

Отчёт по лабораторной работе №12

Операционные системы

Самсонова Мария Ильинична

Содержание

Цель работы	1
Задание	1
Выполнение лабораторной работы	2
Вывод.....	14
Ответы на контрольные вопросы	14

Цель работы

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

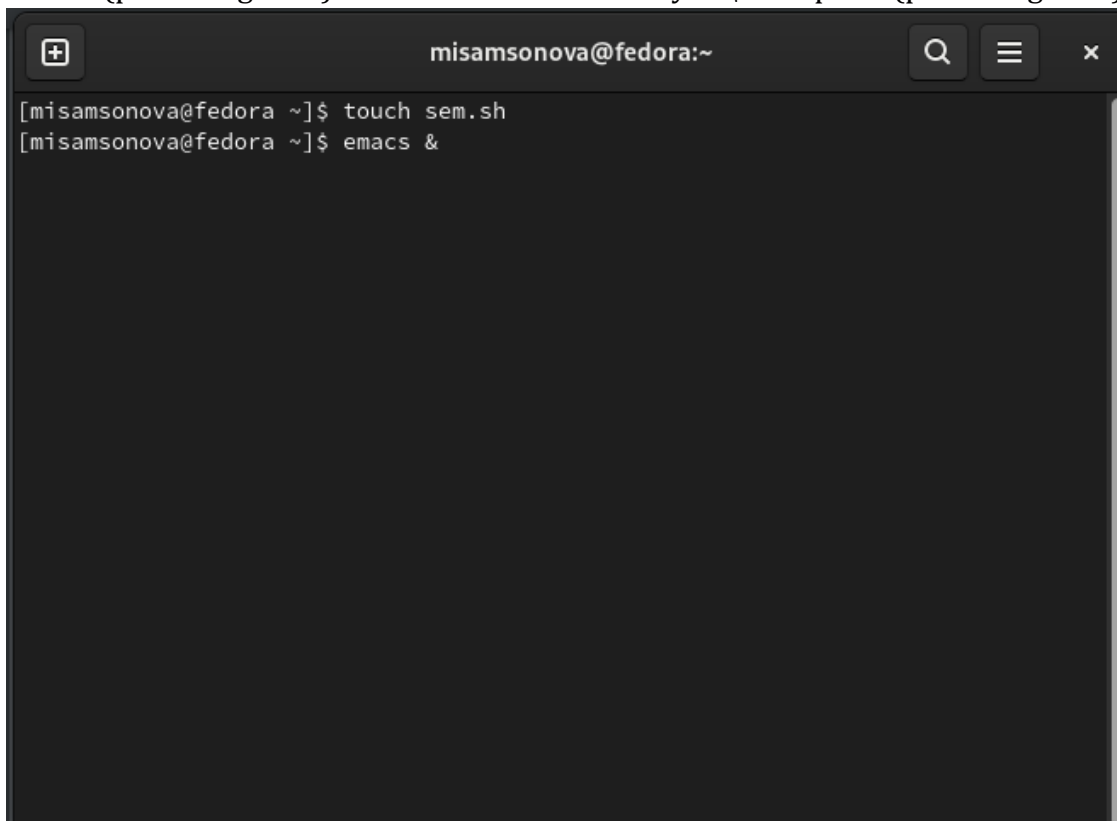
Задание

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 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 32767.4.

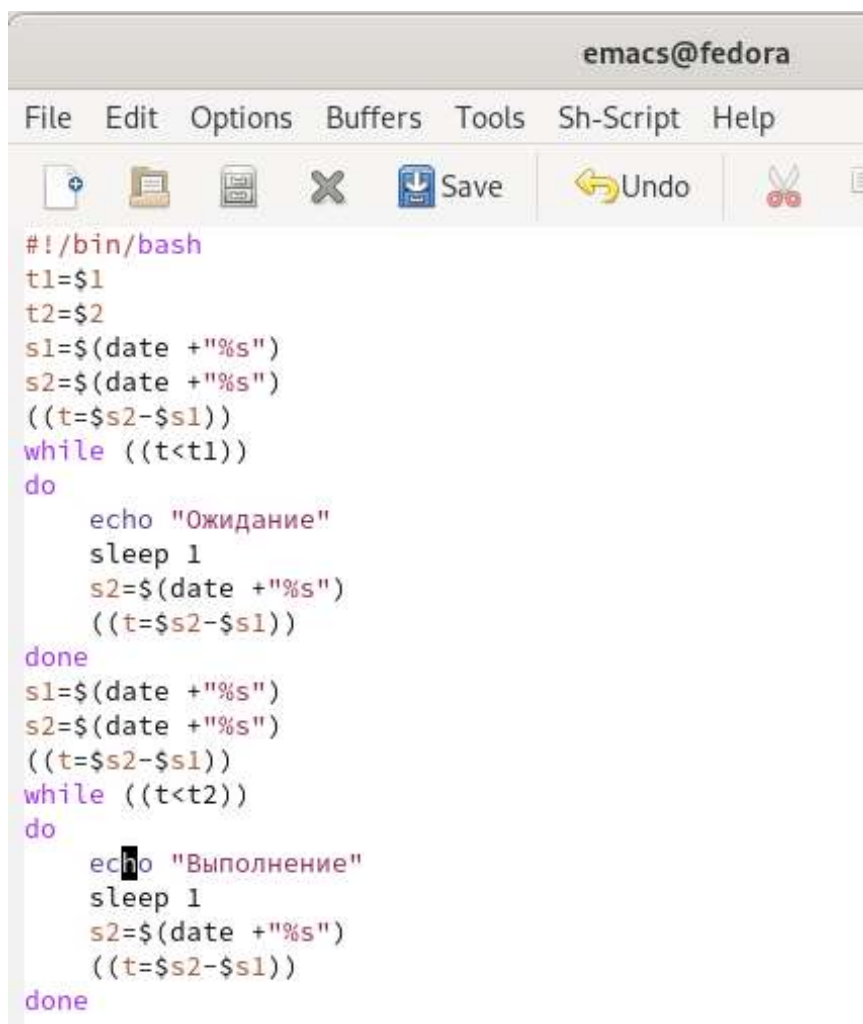
Выполнение лабораторной работы

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 two commands: 'chmod +x sem.sh' and './sem.sh 2 6'. The output of the second command consists of the word 'Ожидание' followed by five instances of 'Выполнение'. The prompt returns to the user.

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

Проверка работы скрипта

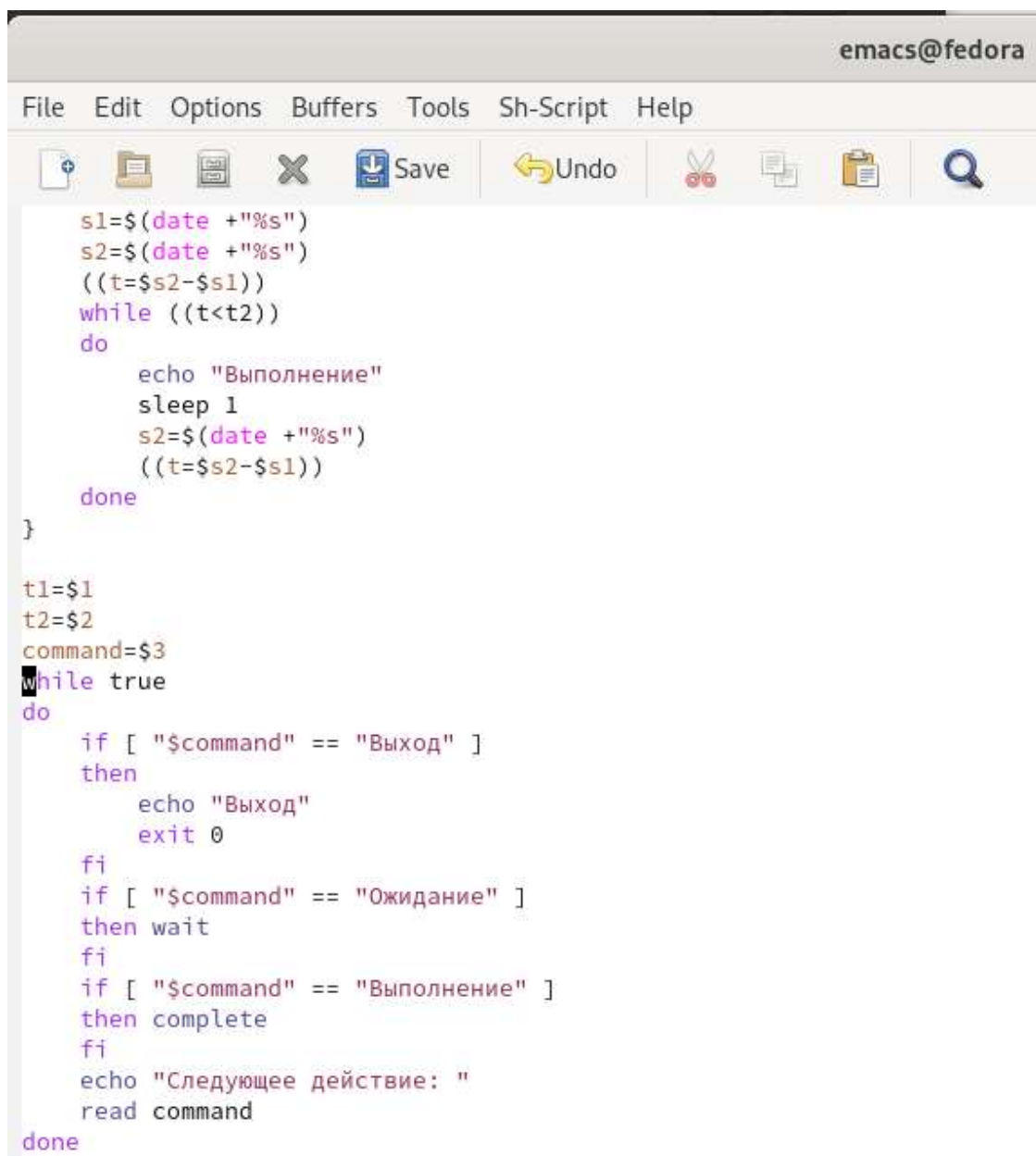
- После этого мы изменили скрипт так, чтобы его можно было выполнить в нескольких терминалах и проверили его работу (например, команда «./**sem.sh 2 3 Ожидание** > /dev/pts/1 &») (рис. -@fig:004 , -@fig:005 , -@fig:006). Однако у нас не получилось проверить работу скрипта, так как было отказно в доступе.

```
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`.

```
[misamsonova@fedora ~]$ ls /usr/share/man/man1
:~.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
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
abrt-action-save-package-data.1.gz
abrt-action-trim-files.1.gz
abrt-applet.1.gz
abrt-auto-reporting.1.gz
abrt-bodhi.1.gz
abrt-cli.1.gz
abrt-dump-journal-core.1.gz
abrt-dump-journal-oops.1.gz
abrt-dump-journal-xorg.1.gz
abrt-dump-oops.1.gz
abrt-dump-xorg.1.gz
abrt-handle-upload.1.gz
abrt-harvest-pstoreoops.1.gz
abrt-harvest-vmcore.1.gz
abrt-merge-pstoreoops.1.gz
abrt-retrace-client.1.gz
abrt-server.1.gz
abrt-watch-log.1.gz
ac.1.gz
aconnect.1.gz
addr2line.1.gz
airscan-discover.1.gz
alias.1.gz
```

Реализациями команды *man*



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 positioned at the end of the last line of the script.

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
.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 specified.
.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
with \fB\-l\fR, print the author of each file
.TP
\fB\-b\fR, \fB\-\-escape\fR
print C\-style escapes for nongraphic characters
.TP
\fB\-\-block-size=SIZE\fR[=FILE], \fB\-\-block-size=SIZE\fR
with \fB\-l\fR, scale sizes by SIZE when printing them;
e.g., '\fB\-\-block-size=M\fR'; see SIZE format below
.TP
\fB\-B\fR, \fB\-\-ignore-backups\fR
do not list implied entries ending with ~
.TP
\fB\-c\fR
with \fB\-lt\fR: sort by, and show, ctime (time of last
modification of file status information);
with \fB\-l\fR: show ctime and sort by name;
otherwise: sort by ctime, newest first
.TP
\fB\-C\fR
list entries by columns
.TP
\fB\-\-color\fR[=FILE], \fB\-\-color\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\{\fR\}... {\fI\}DIRECTORY\{\fR\}...]
.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\}-h{\fR}, {\fB\}-help{\fR}
display this help and exit
.TP
{\fB\}-V{\fR}, {\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>.

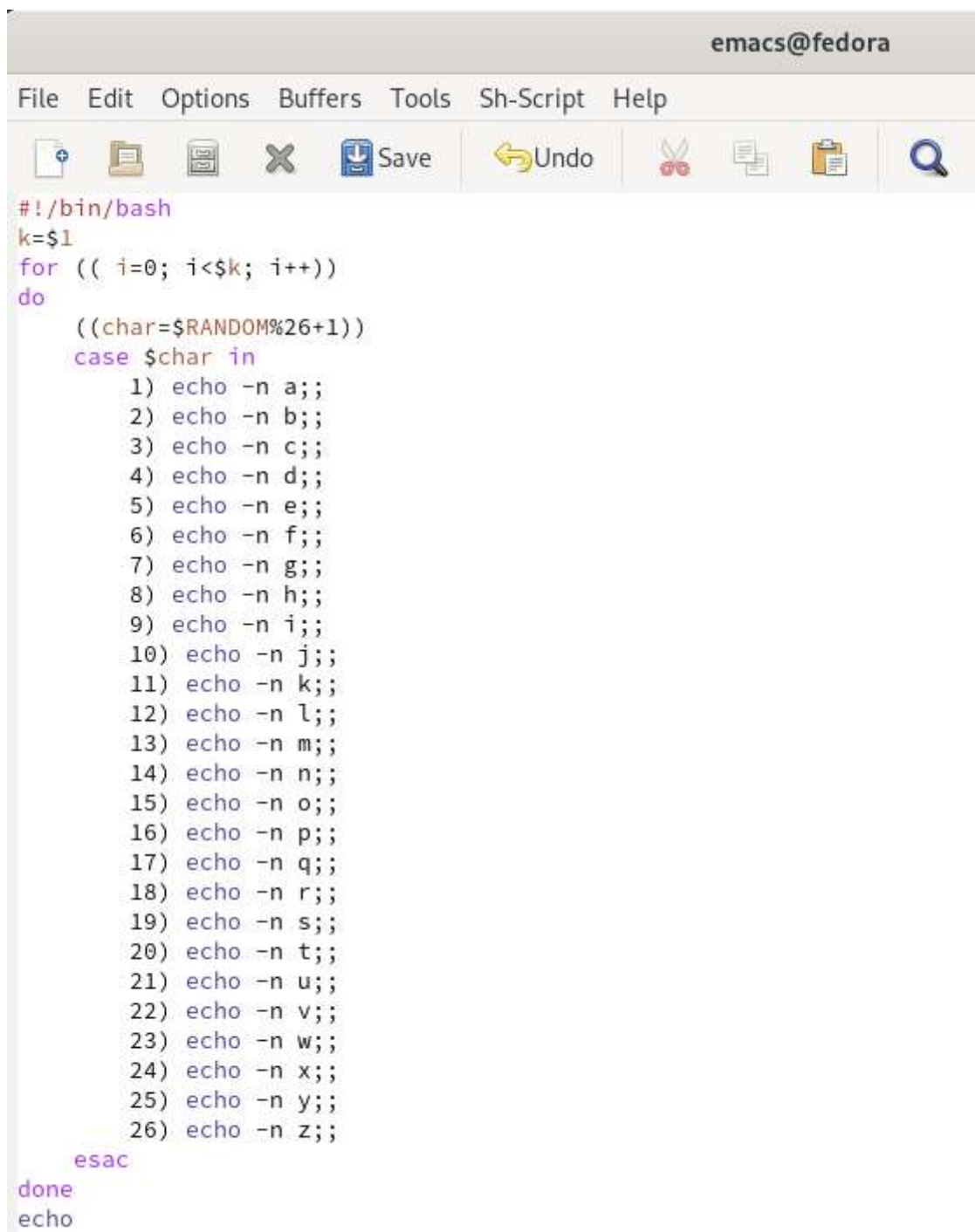
```

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.

```
misamsonova@fedora:~  
[misamsonova@fedora ~]$ chmod +x random.sh  
[misamsonova@fedora ~]$ ./random.sh 6  
fcyhdc  
[misamsonova@fedora ~]$ ./random.sh 28  
pyfmcqgtgminzasshpezk  
[misamsonova@fedora ~]$ ./random.sh 14  
cwrqligukyohcz  
[misamsonova@fedora ~]$
```

Проверка работы скрипта

Вывод

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.

Ответы на контрольные вопросы

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="$VAR1VAR2"
```

```
echo "$VAR3"
```

Результат: Hello, World

- Второй:

```
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.