

# Краткий отчёт по лабораторной работе №12

Samsonova Maria, Student of RUDN University, Moscow, Russian Federation

## Цель выполнения лабораторной работы №12

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

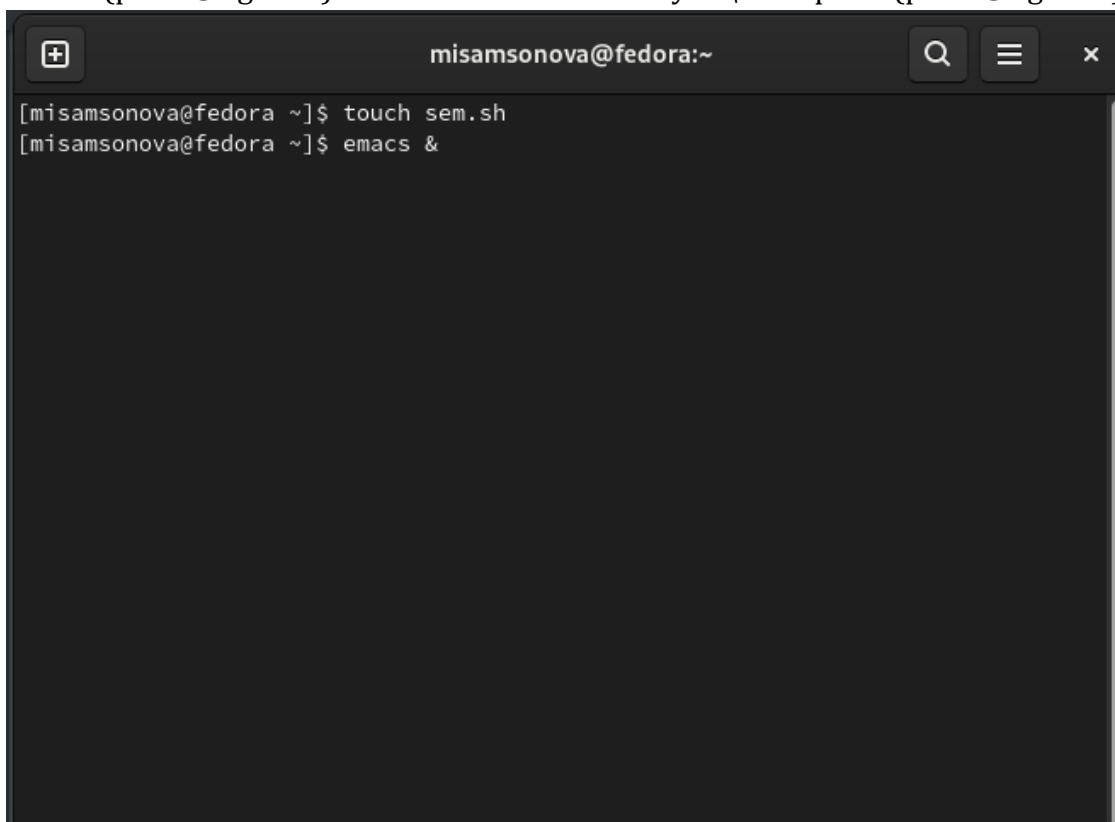
## Задание

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 имела 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 32767.4.

## Ход выполнения лабораторной работы №12

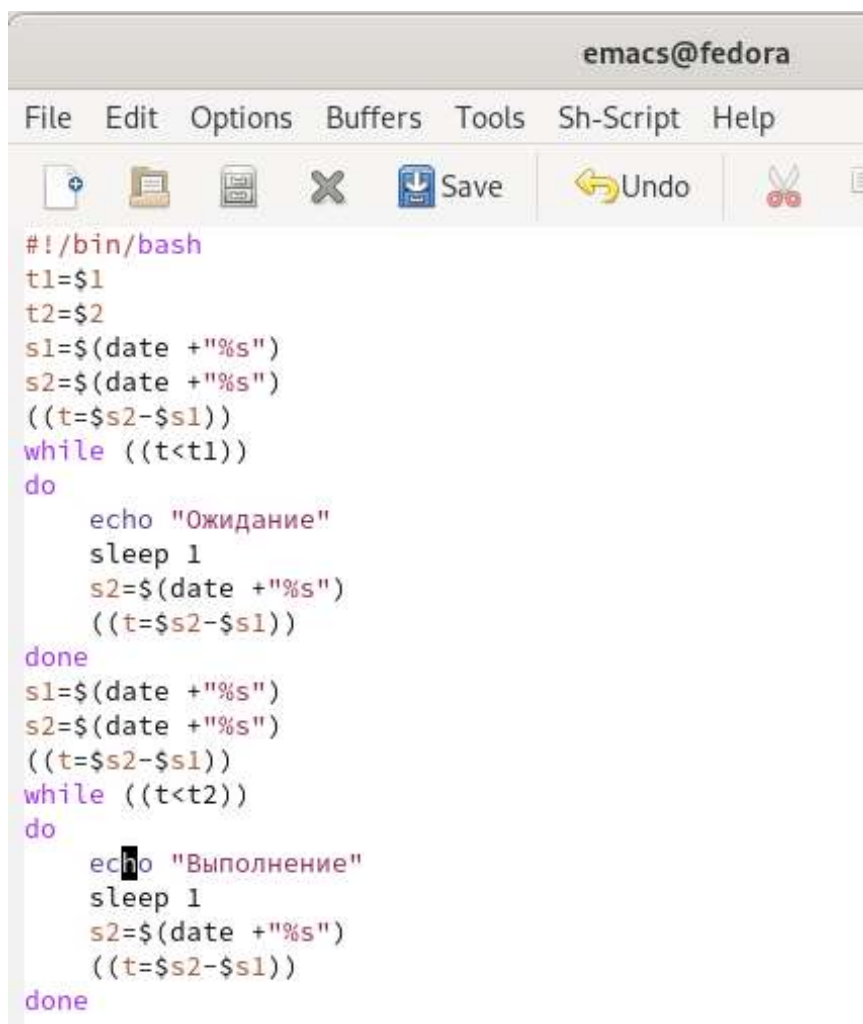
1. Для начала написали командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени  $t_1$  дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени  $t_2 < t_1$ , также выдавая информацию о том, что ресурс используется соответствующим

командным файлом (процессом). Для данной задачи мы создали файл: sem.sh (рис. -@fig:001 ) и написали соответствующий скрипт (рис. -@fig:002 ).



```
misamsonova@fedora:~  
[misamsonova@fedora ~]$ touch sem.sh  
[misamsonova@fedora ~]$ emacs &
```

*Создание файла*



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

#### Скринш №1

- Далее мы проверили работу написанного скрипта (команда «./sem.sh47»), предварительно добавив право на исполнение файла (команда «**chmod +x sem.sh**») (рис. -@fig:003 ). Скрипт работает корректно.

A terminal window with a dark background. The title bar shows a window icon and the text 'misamsonova@fedora:~'. The terminal content shows a user running 'chmod +x sem.sh' and then './sem.sh 2 6'. This is followed by a series of status messages: 'Ожидание' (Waiting), 'Выполнение' (Execution), and 'Выполнение' (Execution), each appearing twice. The terminal ends with a prompt '[misamsonova@fedora ~]\$' and a cursor.

```
[misamsonova@fedora ~]$ chmod +x sem.sh
[misamsonova@fedora ~]$ ./sem.sh 2 6
Ожидание
Ожидание
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
Выполнение
[misamsonova@fedora ~]$
```

### *Проверка работы скрипта*

- После этого мы изменили скрипт так, чтобы его можно было выполнить в нескольких терминалах и проверили его работу (например, команда «./**sem.sh 2 3 Ожидание** > /dev/pts/1 &») (рис. -@fig:004 , -@fig:005 , -@fig:006 ). Однако у нас не получилось проверить работу скрипта, так как было отказно в доступе.

```
emacs@fedora
File Edit Options Buffers Tools Sh-Script Help
Save Undo
#!/bin/bash
function wait{
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=s2-s1))
    while ((t<t1))
    do
        echo "Ожидание"
        sleep 1
        s2=$(date +%s)
        ((t=s2-s1))
    done
}

function complete{
    s1=$(date +%s)
    s2=$(date +%s)
    ((t=s2-s1))
    while ((t<t2))
    do
        echo "Выполнение"
        sleep 1
        s2=$(date +%s)
        ((t=s2-s1))
    done
}

t1=$1
t2=$2
```

*Изменённый скрипт №1*

```
emacs@fedora
File Edit Options Buffers Tools Sh-Script Help
Save Undo
s1=$(date +%s")
s2=$(date +%s")
((t=s2-s1))
while ((t<t2))
do
    echo "Выполнение"
    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 [ "$command" == "Ожидание" ]
    then wait
    fi
    if [ "$command" == "Выполнение" ]
    then complete
    fi
    echo "Следующее действие: "
    read command
done
```

*Изменённый скрипт №1*

```
misamsonova@fedora:~  
[misamsonova@fedora ~]$ ./sem.sh 2 3 Ожидание > /dev/pts/1 &  
[1] 4143  
bash: /dev/pts/1: Отказано в доступе  
[1]+ Выход 1 ./sem.sh 2 3 Ожидание > /dev/pts/1  
[misamsonova@fedora ~]$ sudo ./sem.sh 2 3 Ожидание > /dev/pts/1 &  
[1] 4168  
[misamsonova@fedora ~]$ bash: /dev/pts/1: Отказано в доступе  
[1]+ Выход 1 sudo ./sem.sh 2 3 Ожидание > /dev/pts/1  
[misamsonova@fedora ~]$ ./sem.sh 2 5 Ожидание > /dev/pts/2 &  
[1] 4191  
bash: /dev/pts/2: Отказано в доступе  
[1]+ Выход 1 ./sem.sh 2 5 Ожидание > /dev/pts/2  
[misamsonova@fedora ~]$ ./sem.sh 2 5 Выполнение > /dev/pts/2 &  
[1] 4210  
bash: /dev/pts/2: Отказано в доступе  
[1]+ Выход 1 ./sem.sh 2 5 Выполнение > /dev/pts/2  
[misamsonova@fedora ~]$
```

### Проверка работы скрипта

2. Теперь реализовали команду `man` с помощью командного файла. Изучили содержимое каталога `/usr/share/man/man1` (рис. -@fig:007, -@fig:008). 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`.



## Реализации команды tap







The screenshot shows the Emacs editor interface on a Fedora system. The title bar reads "emacs@fedora". The menu bar includes "File", "Edit", "Options", "Buffers", "Tools", "Sh-Script", and "Help". The toolbar contains icons for opening files, saving, undo, and other standard editing functions. The main text area displays a shell script with the following content:

```
#!/bin/bash
c=$1
if [ -f /usr/share/man/man1/$c.1.gz ]
then
  gunzip -c /usr/share/man/man1/$1.1.gz | less
else
  echo "Справки по данной команде нет"
fi
```

A black cursor is visible on the line following the "fi" statement.

## *Скрипт №2*



The screenshot shows a terminal window with the prompt "misamsonova@fedora:-". The user has executed the following commands:

```
[misamsonova@fedora ~]$ chmod +x man.sh
[misamsonova@fedora ~]$ ./man.sh ls
[misamsonova@fedora ~]$ ./man.sh mkdir
[misamsonova@fedora ~]$
```

## *Проверка работы скрипта*

```
misamsonova@fedora:~ -- /bin/bash ./man.sh ls

.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.47.3.
.TH LS "1" "March 2022" "GNU coreutils 8.32" "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
.\"
List information about the FILES (the current directory by default).
Sort entries alphabetically if none of \fB-\fC\fR, \fB-\fS\fR nor \fB-\fO\fR is specified.
.\"
Mandatory arguments to long options are mandatory for short options too.
.TP
\fB-\fA\fR, \fB-\fO\fR
do not ignore entries starting with .
.TP
\fB-\fA\fR, \fB-\fO\fR
do not list implied . and ..
.TP
\fB-\fO\fR
with \fB-\fO\fR, print the owner of each file
.TP
\fB-\fO\fR, \fB-\fE\fR
print C-style escapes for non-graphic characters
.TP
\fB-\fO\fR, \fB-\fS\fR
with \fB-\fS\fR, scale sizes by SIZE when printing them;
e.g., \fB-\fO\fR, \fB-\fS\fR; see SIZE format below
.TP
\fB-\fO\fR, \fB-\fI\fR
do not list implied entries ending with -
.TP
\fB-\fO\fR
with \fB-\fO\fR: sort by, and show, ctime (time of last
modification of file status information);
with \fB-\fO\fR: show ctime and sort by name;
otherwise: sort by ctime, newest first
.TP
\fB-\fO\fR
list entries by columns
.TP
\fB-\fO\fR, \fB-\fC\fR
colorize the output; WHEN can be 'always' (default
```

*Проверка работы скрипта*

```
misamsonova@fedora:~$ cat /bin/bash ./man.sh mkdir

.\" DO NOT MODIFY THIS FILE! It was generated by help2man 1.47.3.
.TH MKDIR "1" "March 2022" "GNU coreutils 8.32" "User Commands"
.SH NAME
mkdir \- make directories
.SH SYNOPSIS
.B mkdir
[{\FI}\OPTION\{\FI}... {\FI}\DIRECTORY\{\FI}...
.SH DESCRIPTION
.\" Add any additional description here
.PP
Create the DIRECTORY(ies), if they do not already exist.
.PP
Mandatory arguments to long options are mandatory for short options too.
.TP
{\FB}\m{\FR}, {\FB}\-mode{\FR}={\FI}\MODE\{\FR}
set file mode (as in chmod), not a=rwx \- umask
.TP
{\FB}\p{\FR}, {\FB}\-parents{\FR}
no error if existing, make parent directories as needed
.TP
{\FB}\v{\FR}, {\FB}\-verbose{\FR}
print a message for each created directory
.TP
{\FB}\Z{\FR}
set SELinux security context of each created directory
to the default type
.TP
{\FB}\-context{\FR}={\FI}\CTX\{\FR}
like {\FB}\Z{\FR}, or if CTX is specified then set the SELinux
or SMACK security context to CTX
.TP
{\FB}\-help{\FR}
display this help and exit
.TP
{\FB}\-version{\FR}
output version information and exit
.SH AUTHOR
Written by David MacKenzie.
.SH "REPORTING BUGS"
GNU coreutils online help: <https://www.gnu.org/software/coreutils/>
.br
Report any translation bugs to <https://translationproject.org/team/>
.SH COPYRIGHT
Copyright \{c 2020 Free Software Foundation, Inc.
License GPLv3+: GNU GPL version 3 or later <https://gnu.org/licenses/gpl.html>.

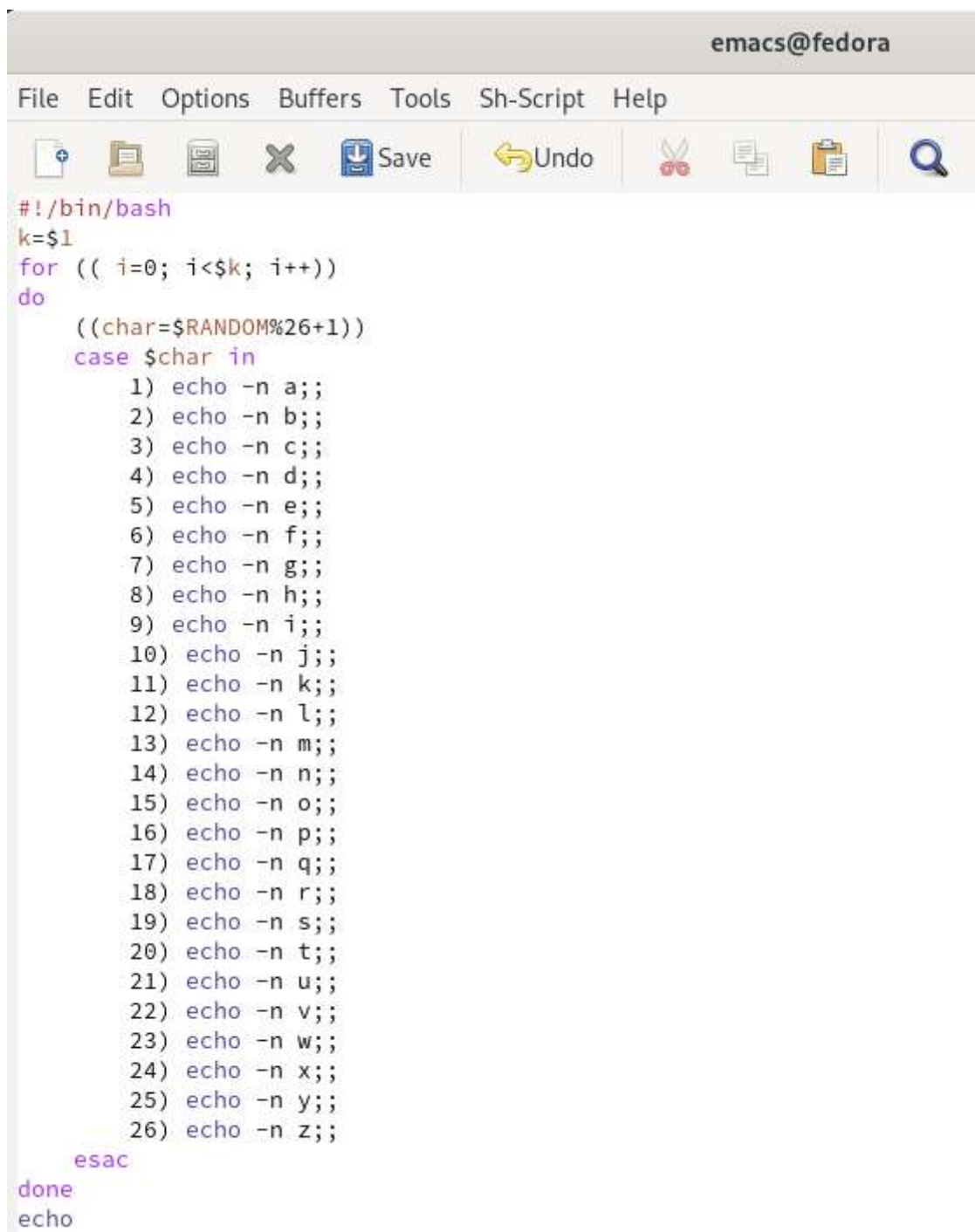
```

### Проверка работы скрипта

- Используя встроенную переменную \$RANDOM, написали командный файл, генерирующий случайную последовательность букв латинского алфавита. Для данной задачи мы создали файл: random.sh (рис. -@fig:014 ) и написали соответствующий скрипт (рис. -@fig:015 ).

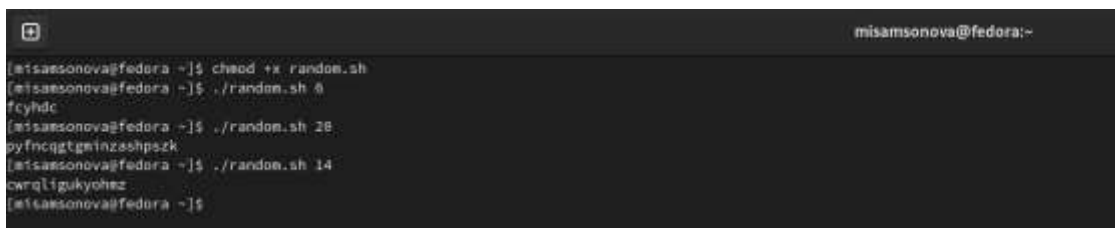
```
misamsonova@fedora:~$ touch random.sh
misamsonova@fedora:~$ emacs &
```

### Создание файла



### Скринш №3

Далее мы проверили работу написанного скрипта (команды «./random.sh 6» и «./random.sh 20»), предварительно добавив право на исполнение файла (команда «chmod +x random.sh») (рис. -@fig:014 ). Скрипт работает корректно.

A terminal window with a dark background. The title bar shows a window icon and the text 'misamsonova@fedora:~'. The terminal content shows a user running a script named 'random.sh' with different arguments. The first run with argument '0' produces the output 'fcyhdc'. The second run with argument '28' produces the output 'pyfmcqgtgminzasshpezk'. The third run with argument '14' produces the output 'cwrqligukyohcz'.

```
misamsonova@fedora ~]$ chmod +x random.sh
misamsonova@fedora ~]$ ./random.sh 0
fcyhdc
misamsonova@fedora ~]$ ./random.sh 28
pyfmcqgtgminzasshpezk
misamsonova@fedora ~]$ ./random.sh 14
cwrqligukyohcz
misamsonova@fedora ~]$
```

*Проверка работы скрипта*

## Вывод выполнения лабораторной работы №12

В процессе выполнения данной лабораторной работы мы изучили основы программирования в оболочке ОС UNIX и научились писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.