

РОССИЙСКИЙ УНИВЕРСИТЕТ ДРУЖБЫ НАРОДОВ

Факультет физико-математических и естественных наук

Кафедра прикладной информатики и теории вероятностей

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ № 13

дисциплина: *Операционные системы*

Студент: **ГЕОРГЕС Геден** Группа: **НПМбд-02-20** Студ. бил.:№ **1032204593**

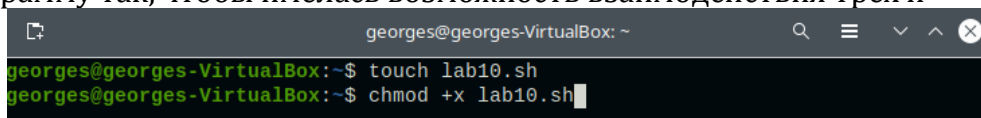
МОСКВА 2021 г.

Цель работы :

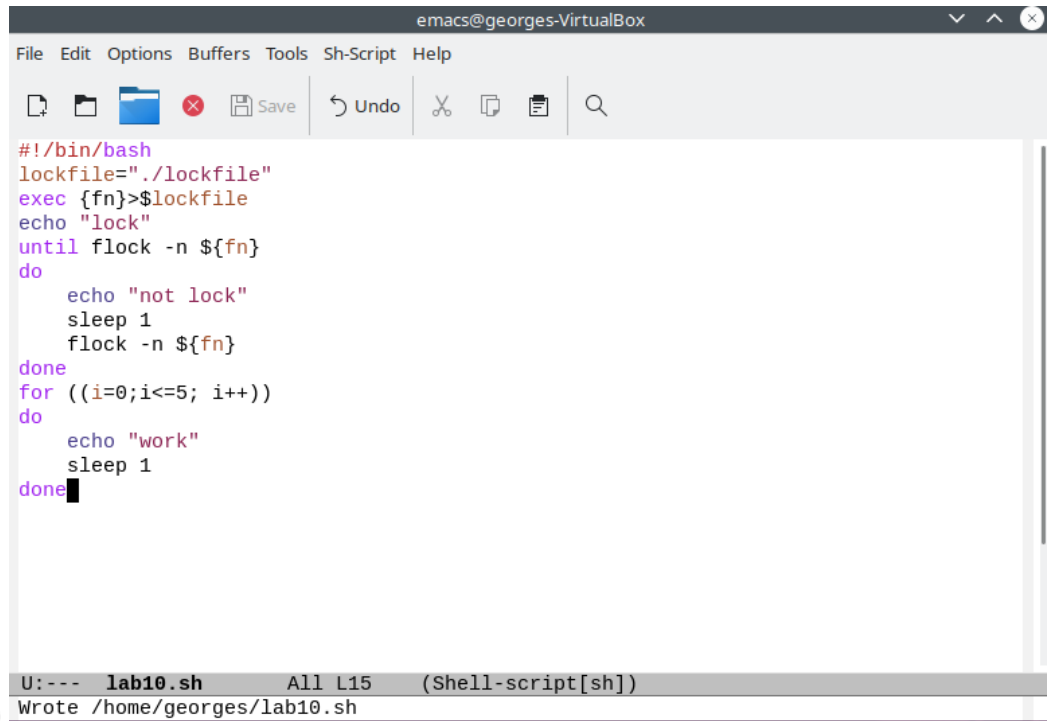
Изучить основы программирования в оболочке ОС UNIX. Научиться писать более сложные командные файлы с использованием логических управляющих конструкций и циклов. # Ход работы: 1. Написал командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t_1 дожидаться 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). Запустила 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.



```
georges@georges-VirtualBox: ~
georges@georges-VirtualBox:~$ touch lab10.sh
georges@georges-VirtualBox:~$ chmod +x lab10.sh
```

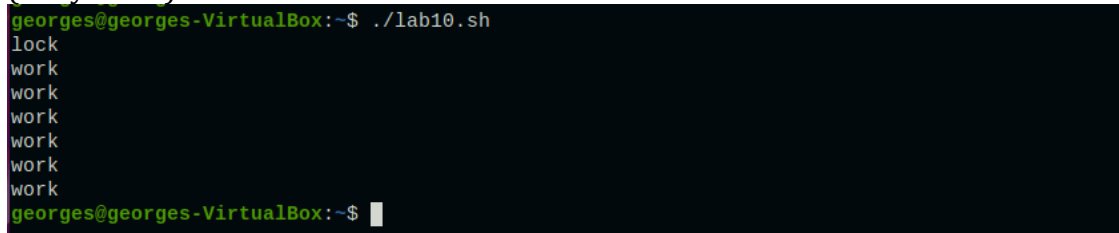


```
#!/bin/bash
lockfile='./lockfile'
exec {fn}>$lockfile
echo "lock"
until flock -n ${fn}
do
    echo "not lock"
    sleep 1
    flock -n ${fn}
done
for ((i=0;i<=5; i++))
do
    echo "work"
    sleep 1
done
```

U:--- lab10.sh All L15 (Shell-script[sh])
Wrote /home/georges/lab10.sh

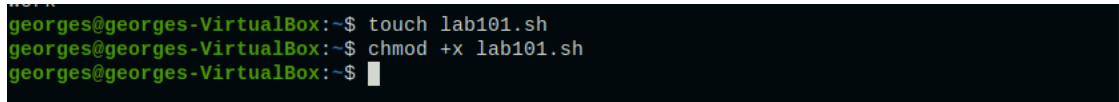
(Рисунок 1)

(Рисунок 2)



```
georges@georges-VirtualBox:~$ ./lab10.sh
lock
work
work
work
work
work
work
georges@georges-VirtualBox:~$
```

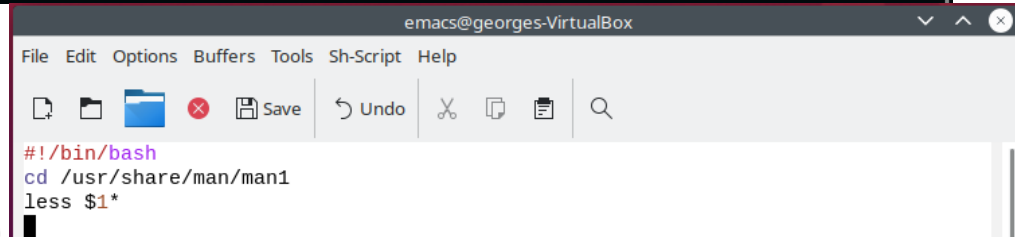
(Рисунок 3) 2. Реализовал команду man с помощью командного файла. Изучил содержимое каталога /usr/share/man/man1. В нем находятся архивы текстовых файлов, содержащих справку по большинству установленных в системе программ и команд. Каждый архив можно открыть командой less сразу же просмотрев содержимое 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.



```
georges@georges-VirtualBox:~$ touch lab101.sh
georges@georges-VirtualBox:~$ chmod +x lab101.sh
georges@georges-VirtualBox:~$
```

(Рисунок 4)

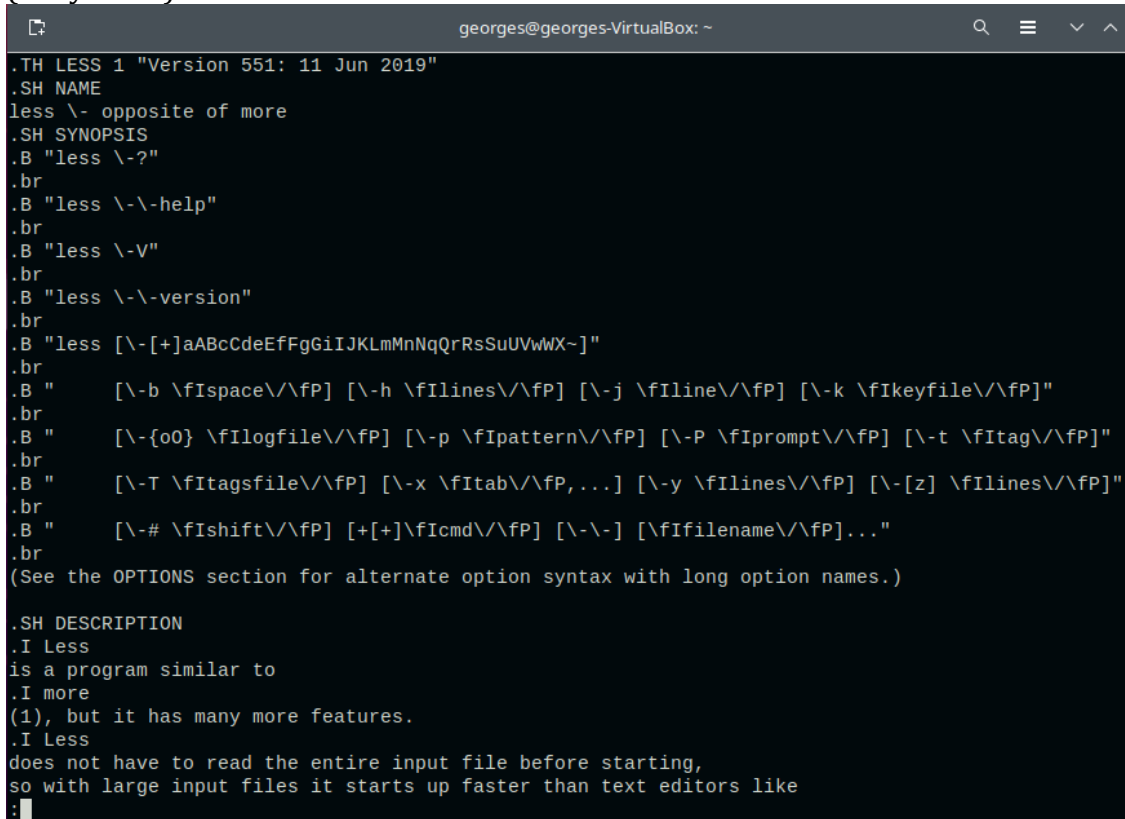
(Рисунок 5)



```
#!/bin/bash
cd /usr/share/man/man1
less $1*
```

```
georges@georges-VirtualBox:~$ ./lab101.sh less
georges@georges-VirtualBox:~$
```

(Рисунок 6)

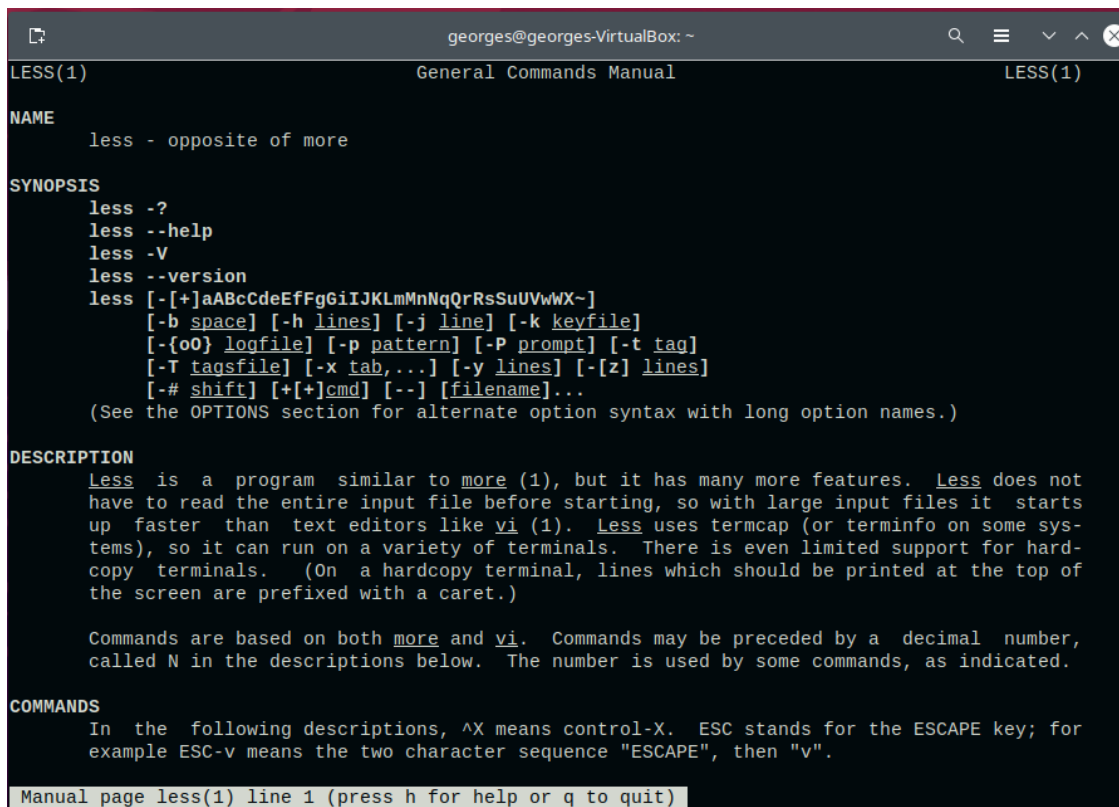


The screenshot shows a terminal window titled 'georges@georges-VirtualBox: ~'. The terminal displays the output of the 'less' command, which is the help text for the 'less' utility. The text is as follows:

```
.TH LESS 1 "Version 551: 11 Jun 2019"
.SH NAME
less \- opposite of more
.SH SYNOPSIS
.B "less \-?"
.br
.B "less \- \-help"
.br
.B "less \-v"
.br
.B "less \- \-version"
.br
.B "less [\- \+ ]aABcCdeEfFgGiIJKLmMnNqQrRsSuUVwWx~]"
.br
.B "      [\-b \fIspace\ /\fP] [\-h \fIlines\ /\fP] [\-j \fIline\ /\fP] [\-k \fIkeyfile\ /\fP]"
.br
.B "      [\- {o0} \fIlogfile\ /\fP] [\-p \fIpattern\ /\fP] [\-P \fIprompt\ /\fP] [\-t \fItag\ /\fP]"
.br
.B "      [\-T \fItagsfile\ /\fP] [\-x \fItab\ /\fP, ...] [\-y \fIlines\ /\fP] [\-z \fIlines\ /\fP]"
.br
.B "      [\-# \fIshift\ /\fP] [\+ \+ ]\fIcmd\ /\fP] [\- \-] [\fIfilename\ /\fP]..."
.br
(See the OPTIONS section for alternate option syntax with long option names.)

.SH DESCRIPTION
.I Less
is a program similar to
.I more
(1), but it has many more features.
.I Less
does not have to read the entire input file before starting,
so with large input files it starts up faster than text editors like
:
```

(Рисунок 7)



```
LESS(1)                                     General Commands Manual                                     LESS(1)

NAME
    less - opposite of more

SYNOPSIS
    less -?
    less --help
    less -V
    less --version
    less [-[+]aABcCdeEfFgGiIJKLmMnNqQrRsSuUVvWxX~]
        [-b space] [-h lines] [-j line] [-k keyfile]
        [-{o0} logfile] [-p pattern] [-P prompt] [-t tag]
        [-T tagsfile] [-x tab,...] [-y lines] [-[z] lines]
        [-# shift] [+[[+]cmd] [--] [filename]...
    (See the OPTIONS section for alternate option syntax with long option names.)

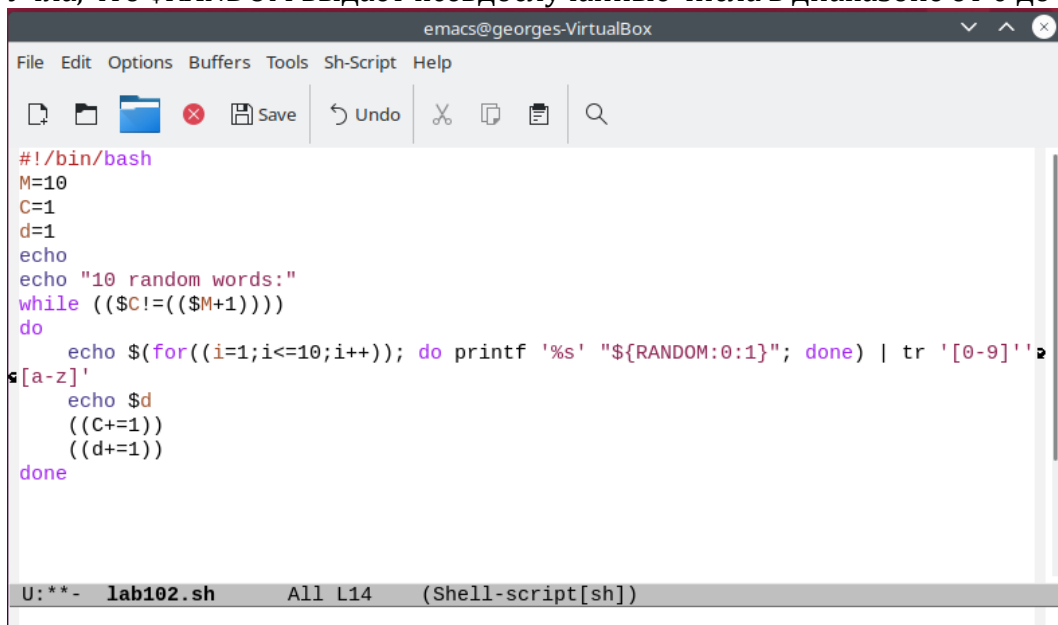
DESCRIPTION
    Less is a program similar to more (1), but it has many more features. Less does not
    have to read the entire input file before starting, so with large input files it starts
    up faster than text editors like vi (1). Less uses termcap (or terminfo on some sys-
    tems), so it can run on a variety of terminals. There is even limited support for hard-
    copy terminals. (On a hardcopy terminal, lines which should be printed at the top of
    the screen are prefixed with a caret.)

    Commands are based on both more and vi. Commands may be preceded by a decimal number,
    called N in the descriptions below. The number is used by some commands, as indicated.

COMMANDS
    In the following descriptions, ^X means control-X. ESC stands for the ESCAPE key; for
    example ESC-v means the two character sequence "ESCAPE", then "v".

Manual page less(1) line 1 (press h for help or q to quit)
```

(Рисунок 8) 3. Используя встроенную переменную \$RANDOM, написал командный файл, генерирующий случайную последовательность букв латинского алфавита. Учла, что \$RANDOM выдаёт псевдослучайные числа в диапазоне от 0 до 32767.



```
#!/bin/bash
M=10
C=1
d=1
echo
echo "10 random words:"
while (($C!=((M+1))))
do
    echo $(for((i=1;i<=10;i++)); do printf '%s' "${RANDOM:0:1}"; done) | tr '[0-9]''[a-z]'
    echo $d
    ((C+=1))
    ((d+=1))
done

U:***- lab102.sh All L14 (Shell-script[sh])
```

(Рисунок 9)

```

georges@georges-VirtualBox:~$ cd /usr/share/man/man1
georges@georges-VirtualBox:~$ ./lab102.sh

10 random words:
bbjccccbbc
1
bdfcccccb
2
bbiddccbbd
3
cbebfccbcd
4
bcdbfbcccb
5
ccbdbdbcbc
6
cbjbbcbcbg
7
bbbbbdicbd
8
bdcfecbdci
9
bbccdcdbdb
10
georges@georges-VirtualBox:~$

```

(Рисунок 10) 4. Реализовал команду man с помощью командного файла. Изучил содержимое каталога /usr/share/man/man1 (Рисунок 11). 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.

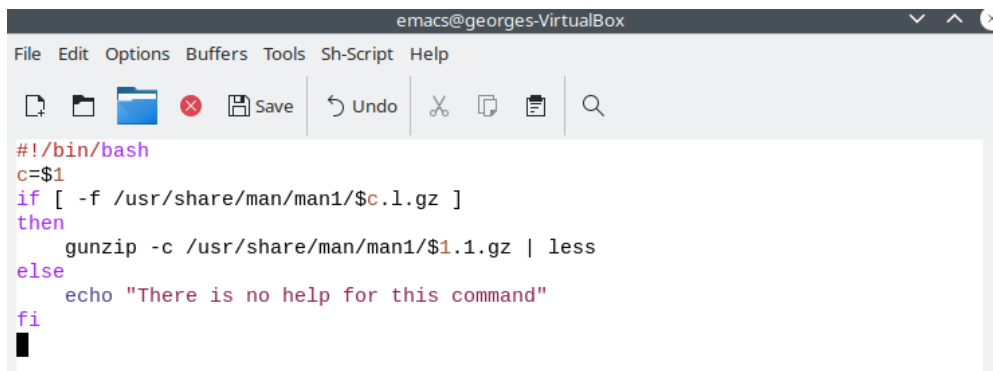
```

georges@georges-VirtualBox:~$ cd /usr/share/man/man1
georges@georges-VirtualBox:/usr/share/man/man1$ ls
'.1.gz'          mkocp.1.gz
4i1toppm.1.gz    mkofm.1.gz
7z.1.gz          mksquashfs.1.gz
7za.1.gz         mktemp.1.gz
7zr.1.gz         mktexfmt.1.gz
aa-enabled.1.gz  mktexlsr.1.gz
aa-exec.1.gz     mktexmf.1.gz
aconnect.1.gz    mktexpk.1.gz
add-apt-repository.1.gz
addr2line.1.gz   mktextfm.1.gz
afm2pl.1.gz      mkzftree.1.gz
afm2tfm.1.gz     mlabel.1.gz
aleph.1.gz       mlocate.1.gz
alsabat.1.gz     mmcli.1.gz
alsactl.1.gz     mmd.1.gz
alsaloop.1.gz    mmount.1.gz
alsamixer.1.gz   mmove.1.gz
alsatplg.1.gz    mogrify.1.gz
alsaucm.1.gz     mogrify-im6.1.gz
amidi.1.gz       mogrify-im6.q16.1.gz
amixer.1.gz      montage.1.gz
animate.1.gz     montage-im6.1.gz
animate-im6.1.gz montage-im6.q16.1.gz
animate-im6.q16.1.gz
anytopnm.1.gz    more.1.gz
apg.1.gz         mountpoint.1.gz
apgbfm.1.gz      mousetweaks.1.gz
aplay.1.gz       mpartition.1.gz
aplaymidi.1.gz   mpost.1.gz
apport-bug.1.gz  mpris-proxy.1.gz
                mptopdf.1.gz
                mrd.1.gz

```

(Рисунок 11)

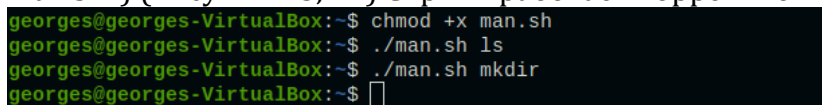
4. 1. Для данной задачи я создал файл: man.sh (Рисунок 12) и написал соответствующий скрипт



```
#!/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 "There is no help for this command"
fi
```

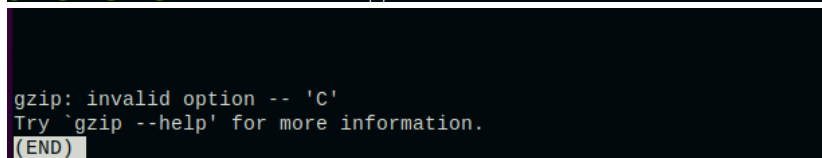
(Рисунок 12) 5.

Далее я проверил работу написанного скрипта (команды «./man.sh ls» и «./man.sh mkdir»), предварительно добавив право на исполнение файла (команда «chmod +x man.sh») (Рисунки 13, 14) Скрипт работает корректно.



```
georges@georges-VirtualBox:~$ chmod +x man.sh
georges@georges-VirtualBox:~$ ./man.sh ls
georges@georges-VirtualBox:~$ ./man.sh mkdir
georges@georges-VirtualBox:~$
```

(Рисунок 13)



```
gzip: invalid option -- 'C'
Try 'gzip --help' for more information.
(END)
```

(Рисунок 14) # Вывод:

изучила основы программирования в оболочке ОС UNIX, научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов # Ответы на контрольные вопросы: 1. В строке while [\$1 != "exit"] квадратные скобки надо заменить на круглые. 2. Есть несколько видов конкатенации строк. Например, VAR1="Hello," VAR2=" World" VAR3="VAR1VAR2" echo "\$VAR3" 3. Команда seq выводит последовательность целых или действительных чисел, подходящую для передачи в другие программы. В bash можно использовать seq с циклом for, используя подстановку команд. Например, \$ for i in \$(seq 1 0.5 4) do echo "The number is \$i" done 4. Результатом вычисления выражения \$((10/3)) будет число 3. 5. Список того, что можно получить, используя Z Shell вместо Bash: Встроенная команда zmv поможет массово переименовать файлы/директории, например, чтобы добавить '.txt' к имени каждого файла, запустите zmv -C '*(*)' '#q.' '\$1.txt'. Утилита zcalc — это замечательный калькулятор командной строки, удобный способ считать быстро, не покидая терминал. Команда zparseopts — это однострочник, который поможет разобрать сложные варианты, которые предоставляются скрипту. Команда autopushd позволяет делать popd после того, как с помощью cd, чтобы вернуться в предыдущую директорию. Поддержка чисел с плавающей точкой (коей Bash не содержит). Поддержка для структур данных «хэш». Есть также ряд особенностей, которые присутствуют только в Bash: Опция командной строки -погс, которая позволяет пользователю иметь дело с инициализацией командной строки, не читая файл .bashrc Использование опции -rcfile с bash позволяет исполнять команды из определённого файла. Отличные возможности вызова (набор опций для командной строки) Может быть вызвана командой sh Bash можно запустить в определённом режиме POSIX. Примените set -o posix, чтобы включить режим, или --posix при запуске. Можно управлять видом командной строки в Bash. Настройка переменной PROMPT_COMMAND с одним или

более специальными символами настроит её за вас. Bash также можно включить в режиме ограниченной оболочки (с `rbash` или `-restricted`), это означает, что некоторые команды/действия больше не будут доступны: Настройка и удаление значений служебных переменных `SHELL`, `PATH`, `ENV`, `BASH_ENV` Перенаправление вывода с использованием операторов `>`, `>|`, `<>`, `>&`, `&>`, `>>` Разбор значений `SHELLOPTS` из окружения оболочки при запуске Использование встроенного оператора `exes`, чтобы заменить оболочку другой командой 6. Синтаксис конструкции `for ((a=1; a <= LIMIT; a++))` верен. 7. Язык `bash` и другие языки программирования: - Скорость работы программ на ассемблере может быть более 50% медленнее, чем программ на `си/си++`, скомпилированных с максимальной оптимизацией; - Скорость работы виртуальной ява-машины с байт-кодом часто превосходит скорость аппаратуры с кодами, получаемыми трансляторами с языков высокого уровня. Ява-машина уступает по скорости только ассемблеру и лучшим оптимизирующим трансляторам; - Скорость компиляции и исполнения программ на `яваскрипт` в популярных браузерах лишь в 2-3 раза уступает лучшим трансляторам и превосходит даже некоторые качественные компиляторы, безусловно намного (более чем в 10 раз) обгоняя большинство трансляторов других языков сценариев и подобных им по скорости исполнения программ; - Скорость кодов, генерируемых компилятором языка `си` фирмы `Intel`, оказалась заметно меньшей, чем компилятора `GNU` и иногда `LLVM`; - Скорость ассемблерных кодов `x86-64` может меньше, чем аналогичных кодов `x86`, примерно на 10%; - Оптимизация кодов лучше работает на процессоре `Intel`; - Скорость исполнения на процессоре `Intel` была почти всегда выше, за исключением языков `лисп`, `эрланг`, `аук` (`gawk`, `mawk`) и `бэш`. Разница в скорости по `бэш` скорее всего вызвана разными настройками окружения на тестируемых системах, а не собственно транслятором или железом. Преимущество `Intel` особенно заметно на 32-разрядных кодах; - Стек большинства тестируемых языков, в частности, `ява` и `яваскрипт`, поддерживают только очень ограниченное число рекурсивных вызовов. Некоторые трансляторы (`gcc`, `icc`, ...) позволяют увеличить размер стека изменением переменных среды исполнения или параметром; - В рассматриваемых версиях `gawk`, `php`, `perl`, `bash` реализован динамический стек, позволяющий использовать всю память компьютера. Но `perl` и, особенно, `bash` используют стек настолько экстенсивно, что 8-16 ГБ не хватает для расчета `ask(5,2,3)`.