

Лабораторная работа 14

**Программирование в командном процессоре ОС UNIX.
Расширенное программирование.**

Головина Мария Игоревна

Содержание

1	Цель работы	5
2	Задание	6
3	Теоретическое введение	7
4	Выполнение лабораторной работы	10
5	Ответы на контрольные вопросы	16
6	Выводы	18
	Список литературы	19

Список иллюстраций

4.1	Скрипт №1	10
4.2	Запуск	11
4.3	Запуск	12
4.4	Скрипт №2	13
4.5	Запуск	13
4.6	Результат	14
4.7	Скрипт №3	15
4.8	Запуск	15

Список таблиц

1 Цель работы

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

2 Задание

1. Написать командный файл, реализующий упрощённый механизм семафоров. 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.
2. Реализовать команду `map` с помощью командного файла.
3. Используя встроенную переменную `$RANDOM`, напишите командный файл, генерирующий случайную последовательность букв латинского алфавита.
4. Ответить на контрольные вопросы.

3 Теоретическое введение

Командный процессор (командная оболочка, интерпретатор команд shell) — это программа, позволяющая пользователю взаимодействовать с операционной системой компьютера. В операционных системах типа UNIX/Linux наиболее часто используются следующие реализации командных оболочек: • оболочка Борна (Bourne shell или sh) — стандартная командная оболочка UNIX/Linux, содержащая базовый, но при этом полный набор функций; • C-оболочка (или csh) — надстройка на оболочке Борна, использующая C-подобный синтаксис команд с возможностью сохранения истории выполнения команд; • оболочка Корна (или ksh) — напоминает оболочку C, но операторы управления программой совместимы с операторами оболочки Борна; • BASH — сокращение от Bourne Again Shell (опять оболочка Борна), в основе своей совмещает свойства оболочек C и Корна (разработка компании Free Software Foundation). POSIX (Portable Operating System Interface for Computer Environments) — набор стандартов описания интерфейсов взаимодействия операционной системы и прикладных программ. Стандарты POSIX разработаны комитетом IEEE (Institute of Electrical and Electronics Engineers) для обеспечения совместимости различных UNIX/Linux-подобных операционных систем и переносимости прикладных программ на уровне исходного кода. POSIX-совместимые оболочки разработаны на базе оболочки Корна.

Командный процессор `bash` обеспечивает возможность использования переменных типа строка символов. Имена переменных могут быть выбраны пользователем. Пользователь имеет возможность присвоить переменной значение некоторой строки символов.

Оболочка `bash` поддерживает встроенные арифметические функции. Команда `let` является показателем того, что последующие аргументы представляют собой выражение, подлежащее вычислению. Простейшее выражение — это единичный терм (`term`), обычно целочисленный.

Целые числа можно записывать как последовательность цифр или в любом базовом формате типа `radix#number`, где `radix` (основание системы счисления) — любое число не более 26. Для большинства команд используются следующие основания систем исчисления: 2 (двоичная), 8 (восьмеричная) и 16 (шестнадцатеричная). Простейшими математическими выражениями являются сложение (+), вычитание (-), умножение (*), целочисленное деление (/) и целочисленный остаток от деления (%).

Команда `let` берет два операнда и присваивает их переменной. Положительным моментом команды `let` можно считать то, что для идентификации переменной ей не нужен знак доллара.

Последовательность команд может быть помещена в текстовый файл. Такой файл называется командным.

При вызове командного файла на выполнение параметры ему могут быть переданы точно таким же образом, как и выполняемой программе. С точки зрения командного файла эти параметры являются позиционными. Символ `$` является метасимволом командного процессора. Он используется, в частности, для ссылки на параметры, точнее, для получения их значений в командном файле.

Весьма необходимой при программировании является команда `getopts`, которая осуществляет синтаксический анализ командной строки, выделяя флаги, и используется для объявления переменных.

Флаги — это опции командной строки, обычно помеченные знаком минус.

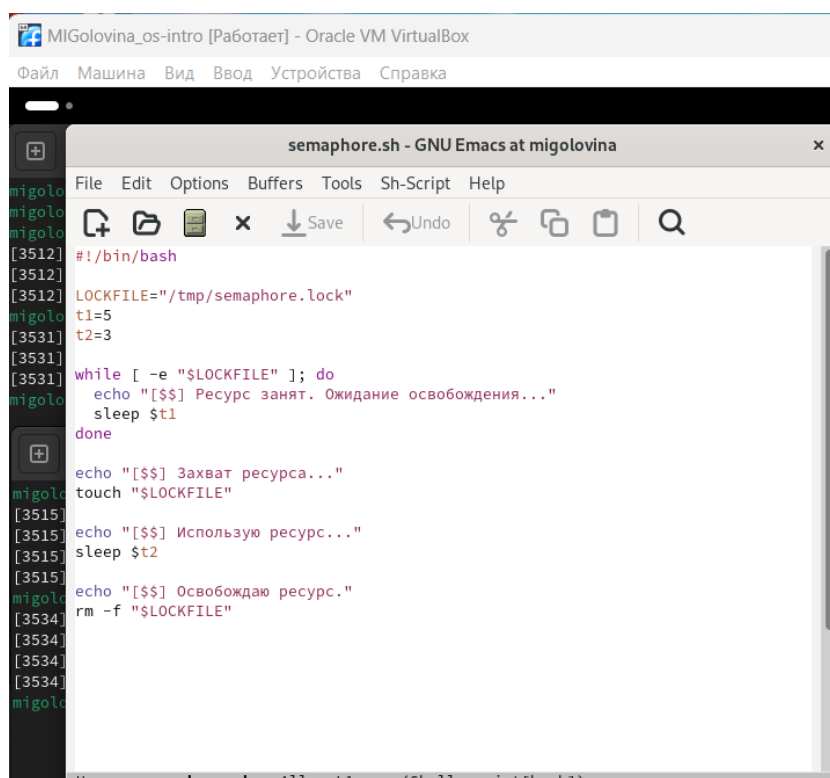
Строка опций `option-string` — это список возможных букв и чисел соответствующего флага. Если ожидается, что некоторый флаг будет сопровождаться некоторым аргументом, то за символом, обозначающим этот флаг, должно следовать двоеточие.

Часто бывает необходимо обеспечить проведение каких-либо действий циклически и управление дальнейшими действиями в зависимости от результатов проверки некоторого условия. Для решения подобных задач язык программирования `bash` предоставляет возможность использовать такие управляющие конструкции, как `for`, `case`, `if` и `while`.

Более подробно о Linux см. в [1-7]

4 Выполнение лабораторной работы

1. Написала командный файл, реализующий упрощённый механизм семафоров (рис. 4.1).



```
semaphore.sh - GNU Emacs at migolovina
File Edit Options Buffers Tools Sh-Script Help
[3512] #!/bin/bash
[3512] LOCKFILE="/tmp/semaphore.lock"
[3512] t1=5
[3531] t2=3
[3531] while [ -e "$LOCKFILE" ]; do
[3531]   echo "[$$] Ресурс занят. Ожидание освобождения..."
[3512]   sleep $t1
[3531] done
[3515] echo "[$$] Захват ресурса..."
[3515] touch "$LOCKFILE"
[3515] echo "[$$] Использую ресурс..."
[3515] sleep $t2
[3515] echo "[$$] Освобождаю ресурс."
[3534] rm -f "$LOCKFILE"
[3534]
[3534]
[3534]
```

Рис. 4.1: Скрипт №1

2. Запустила скрипт №1 (рис. 4.2).

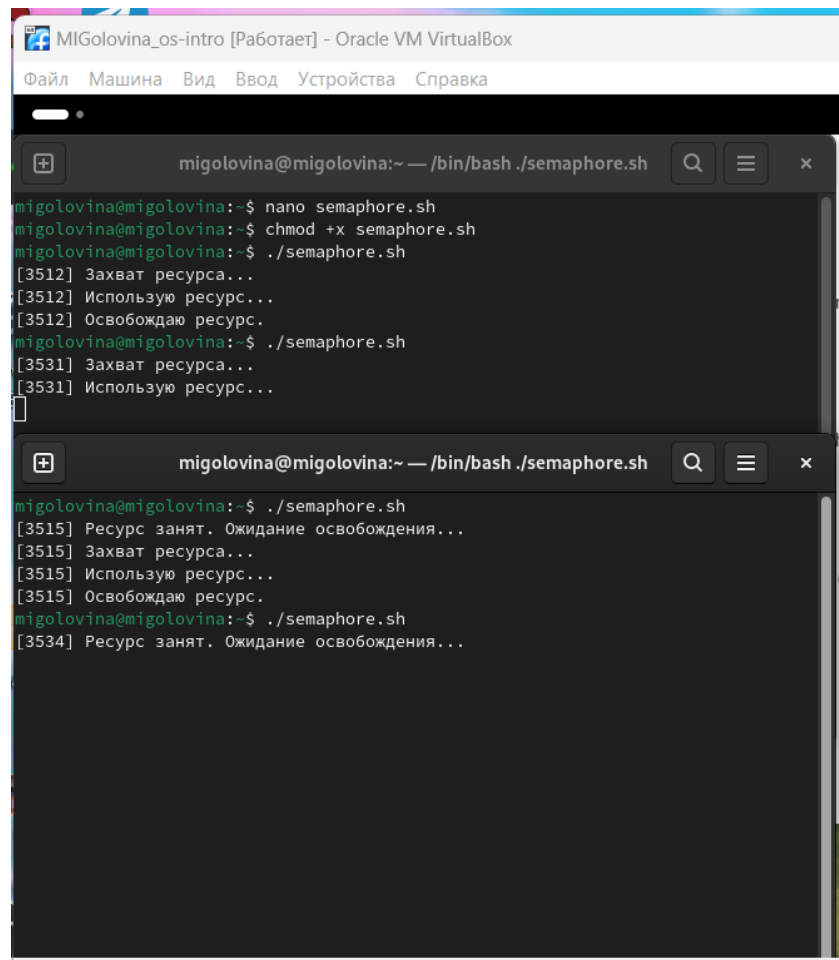
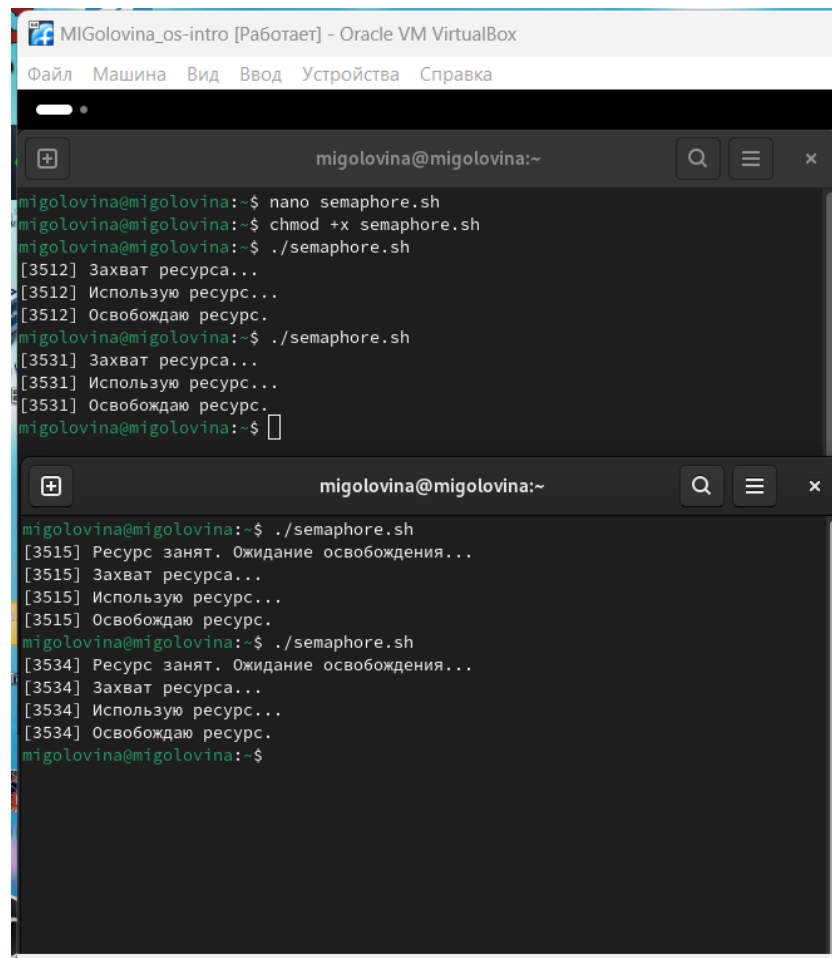


Рис. 4.2: Запуск



```
MIGolovina_os-intro [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка

migolovina@migolovina:~
[+]  migolovina@migolovina:~  🔍  ☰  ✕

migolovina@migolovina:~$ nano semaphore.sh
migolovina@migolovina:~$ chmod +x semaphore.sh
migolovina@migolovina:~$ ./semaphore.sh
[3512] Захват ресурса...
[3512] Использую ресурс...
[3512] Освобождаю ресурс.
migolovina@migolovina:~$ ./semaphore.sh
[3531] Захват ресурса...
[3531] Использую ресурс...
[3531] Освобождаю ресурс.
migolovina@migolovina:~$

migolovina@migolovina:~
[+]  migolovina@migolovina:~  🔍  ☰  ✕

migolovina@migolovina:~$ ./semaphore.sh
[3515] Ресурс занят. Ожидание освобождения...
[3515] Захват ресурса...
[3515] Использую ресурс...
[3515] Освобождаю ресурс.
migolovina@migolovina:~$ ./semaphore.sh
[3534] Ресурс занят. Ожидание освобождения...
[3534] Захват ресурса...
[3534] Использую ресурс...
[3534] Освобождаю ресурс.
migolovina@migolovina:~$
```

Рис. 4.3: Запуск

3. Реализовала команду `map` с помощью командного файла (рис. 4.4).

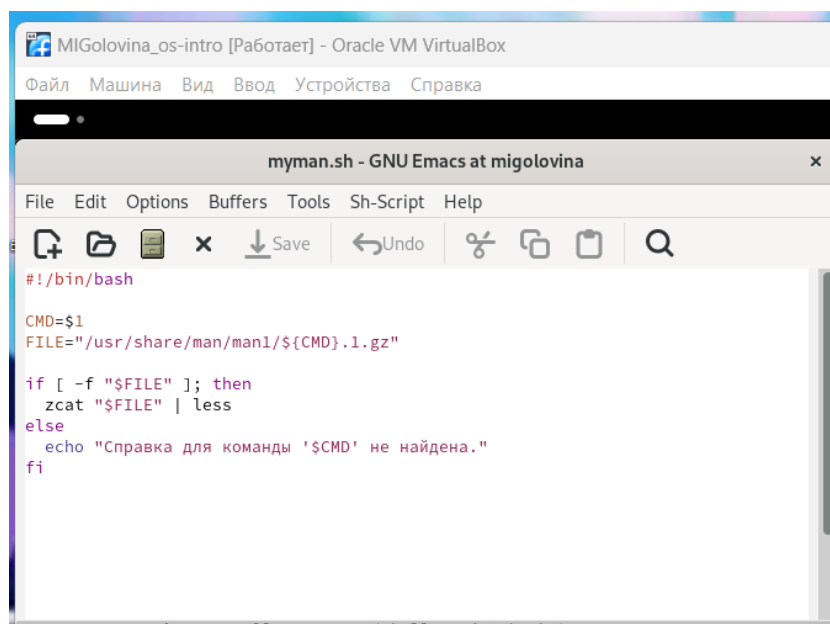


Рис. 4.4: Скрипт №2

4. Запустила скрипт №2 (рис. 4.5).

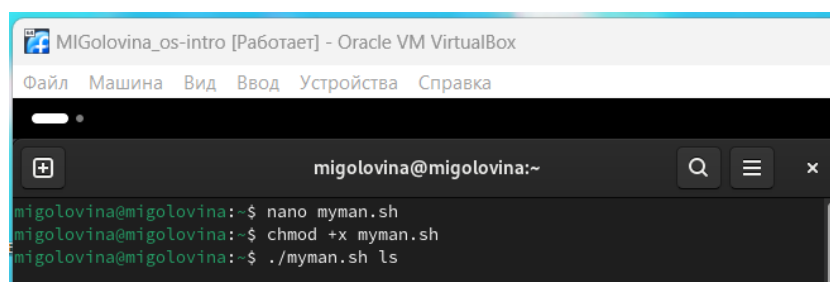
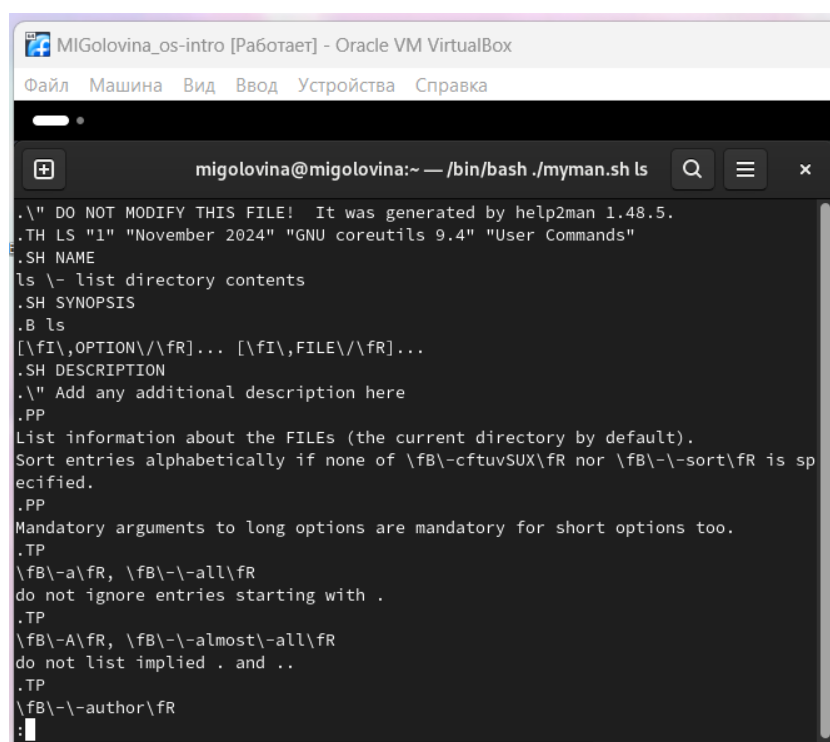


Рис. 4.5: Запуск

5. Результат выполнения скрипта №2 (рис. 4.6).



```
MIIGolovina_os-intro [Работает] - Oracle VM VirtualBox
Файл  Машина  Вид  Ввод  Устройства  Справка

migolovina@migolovina:~ — /bin/bash ./myman.sh ls

.\ " DO NOT MODIFY THIS FILE! It was generated by help2man 1.48.5.
.TH LS "1" "November 2024" "GNU coreutils 9.4" "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 sp
ecified.
.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
:
```

Рис. 4.6: Результат

- Используя встроенную переменную \$RANDOM, написала командный файл, генерирующий случайную последовательность букв латинского алфавита (рис. 4.7).

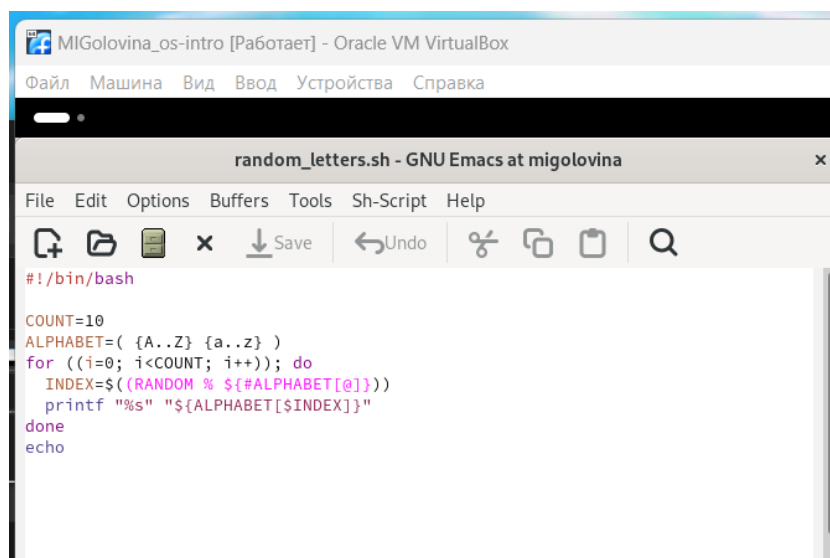


Рис. 4.7: Скрипт №3

8. Запустила скрипт №3 (рис. 4.8).

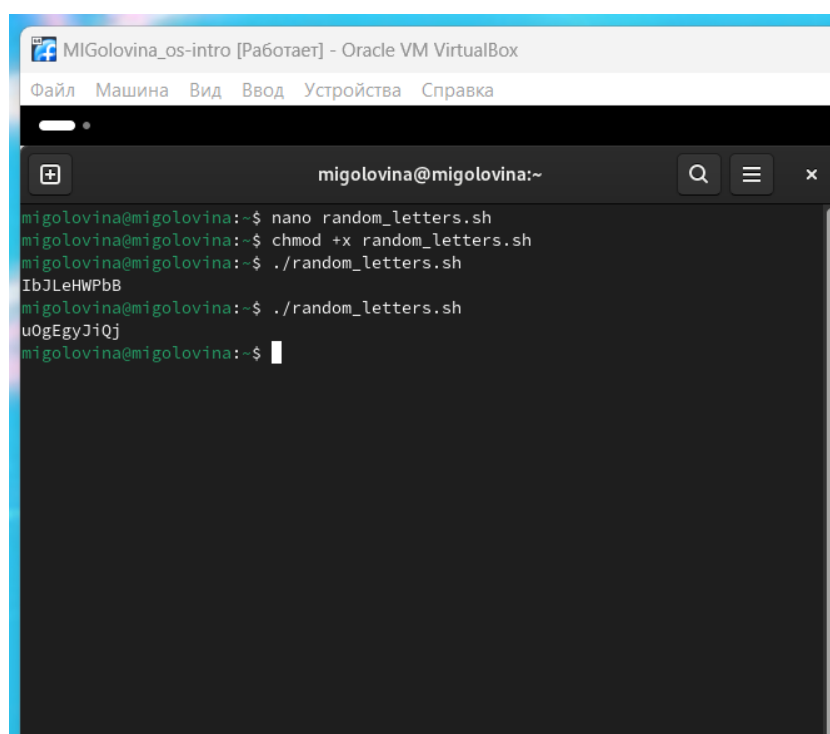


Рис. 4.8: Запуск

5 Ответы на контрольные вопросы

1. Найдите синтаксическую ошибку в следующей строке: `while [$1 !=“exit”]`.

\$1. Так же между скобками должны быть пробелы. В противном случае скобки и рядом стоящие символы будут восприниматься как одно целое

2. Как объединить (конкатенация) несколько строк в одну?

```
migolovina@migolovina:~$ cat file.txt | xargs | sed -e 's/\n /./g'
```

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

`Seq` - выдает последовательность чисел. Реализовать ее функционал можно командой `for n in {1..5} do done`

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

3

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

`Zsh` очень сильно упрощает работу. Но существуют различия. Например, в `zsh` после `for` обязательно вставлять пробел, нумерация массивов в `zsh` начинается с 1 (что не особо удобно на самом деле). Если вы собираетесь писать скрипт, который легко будет запускать множество разработчиков, то я рекомендую `Bash`. Если скрипты вам не нужны - `Zsh` (более простая работа с файлами, например).

6. Проверьте, верен ли синтаксис данной конструкции `for ((a=1; a <= LIMIT; a++))`

верен

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

`Bash` позволяет очень легко работать с файловой системой без лишних конструкций (в отличие от обычного языка программирования). Но относительно обычных языков программирования `bash` очень сжат. Тот же `C` имеет гораздо более широкие возможности для разработчика.

6 Выводы

Я изучила основы программирования в оболочке ОС UNIX. Научилась писать более сложные командные файлы с использованием логических управляющих конструкций и циклов.

Список литературы

1. Dash, P. Getting Started with Oracle VM VirtualBox / P. Dash. – Packt Publishing Ltd, 2013. – 86 сс.
2. Colvin, H. VirtualBox: An Ultimate Guide Book on Virtualization with VirtualBox. VirtualBox / H. Colvin. – CreateSpace Independent Publishing Platform, 2015. – 70 сс.
3. Vugt, S. van. Red Hat RHCSA/RHCE 7 cert guide : Red Hat Enterprise Linux 7 (EX200 and EX300) : Certification Guide. Red Hat RHCSA/RHCE 7 cert guide / S. van Vugt. – Pearson IT Certification, 2016. – 1008 сс.
4. Робачевский, А. Операционная система UNIX / А. Робачевский, С. Немнюгин, О. Стесик. – 2-е изд. – Санкт-Петербург : БХВ-Петербург, 2010. – 656 сс.
5. Немет, Э. Unix и Linux: руководство системного администратора. Unix и Linux / Э. Немет, Г. Снайдер, Т.Р. Хейн, Б. Уэйли. – 4-е изд. – Вильямс, 2014. – 1312 сс.
6. Колисниченко, Д.Н. Самоучитель системного администратора Linux : Системный администратор / Д.Н. Колисниченко. – Санкт-Петербург : БХВ-Петербург, 2011. – 544 сс.
7. Robbins, A. Bash Pocket Reference / A. Robbins. – O'Reilly Media, 2016. – 156 сс.