

Отчёт по лабораторной работе № 12

Программирование в командном процессоре ОС UNIX. Расширенное программирование

Кристина Алексеевна Антипина НБИбд-01-21

Содержание

Цель работы	1
Задания.....	1
Ход работы:	2
Вывод:	7
Ответы на контрольные вопросы:	7

–

Цель работы

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

Задания

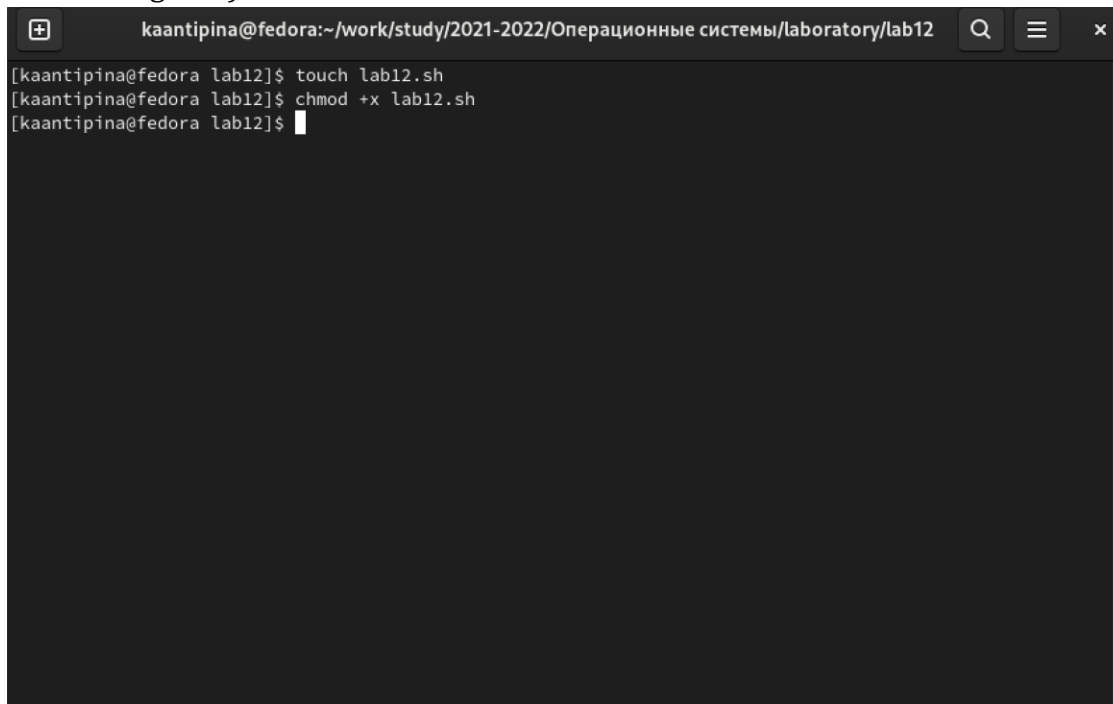
1. Написать командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени t_1 дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени $t_2 < t_1$, также выдавая информацию 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 случайные числа в диапазоне от 0 до 32767.

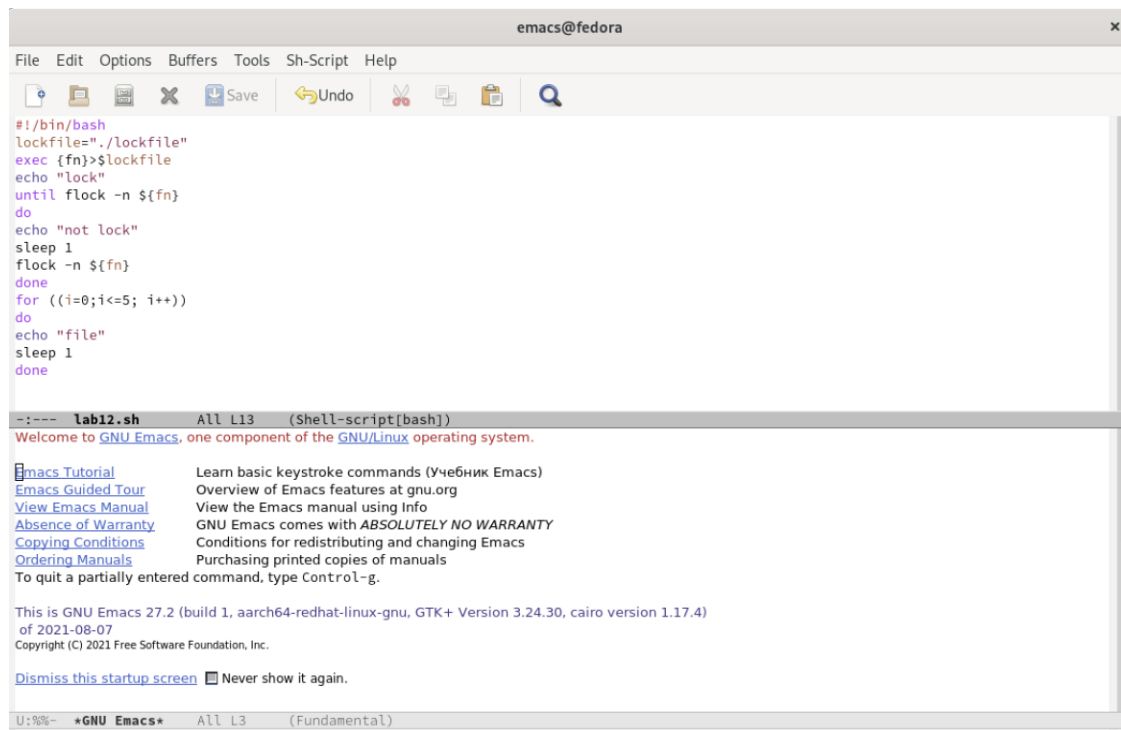
Ход работы:

1. Напишу командный файл, реализующий упрощённый механизм семафоров. Командный файл должен в течение некоторого времени `t1` дожидаться освобождения ресурса, выдавая об этом сообщение, а дождавшись его освобождения, использовать его в течение некоторого времени `t2 < t1`, также выдавая информацию о том, что ресурс используется соответствующим командным файлом (процессом). Запускаю 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. (рис. -@fig:001)(рис. -@fig:002)(рис. -@fig:003)

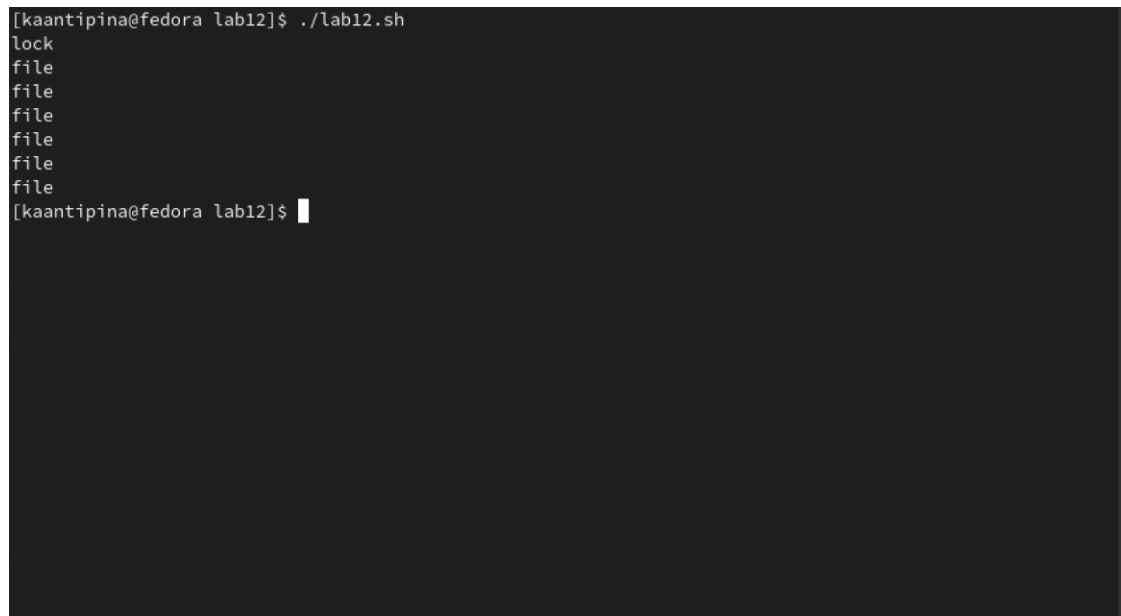


```
kaantipina@fedora:~/work/study/2021-2022/Операционные системы/laboratory/lab12
[kaantipina@fedora lab12]$ touch lab12.sh
[kaantipina@fedora lab12]$ chmod +x lab12.sh
[kaantipina@fedora lab12]$
```

Создаю файл



Напишу скрипт



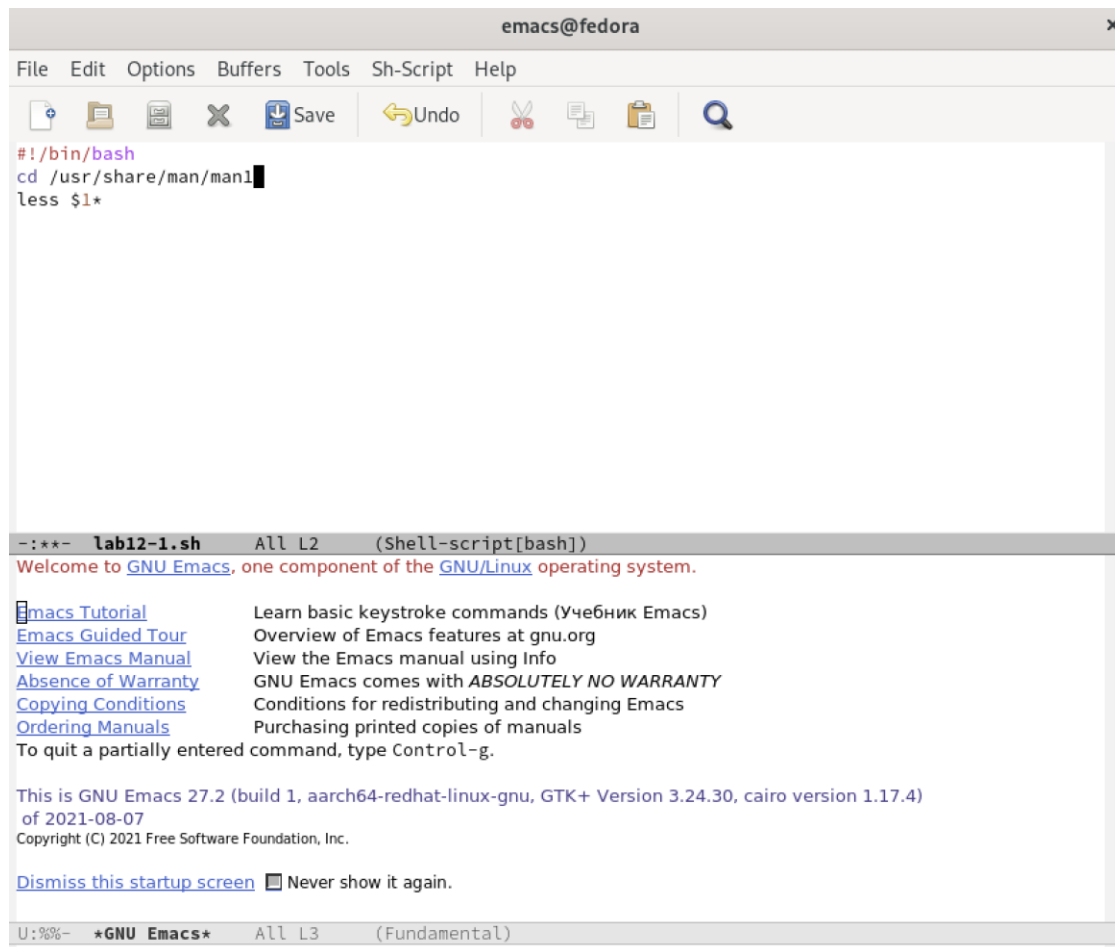
Программа

2. Реализую команду `man` с помощью командного файла. Изучу 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.(рис. -@fig:004)(рис. -@fig:005)(рис. -@fig:006)(рис. -@fig:007)(рис. -@fig:008)

```
[kaantipina@fedora lab12]$ touch lab12-1.sh
[kaantipina@fedora lab12]$ chmod +x lab12-1.sh
[kaantipina@fedora lab12]$
```

Создаю файл



Напишу скринт

```
[kaantipina@fedora lab12]$ ./lab12-1.sh less
```

Программа

```
kaantipina@fedora:~/work/study/2021-2022/Операционные системы/laboratory/lab12 — /bin/bash ./lab12-1.sh L...
LESS(1)                                     General Commands Manual                                     LESS(1)

ESC[1mNAMEESC[0m
less - opposite of more

ESC[1mSYNOPSISESC[0m
ESC[1mless -?ESC[0m
ESC[1mless --helpESC[0m
ESC[1mless -VESC[0m
ESC[1mless --versionESC[0m
ESC[1mless [-[+]aABcCdEeFfGgIiJkLmMnNqQrRsSuUvVwWx~]ESC[0m
ESC[1m[-b ESC[4mESC[22mspaceESC[24mESC[1m] [-h ESC[4mESC[22mlinesESC[24mESC[1m] [-j ESC[4mESC[22mlineESC[24mESC[1m] [-k ESC[4mESC[22mkeyfileESC[24mESC[1m]ESC[0m]
ESC[1m[-{o0} ESC[4mESC[22mlogfileESC[24mESC[1m] [-p ESC[4mESC[22mpatternESC[24mESC[1m] [-P ESC[4mESC[22mpromptESC[24mESC[1m] [-t ESC[4mESC[22mtagESC[24mESC[1m]ESC[0m]
ESC[1m[-T ESC[4mESC[22mtagsfileESC[24mESC[1m] [-x ESC[4mESC[22mtabESC[24mESC[1m,...] [-y ESC[4mESC[22mlinesESC[24mESC[1m] [-z] ESC[4mESC[22mlinesESC[24mESC[1m]ESC[0m]
ESC[1m[-# ESC[4mESC[22mshiftESC[24mESC[1m] [+][+ESC[4mESC[22mcmdESC[24mESC[1m] [--] ESC[4mESC[22mfilenameESC[24mESC[1m]...ESC[0m]
(See the OPTIONS section for alternate option syntax with long option names.)

ESC[1mDESCRIPTIONESC[0m
ESC[4mLessESC[24m is a program similar to ESC[4mmoreESC[24m(1), but which allows backward movement in the file as well as forward movement. Also, ESC[4mlessESC[24m does not have to read the entire input file before starting, so with large input files it starts up faster than text editors like ESC[4mviESC[24m(1). ESC[4mLessESC[24m uses termcap

.
```

Программа

```
kaantipina@fedora:~/work/study/2021-2022/Операционные системы/laboratory/lab12 — /bin/bash ./lab12-1.sh L...

SPACE or ^V or f or ^F
Scroll forward N lines, default one window (see option -z below). If N is more than the screen size, only the final screenful is displayed. Warning: some systems use ^V as a special literalization character.

z
Like SPACE, but if N is specified, it becomes the new window size.

ESC-SPACE
Like SPACE, but scrolls a full screenful, even if it reaches end-of-file in the process.

ENTER or RETURN or ^N or e or ^E or j or ^J
Scroll forward N lines, default 1. The entire N lines are displayed, even if N is more than the screen size.

d or ^D
Scroll forward N lines, default one half of the screen size. If N is specified, it becomes the new default for subsequent d and u commands.

b or ^B or ESC-v
Scroll backward N lines, default one window (see option -z below). If N is more than the screen size, only the final screenful is displayed.

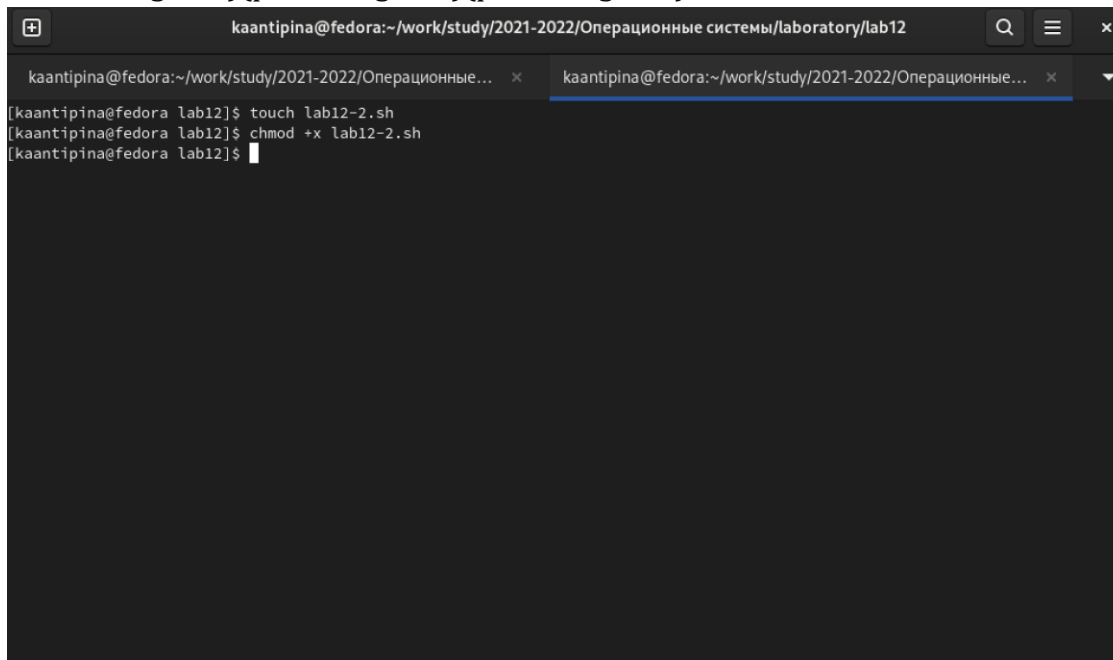
w
Like ESC-v, but if N is specified, it becomes the new window size.

y or ^Y or ^P or k or ^K
Scroll backward N lines, default 1. The entire N lines are displayed, even if N is
```

Программа

- Используя встроенную переменную \$RANDOM, напишу командный файл, генерирующий случайную последовательность букв латинского алфавита.

Учту, что \$RANDOM выдаёт случайные числа в диапазоне от 0 до 32767.(рис. - @fig:009)(рис. -@fig:010)(рис. -@fig:011)



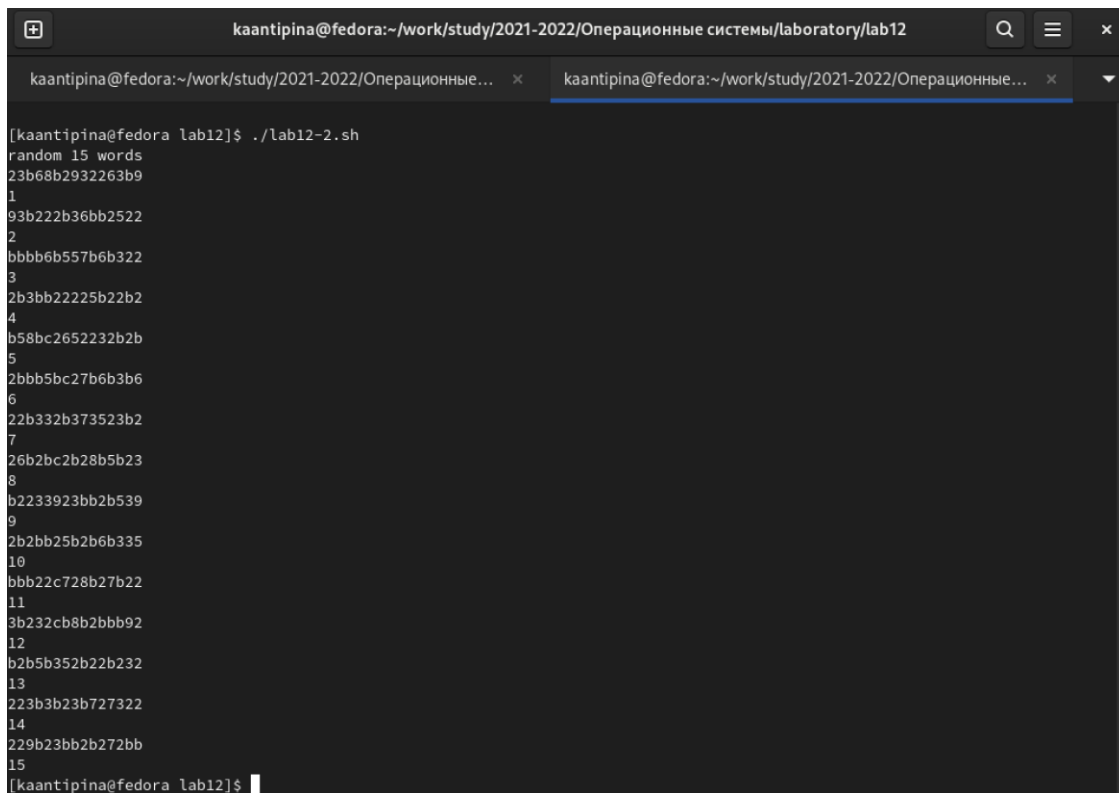
```
kaantipina@fedora:~/work/study/2021-2022/Операционные системы/laboratory/lab12
[kaantipina@fedora lab12]$ touch lab12-2.sh
[kaantipina@fedora lab12]$ chmod +x lab12-2.sh
[kaantipina@fedora lab12]$
```

Создаю текстовый файл



```
lab12-2.sh
~/work/study/2021-2022/Операционные системы/laboratory/lab12
1 #!/bin/bash
2 X=15
3 A=1
4 K=1
5 echo "random 15 words"
6 while (($A!=($X+1)))
7 do
8 echo ${for((i=1;i<=15;i++)); do printf '%s' "${RANDOM:0:1}"; done} | tr '[:0-14:]' '[:a-z:]'
9 echo $K
10 ((A+=1))
11 ((K+=1))
12 done
```

Напишу скрипт



```
[kaantipina@fedora lab12]$ ./lab12-2.sh
random 15 words
23b68b2932263b9
1
93b222b36bb2522
2
bbbb6b557b6b322
3
2b3bb22225b22b2
4
b58bc2652232b2b
5
2bbb5bc27b6b3b6
6
22b332b373523b2
7
26b2bc2b28b5b23
8
b2233923bb2b539
9
2b2bb25b2b6b335
10
bbb22c728b27b22
11
3b232cb8b2bbb92
12
b2b5b352b22b232
13
223b3b23b727322
14
229b23bb2b272bb
15
[kaantipina@fedora lab12]$
```

Программа

Вывод:

В данной лабораторной работе № 12 я 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 `-norc`, 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, iss, ...) 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)