

# **Отчёт по лабораторной работе № 12**

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

Азимов Миразим

# Содержание

<b>1</b>	<b>Цель работы</b>	<b>4</b>
<b>2</b>	<b>Задание</b>	<b>5</b>
<b>3</b>	<b>Выполнение лабораторной работы</b>	<b>6</b>
3.1	Написание программ . . . . .	6
<b>4</b>	<b>Выводы</b>	<b>14</b>
<b>5</b>	<b>Ответы на контрольные вопросы</b>	<b>15</b>

## Список иллюстраций

3.1	Скрипт 1 . . . . .	6
3.2	Скрипт 1 . . . . .	7
3.3	Скрипт 1 . . . . .	7
3.4	Скрипт 1 . . . . .	8
3.5	Скрипт 1, доработка . . . . .	8
3.6	Скрипт 1, доработка . . . . .	9
3.7	/usr/share/man/man1 . . . . .	10
3.8	Скрипт 2 . . . . .	10
3.9	Скрипт 2 . . . . .	10
3.10	Скрипт 2 . . . . .	11
3.11	Скрипт 2 . . . . .	11
3.12	Скрипт 2 . . . . .	11
3.13	Скрипт 3 . . . . .	12
3.14	Скрипт 3 . . . . .	12
3.15	Скрипт 3 . . . . .	13
3.16	Скрипт 3 . . . . .	13
3.17	Скрипт 3 . . . . .	13

# 1 Цель работы

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

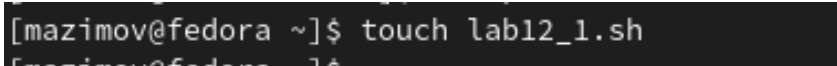
## 2 Задание

- Ознакомиться с теоретическим материалом.
- Выполнить упражнения.
- Ответить на контрольные вопросы.

## 3 Выполнение лабораторной работы

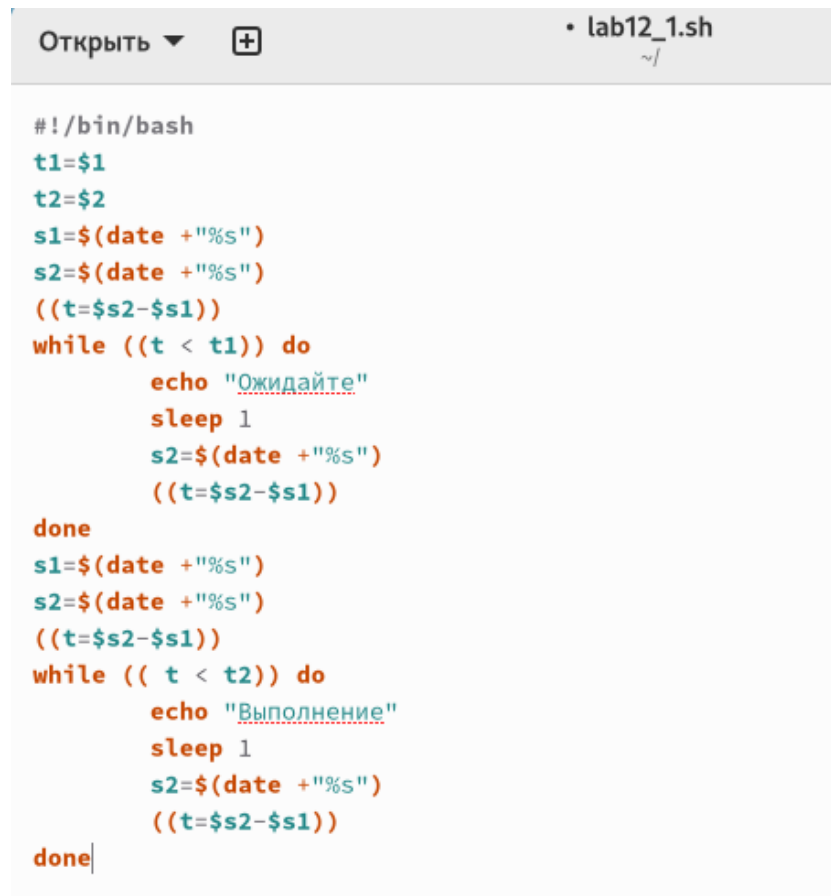
### 3.1 Написание программ

1. Написали 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). Запустили 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. (рис. [3.1]), (рис. [3.2]), (рис. [3.3]), (рис. [3.4])



```
[mazimov@fedora ~]$ touch lab12_1.sh
```

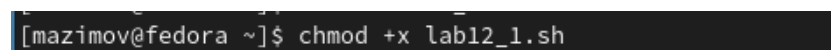
Рис. 3.1: Скрипт 1



```
Открыть ▾ + • lab12_1.sh
~/

#!/bin/bash
t1=$1
t2=$2
s1=$(date +%S)
s2=$(date +%S)
((t=$s2-$s1))
while ((t < t1)) do
    echo "Ожидайте"
    sleep 1
    s2=$(date +%S)
    ((t=$s2-$s1))
done
s1=$(date +%S)
s2=$(date +%S)
((t=$s2-$s1))
while (( t < t2)) do
    echo "Выполнение"
    sleep 1
    s2=$(date +%S)
    ((t=$s2-$s1))
done
```

Рис. 3.2: Скрипт 1



```
[mazimov@fedora ~]$ chmod +x lab12_1.sh
```

Рис. 3.3: Скрипт 1

```
[mazimov@fedora ~]$ ./lab12_1.sh 4 5
Ожидайте
Ожидайте
Ожидайте
Ожидайте
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
```

Рис. 3.4: Скрипт 1

Доработали программу так, чтобы имела возможность взаимодействия трёх и более процессов.(рис. [3.5]), (рис. [3.6])

```
sleep 1
s2=$(date +%s)
((t=s2-$s1))
done
t1=$1
t2=$2
command=$3
while true
do
    if [ "$command" == "Выход" ]
    then echo "Выход"
        exit 0
    fi
    if [ "$command" == "Ожидание" ]
    then pass
    fi
    if [ "$command" == "Выполнение" ]
    then pass
    fi
    echo "Следующее действие"
    read command
done
```

Рис. 3.5: Скрипт 1, доработка



```
Выполнение
[mazimov@fedora ~]$ ./lab12_1.sh 4 5 3
Ожидайте
Ожидайте
Ожидайте
Ожидайте
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
Следующее действие
Следующее действие
Следующее действие
```

Рис. 3.6: Скрипт 1, доработка

2. Реализовали команду `man` с помощью командного файла. Изучили содержимое каталога `/usr/share/man/man1`. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд. Командный файл получает в виде аргумента командной строки название команды и в виде результата выдаёт справку об этой команде или сообщение об отсутствии справки, если соответствующего файла нет в каталоге `man1`. (рис. [3.7]), (рис. [3.8]), (рис. [3.9]), (рис. [3.10]), (рис. [3.11]), (рис. [3.12]), (рис. [3.13])

```
[mazimov@fedora ~]$ cd /usr/share/man/man1
[mazimov@fedora man1]$ ls
.:1.gz
'.1.gz'
ab.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
```

Рис. 3.7: /usr/share/man/man1

```
[mazimov@fedora ~]$ touch lab12_2.sh
```

Рис. 3.8: Скрипт 2

```
Открыть ▾ + lab12_2.sh
~|

#!/bin/bash
c=$1
if [ -f /usr/share/man/man1/${c}.1.gz ]
then
    gunzip -c /usr/share/man/man1/${c}.1.gz | less
else
    echo "Справки по данной команде нет"
fi
```

Рис. 3.9: Скрипт 2

```
[mazimov@fedora ~]$ chmod +x lab12_2.sh
```

Рис. 3.10: Скрипт 2

```
[mazimov@fedora ~]$ ./lab12_2.sh ls
[3]+  Остановлен      ./lab12_2.sh ls
[mazimov@fedora ~]$ ./lab12_2.sh cd
[4]+  Остановлен      ./lab12_2.sh cd
```

Рис. 3.11: Скрипт 2

```
mazimov@fedora:~ — /bin/bash ./lab12_2.sh ls
.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.48.5.
.TH LS "1" "March 2022" "GNU coreutils 9.0" "User Commands"
.SH NAME
ls \- list directory contents
.SH SYNOPSIS
.B ls
[\fI\,OPTION\|\fR]... [\fI\,FILE\|\fR]...
.SH DESCRIPTION
.\" Add any additional description here
.PP
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of \fB\--cftuvSUX\fR nor \fB\--sort\fR is sp
ecified.
.PP
Mandatory arguments to long options are mandatory for short options too.
.TP
\fB\-a\fR, \fB\--all\fR
do not ignore entries starting with .
.TP
\fB\-A\fR, \fB\--almost-all\fR
do not list implied . and ..
.TP
\fB\--author\fR
```

Рис. 3.12: Скрипт 2

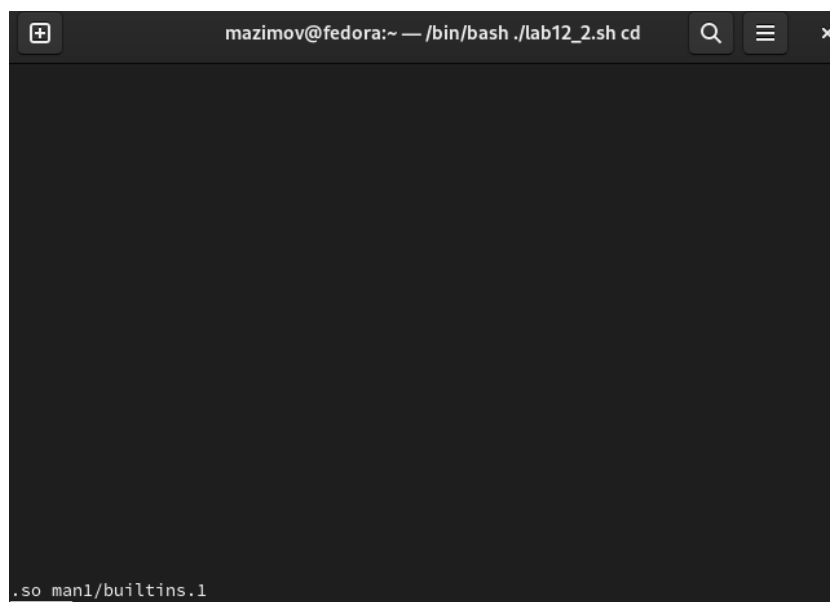


Рис. 3.13: Скрипт 3

3. Используя встроенную переменную \$RANDOM, написали командный файл, генерирующий случайную последовательность букв латинского алфавита, учитывая, что \$RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767. (рис. [3.14]), (рис. [3.15]), (рис. [3.16]), (рис. [3.17])

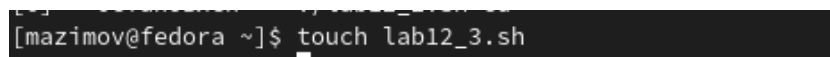
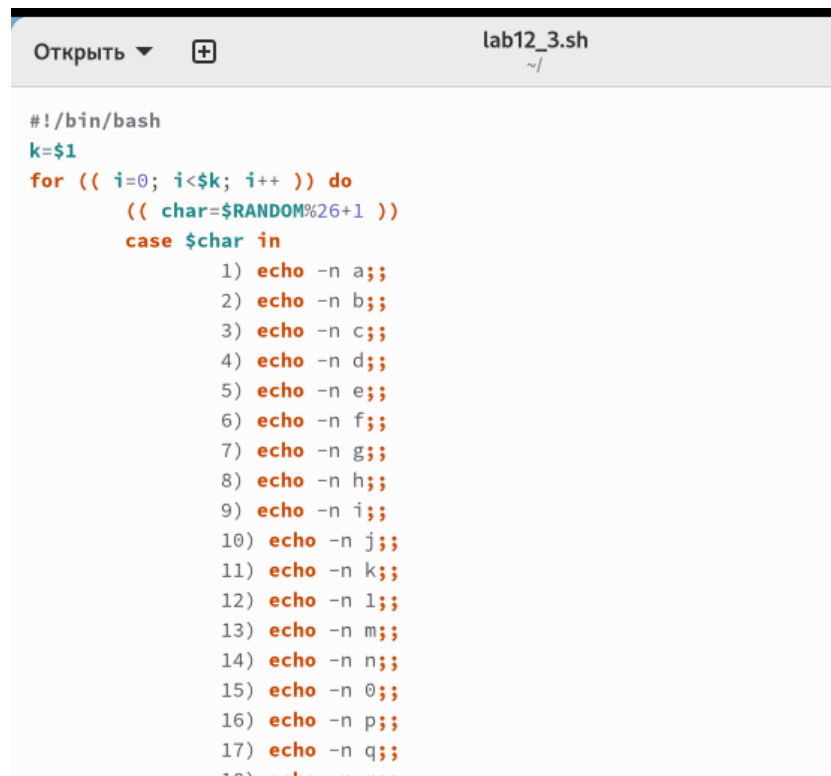


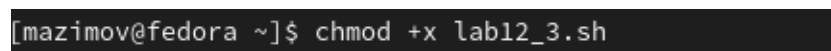
Рис. 3.14: Скрипт 3



The image shows a text editor window with the title bar "lab12\_3.sh". The script content is as follows:

```
#!/bin/bash
k=$1
for (( i=0; i<$k; i++ )) do
    (( char=$RANDOM%26+1 ))
    case $char in
        1) echo -n a;;
        2) echo -n b;;
        3) echo -n c;;
        4) echo -n d;;
        5) echo -n e;;
        6) echo -n f;;
        7) echo -n g;;
        8) echo -n h;;
        9) echo -n i;;
        10) echo -n j;;
        11) echo -n k;;
        12) echo -n l;;
        13) echo -n m;;
        14) echo -n n;;
        15) echo -n o;;
        16) echo -n p;;
        17) echo -n q;;
        18) echo -n r;;
        19) echo -n s;;
        20) echo -n t;;
        21) echo -n u;;
        22) echo -n v;;
        23) echo -n w;;
        24) echo -n x;;
        25) echo -n y;;
        26) echo -n z;;
    esac
done
```

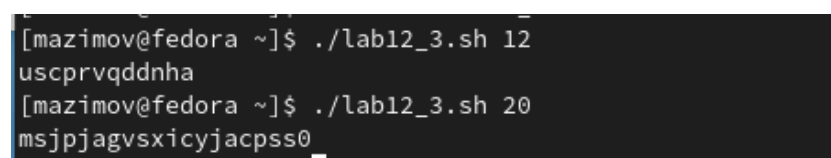
Рис. 3.15: Скрипт 3



The image shows a terminal window with the following command and output:

```
[mazimov@fedora ~]$ chmod +x lab12_3.sh
```

Рис. 3.16: Скрипт 3



The image shows a terminal window with the following commands and output:

```
[mazimov@fedora ~]$ ./lab12_3.sh 12
uscprvqddnha
[mazimov@fedora ~]$ ./lab12_3.sh 20
msjppjagvsxicyjapss0
```

Рис. 3.17: Скрипт 3

## 4 Выводы

В ходе выполнения лабораторной работы были изучены основы программирования в оболочке ОС UNIX. Научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

## 5 Ответы на контрольные вопросы

1. while [\$1 != "exit"] В данной строчке допущены следующие ошибки: • не хватает пробелов после первой скобки [и перед второй скобкой ] • выражение \$1 необходимо взять в “”, потому что эта переменная может содержать пробелы. Таким образом, правильный вариант должен выглядеть так: while ["\$1"!= "exit"]
2. Чтобы объединить несколько строк в одну, можно воспользоваться несколькими способами:

Первый: VAR1="Hello, "VAR2=" World" VAR3="␣␣1VAR2" echo "VAR3" :  
*Hello, World : VAR1 = "Hello,"VAR1+ = "World"echo"VAR1"* Результат: Hello, World

3. Команда seq в Linux используется для генерации чисел от ПЕРВОГО до ПОСЛЕДНЕГО шага INCREMENT. Параметры: seq LAST: если задан только один аргумент, он создает числа от 1 до LAST с шагом шага, равным 1. Если LAST меньше 1, значение is не выдает. seq FIRST LAST: когда заданы два аргумента, он генерирует числа от FIRST до LAST с шагом 1, равным 1. Если LAST меньше FIRST, он не выдает никаких выходных данных. seq FIRST INCREMENT LAST: когда заданы три аргумента, он генерирует числа от FIRST до LAST на шаге INCREMENT . Если LAST меньше, чем FIRST, он не производит вывод. seq -f «FORMAT» FIRST INCREMENT LAST: эта команда используется для генерации последовательности в форматированном виде. FIRST и INCREMENT являются необязательными. seq -s «STRING» ПЕРВЫЙ

ВКЛЮЧЕНО: Эта команда используется для STRING для разделения чисел. По умолчанию это значение равно /n. FIRST и INCREMENT являются необязательными. seq -w FIRST INCREMENT LAST:эта команда используется для выравнивания ширины путем заполнения начальными нулями. FIRST и INCREMENT являются необязательными.

4. Результатом данного выражения  $\$(10/3)$  будет 3, потому что это целочисленное деление без остатка.

5. Отличия командной оболочки zsh от bash:

В zsh более быстрое автодополнение для cdc помощью Tab В zsh существует калькулятор zcalc, способный выполнять вычисления внутри терминала В zsh поддерживаются числа с плавающей запятой В zsh поддерживаются структуры данных «хэш» В zsh поддерживается раскрытие полного пути на основе неполных данных В zsh поддерживается замена части пути В zsh есть возможность отображать разделенный экран, такой же как разделенный экран vim

6. for((a=1; a<= LIMIT; a++)) синтаксис данной конструкции верен, потому что, используя двойные круглые скобки, можно не писать \$ перед переменными ().

7. Преимущества скриптового языка bash:

Один из самых распространенных и ставится по умолчанию в большинстве дистрибутивах Linux, MacOS Удобное перенаправление ввода/вывода Большое количество команд для работы с файловыми системами Linux Можно писать собственные скрипты, упрощающие работу в Linux Недостатки скриптового языка bash: Дополнительные библиотеки других языков позволяют выполнить больше действий Bash не является языком общего назначения Утилиты, при выполнении скрипта, запускают свои процессы, которые, в свою очередь, отражаются на скорости выполнения этого скрипта Скрипты, написанные на bash, нельзя запустить на других операционных системах без дополнительных действий