

**Федеральное государственное автономное
Образовательное учреждение высшего образования
Российский Университет Дружбы Народов**

Математический университет имени Никольского
Факультет Физико-математических и Естественных наук
Кафедра Прикладной математики и информатики

Отчет по лабораторной работе № 12
“Программирование в командном
процессоре ОС UNIX. Расширенное программирование”

Выполнил:
Студент группы НПИМбв-01-10
Адхамова Луиза Шухратовна

Москва
2024 год

Цель работы:

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

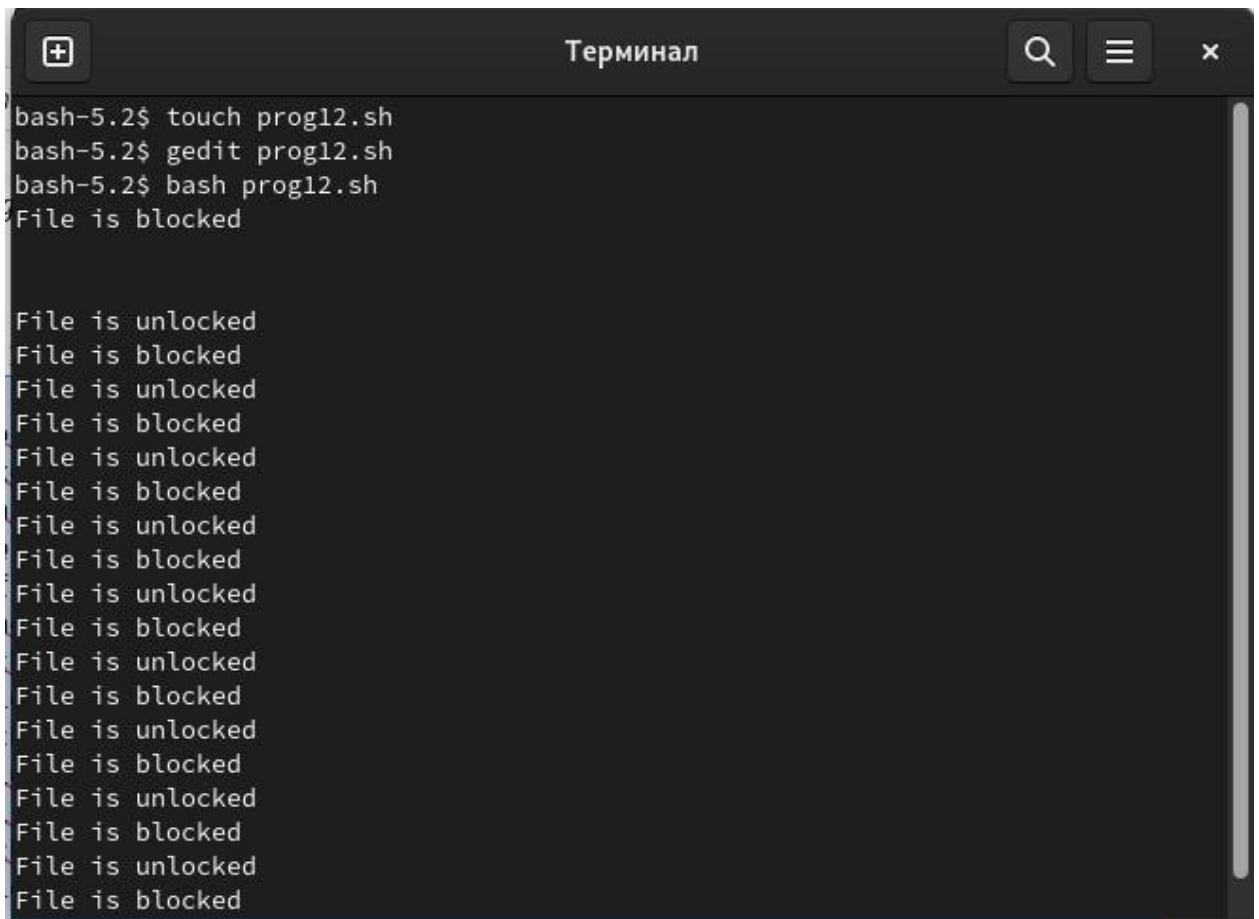
Выполнение:

1. Напишем 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). Запустим 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. Доработаем программу так, чтобы имела возможность взаимодействия трёх и более процессов (рис. 1.1, 1.2).

A screenshot of a terminal window titled "prog12.sh". The window shows a shell script with 16 lines of code. The script uses flock to manage a lockfile. It enters a while loop that checks if the lockfile exists. If it does, it prints "File is blocked" and sleeps for 5 seconds. If it doesn't, it prints "File is unlocked", acquires the lock with flock -u, prints "File is blocked", sleeps for 5 seconds, and releases the lock with flock -u. The script ends with "done".

```
1 #!/bin/bash
2 lockfile="./lock.file"
3 exec {fn}>$lockfile
4 while test -f "$lockfile"
5 do
6 if flock -n ${fn}
7 then
8 echo "File is blocked"
9 sleep 5
10 echo "File is unlocked"
11 flock -u ${fn}
12 else
13 echo "File is blocked"
14 sleep 5
15 fi
16 done
```

Рисунок 1.1. Программа.

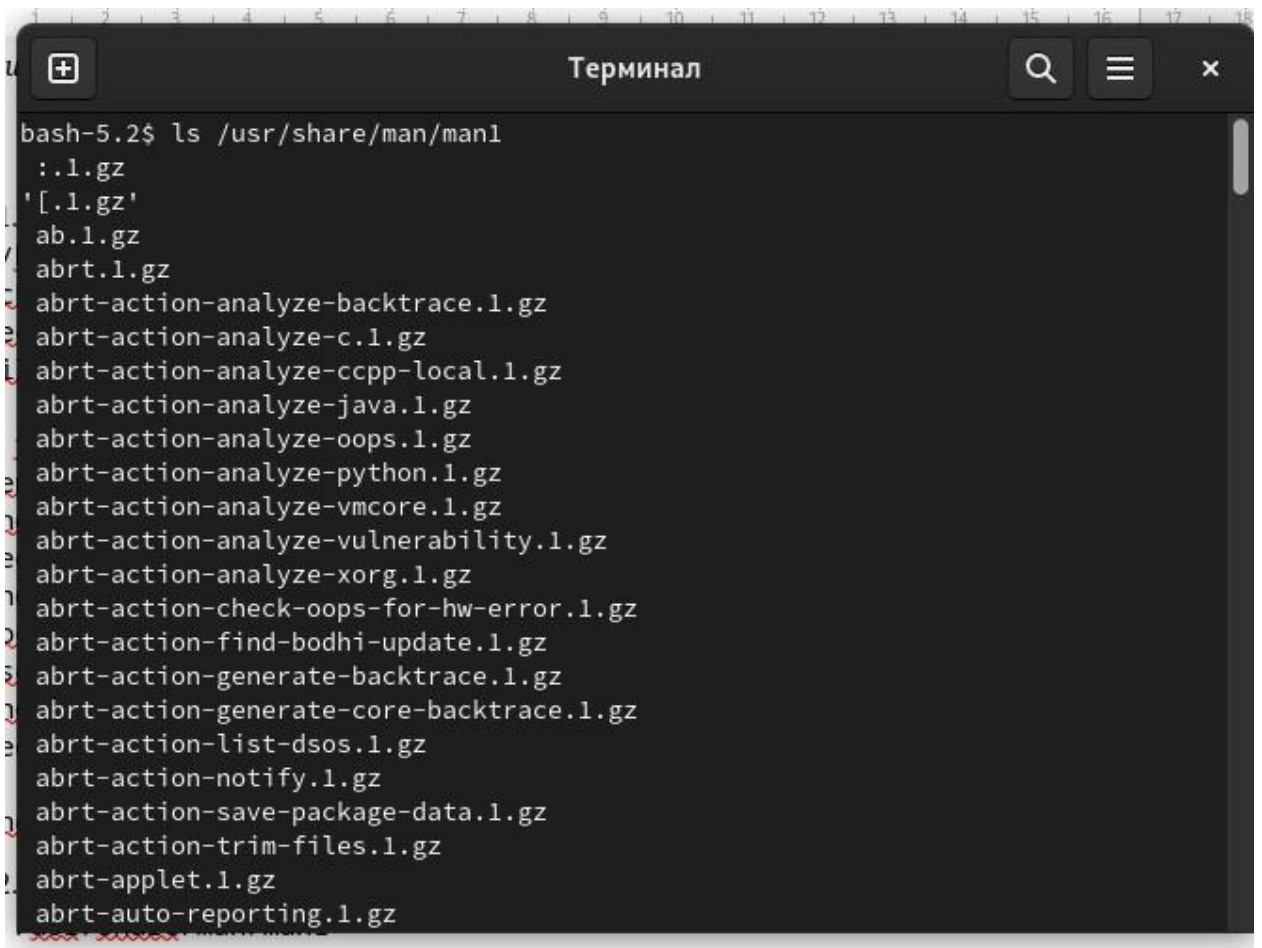


```
bash-5.2$ touch prog12.sh
bash-5.2$ gedit prog12.sh
bash-5.2$ bash prog12.sh
File is blocked

File is unlocked
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
File is unlocked
File is blocked
```

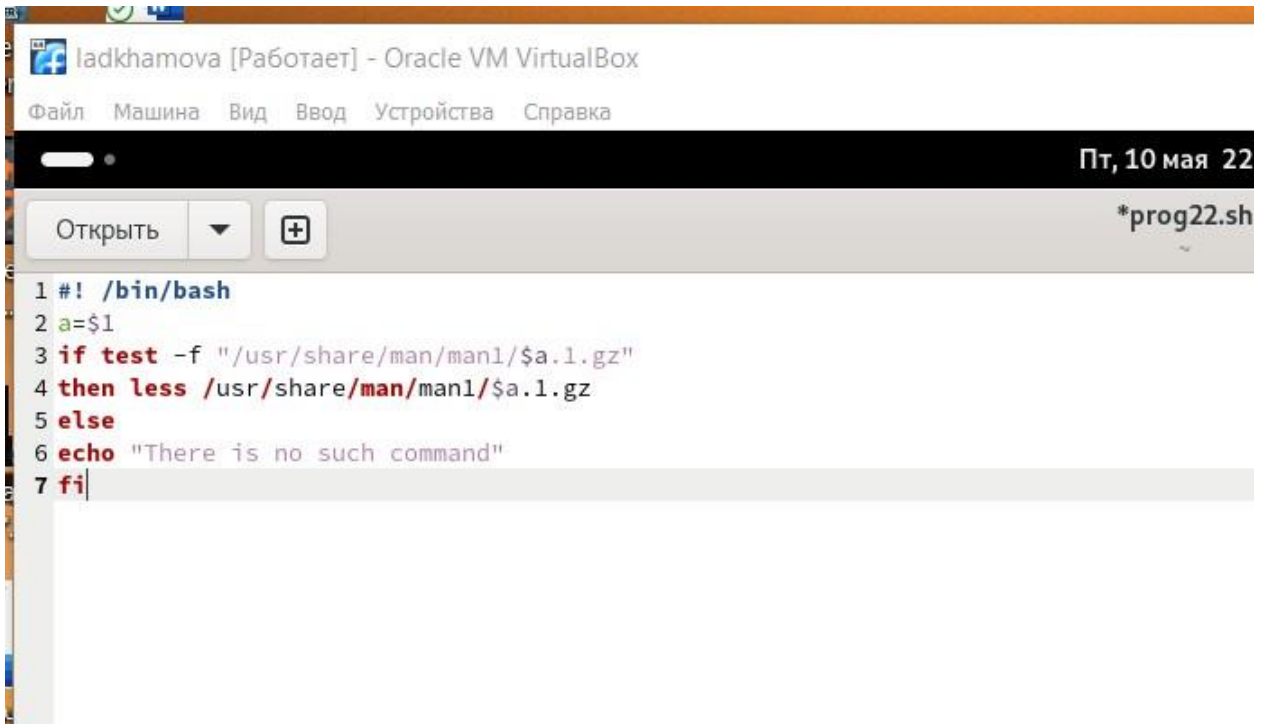
Рисунок 1.2. Результат.

2. Реализуем команду `man` с помощью командного файла. Изучим 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. 2.1, 2.2, 2.3, 2.4).

A terminal window titled "Терминал" (Terminal) with a search icon, a menu icon, and a close icon in the title bar. The terminal shows the command `ls /usr/share/man/man1` and its output, which is a list of 24 gzipped man pages in the `/usr/share/man/man1` directory. The list includes files like `:.1.gz`, `'[.1.gz'`, `ab.1.gz`, `abrt.1.gz`, and various `abrt-action-analyze-...` files, ending with `abrt-auto-reporting.1.gz`.

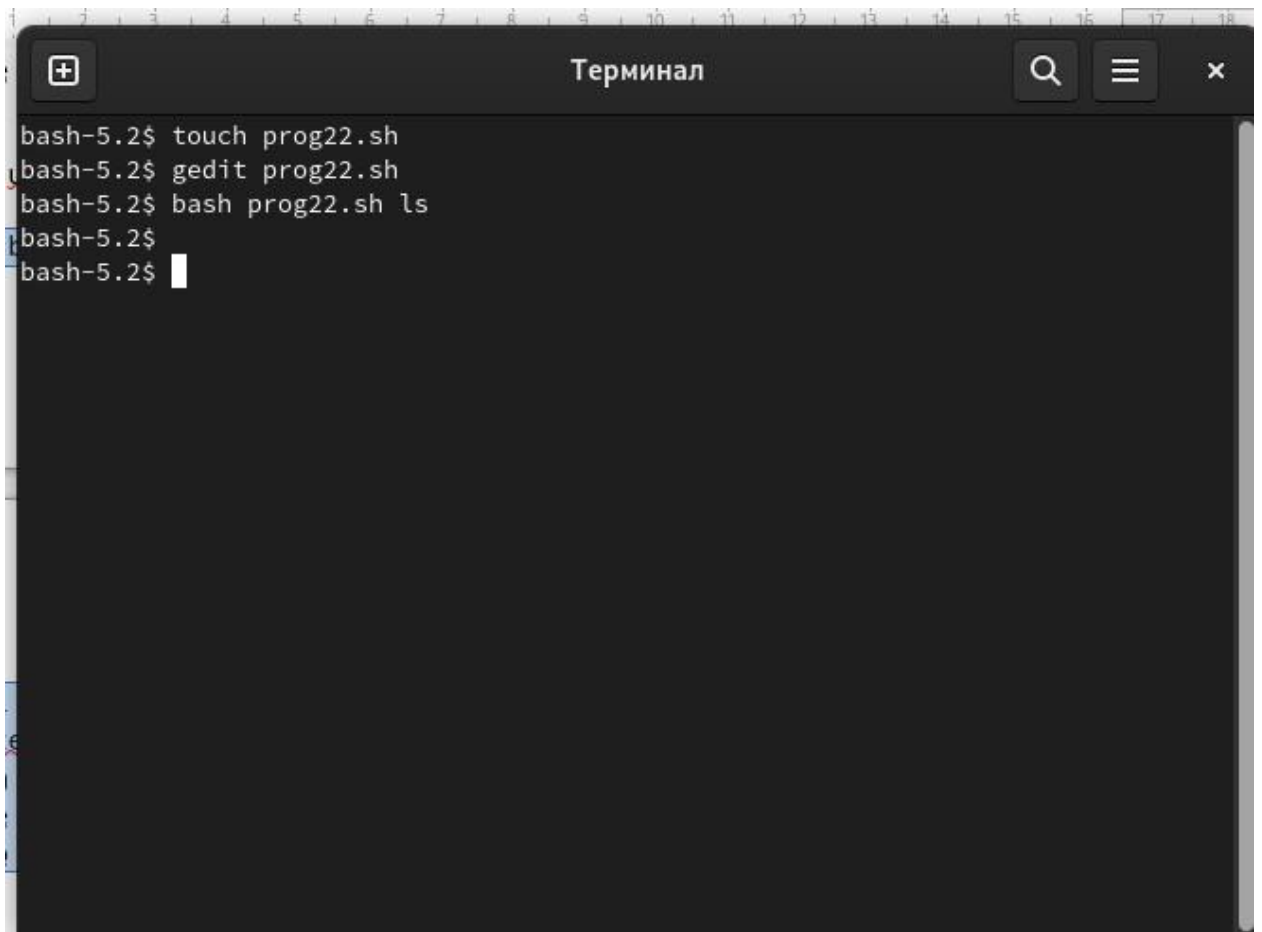
```
bash-5.2$ ls /usr/share/man/man1
:.1.gz
' [.1.gz'
ab.1.gz
abrt.1.gz
abrt-action-analyze-backtrace.1.gz
abrt-action-analyze-c.1.gz
abrt-action-analyze-ccpp-local.1.gz
abrt-action-analyze-java.1.gz
abrt-action-analyze-oops.1.gz
abrt-action-analyze-python.1.gz
abrt-action-analyze-vmcore.1.gz
abrt-action-analyze-vulnerability.1.gz
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-list-dsos.1.gz
abrt-action-notify.1.gz
abrt-action-save-package-data.1.gz
abrt-action-trim-files.1.gz
abrt-applet.1.gz
abrt-auto-reporting.1.gz
```

Рисунок 2.1. Содержание каталога /usr/share/man/man1.

A screenshot of an Oracle VM VirtualBox window titled "ladkhamova [Работает] - Oracle VM VirtualBox". The window shows a terminal window with a menu bar (Файл, Машина, Вид, Ввод, Устройства, Справка) and a title bar. The terminal displays the contents of a script named `*prog22.sh`. The script is a shell script that takes a command as an argument and checks if a corresponding man page exists in `/usr/share/man/man1`. The script code is as follows:

```
1 #! /bin/bash
2 a=$1
3 if test -f "/usr/share/man/man1/$a.1.gz"
4 then less /usr/share/man/man1/$a.1.gz
5 else
6 echo "There is no such command"
7 fi
```

Рисунок 2.2. Программа.

A screenshot of a macOS Terminal window titled "Терминал". The window has a dark background and a light gray title bar with standard macOS window controls (a plus icon on the left, and search, menu, and close icons on the right). A ruler at the top shows line numbers from 2 to 18. The terminal displays a series of commands and their outputs:

```
bash-5.2$ touch prog22.sh
bash-5.2$ gedit prog22.sh
bash-5.2$ bash prog22.sh ls
bash-5.2$
bash-5.2$
```

The cursor is visible at the end of the last line.

bash-5.2\$ touch prog22.sh

bash-5.2\$ gedit prog22.sh

bash-5.2\$ bash prog22.sh ls

bash-5.2\$

bash-5.2\$

Рисунок 2.3. Создание и выполнение командного файла.

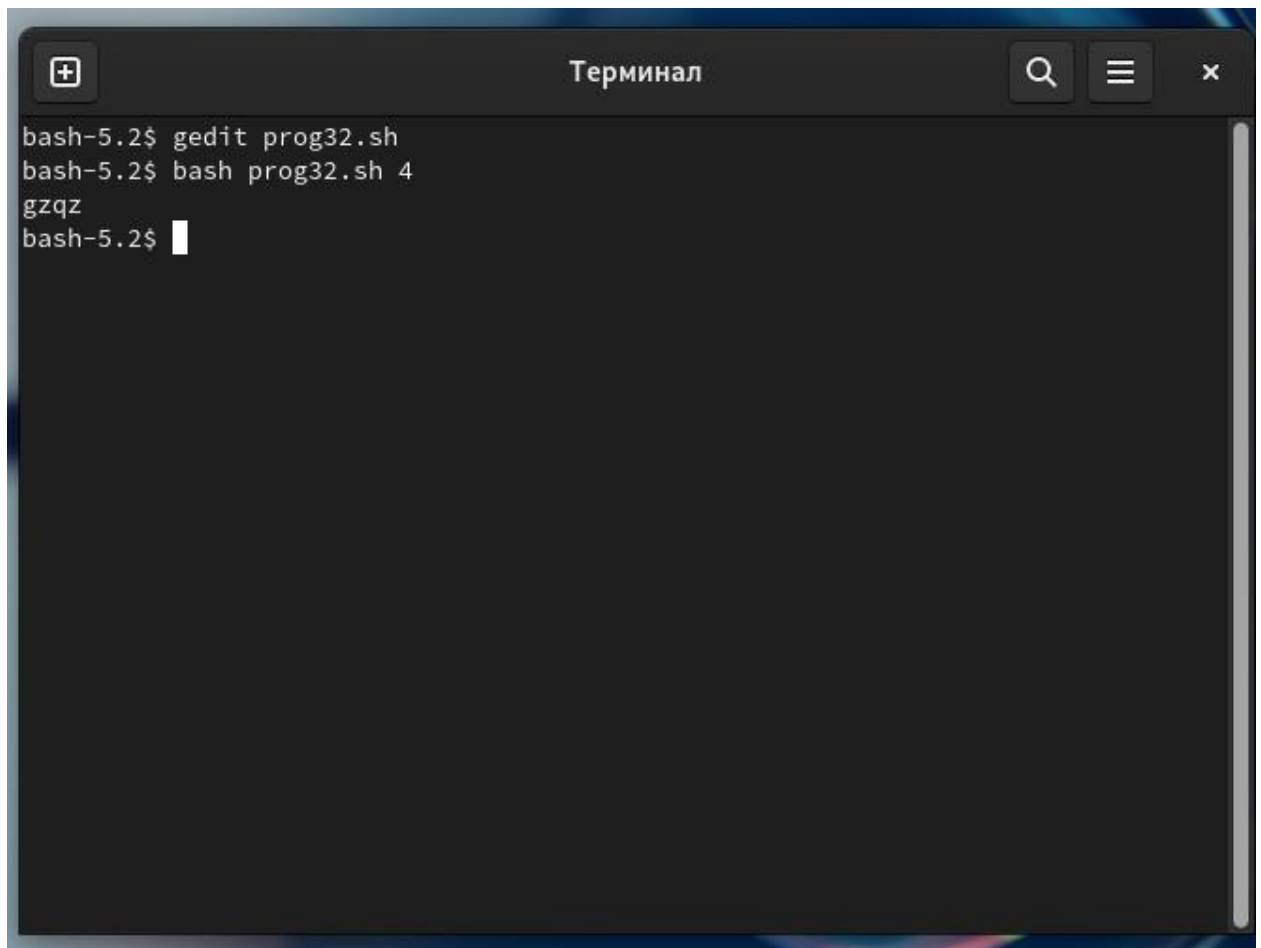


Рисунок 3.2. Выполнение команды.

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. Найдите синтаксическую ошибку в следующей строке: `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. Как объединить (конкатенация) несколько строк в одну?

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. Найдите информацию об утилите seq. Какими иными способами можно реализовать её функционал при программировании на bash?

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. Какой результат даст вычисление выражения $\$((10/3))$?

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

5. Укажите кратко основные отличия командной оболочки zsh от bash.

Отличия командной оболочки zsh от bash: В zsh более быстрое автодополнение для cd с 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. Проверьте, верен ли синтаксис данной конструкции `for ((a=1; a <= LIMIT; a++))`

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

7. Сравните язык bash с какими-либо языками программирования. Какие преимущества у bash по сравнению с ними? Какие недостатки?

Преимущества и недостатки 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.