work3.sh All L2 (Shell-script[sh])
```

#### 3.1.code work3.sh

```
[inna@fedora ~]$ emacs work3.sh
[inna@fedora ~]$ chmod +x work3.sh
[inna@fedora ~]$ ./work3.sh
bh95d
[inna@fedora ~]$ ./work3.sh
b9e5g
[inna@fedora ~]$ ./work3.sh
b4ddh
[inna@fedora ~]$ ./work3.sh
ed85h
[inna@fedora ~]$
```

### 3.2.проверка

## Выводы

---

Я изучила написать более сложные командные файлы, и использовать логические управляющие конструкции и циклы.

## Контрольные вопросы

---

**1.Найдите синтаксическую ошибку в следующей строке:**

```
while |$1 != "exit" | -> while | $1 != "exit" |
```

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

записать их одну за другой:

```
var="Hello,"
```

```
var1=" World"
```

```
var2="$VAR$VAR1"
```

```
echo "$VAR2"
```

строка в bash - это добавление переменных или буквальных строк к переменной с помощью оператора+=

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

Эти утилиты выводят последовательность целых чисел с шагом, заданным пользователем

```
bash$ seq 3
```

```
1
```

```
2
```

```
3
```

4.

**4.Какой результат даст вычисление выражения  $\$(10/3)$ ?**

3

**5.Укажите кратко основные отличия командной оболочки zsh от bash**

Shell -командная строка более гибкая. Можно отобразить команду слева, а другую справа, как в разделённом экране vim. Авто дополнение также быстрее и более изменяемое, чем в Bash.

**6.Проверьте, верен ли синтаксис данной конструкции**

1 for ((a=1; a <= LIMIT; a++)) -синтаксис верен

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

Python — популярная альтернатива Bash для написания сценариев настройки среды, сборки и выпуска.

Bash — это командный язык, а не язык программирования общего назначения. Bash все и всегда воспринимает как команду, потому что это командный язык.

Python можно писать современные сложные а Shell- сценарии.

Python изначально не поддерживает выполнение процесса.