

Лабораторная работа №12

Программирование в командном процессоре ОС UNIX. Расширенное
программирование

Бабенко Артём Сергеевич

Содержание

Цель работы	3
Теоретическое введение	4
Ход работы	5
Выводы	12
Контрольные вопросы	13

Цель работы

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

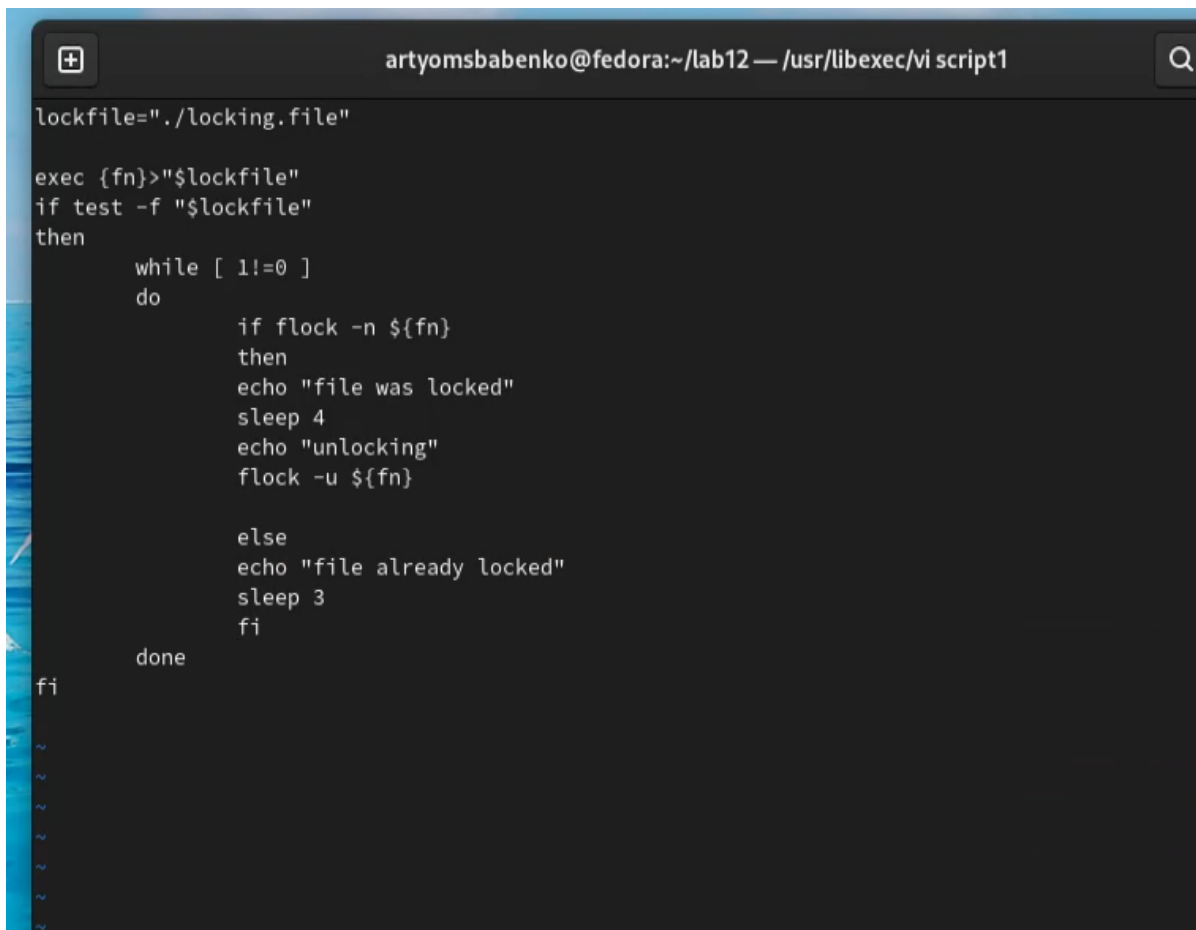
Теоретическое введение

Командные процессоры или оболочки – это программы, позволяющие пользователю взаимодействовать с компьютером. Их можно рассматривать как настоящие интерпретируемые языки, которые воспринимают команды пользователя и обрабатывают их. Поэтому командные процессоры также называют интерпретаторами команд. На языках оболочек можно писать программы и выполнять их подобно любым другим программам. UNIX обладает большим количеством оболочек. Наиболее популярными являются следующие четыре оболочки:

- оболочка Борна – первоначальная командная оболочка UNIX: базовый, но полный набор функций;
- C-оболочка – добавка университета Беркли к коллекции оболочек: она надстраивается над оболочкой Борна, используя C-подобный синтаксис команд, и сохраняет историю выполненных команд;
- оболочка Корна – 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).

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). Запустить командный файл в одном виртуальном терминале в фоновом режиме, перенаправив его вывод в другой, в котором также запущен этот файл, но не фоновом, а в привилегированном режиме (рис.1,2).

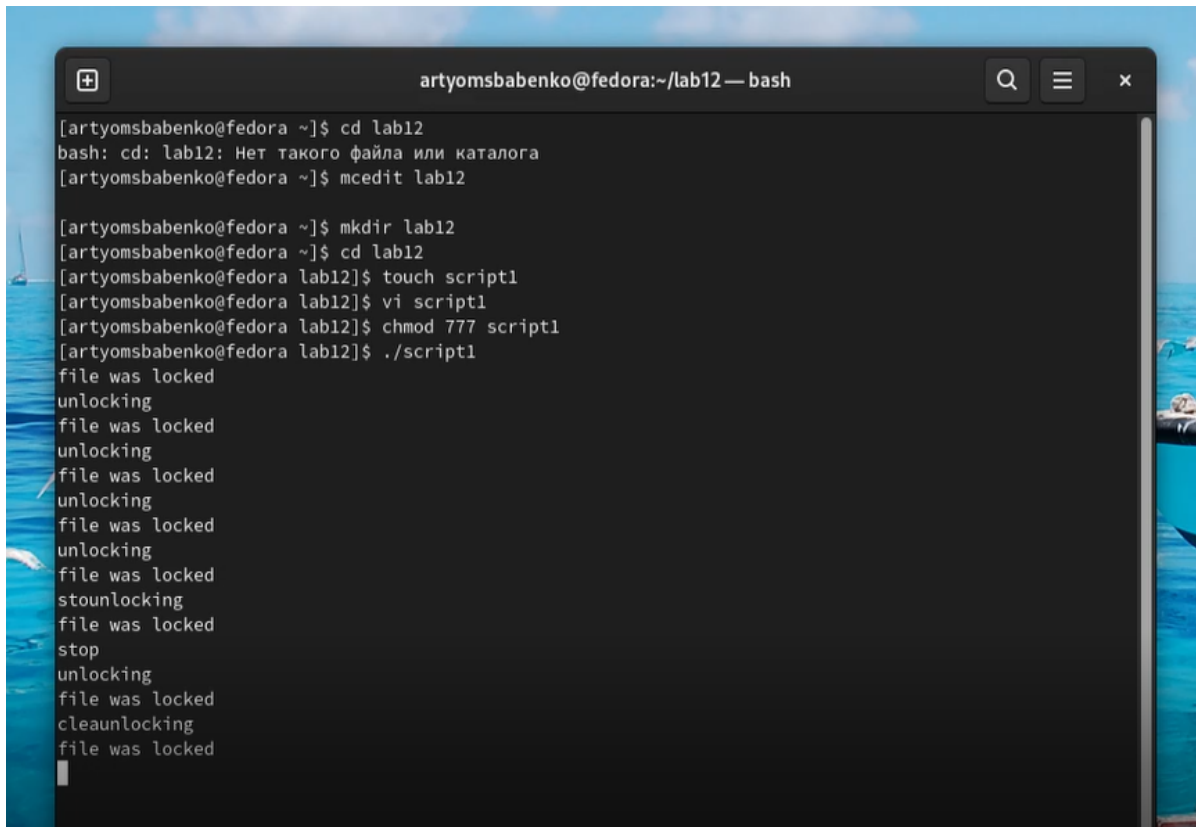
A screenshot of a terminal window with a dark background. The window title bar shows the user 'artyomsbabenko' at 'fedora:~/lab12' with the command '/usr/libexec/vi script1'. The terminal displays a shell script for file locking. The script starts with 'lockfile=./locking.file', followed by 'exec {fn}>"\$lockfile"'. It then enters an 'if test -f "\$lockfile"' block. Inside the 'if' block, there is a 'then' section containing a 'while [1!=0]' loop. The loop has a 'do' section with an 'if flock -n \${fn}' condition. If the condition is true, it prints 'file was locked', sleeps for 4 seconds, prints 'unlocking', and unlocks the file with 'flock -u \${fn}'. If the condition is false, it prints 'file already locked', sleeps for 3 seconds, and does nothing. The 'while' loop ends with 'done', and the 'if' block ends with 'fi'. The script continues with 'fi' and several tilde characters '~' on the following lines.

```
lockfile=./locking.file

exec {fn}>"$lockfile"
if test -f "$lockfile"
then
    while [ 1!=0 ]
    do
        if flock -n ${fn}
        then
            echo "file was locked"
            sleep 4
            echo "unlocking"
            flock -u ${fn}

        else
            echo "file already locked"
            sleep 3
            fi
        done
    fi
fi
~
~
~
~
~
~
~
```

Рис.1. Листинг № 1

A terminal window titled 'artyomsbabenko@fedora:~/lab12 — bash' with search, menu, and close icons. The terminal shows the following commands and output:

```
[artyomsbabenko@fedora ~]$ cd lab12
bash: cd: lab12: Нет такого файла или каталога
[artyomsbabenko@fedora ~]$ mcedit lab12

[artyomsbabenko@fedora ~]$ mkdir lab12
[artyomsbabenko@fedora ~]$ cd lab12
[artyomsbabenko@fedora lab12]$ touch script1
[artyomsbabenko@fedora lab12]$ vi script1
[artyomsbabenko@fedora lab12]$ chmod 777 script1
[artyomsbabenko@fedora lab12]$ ./script1
file was locked
unlocking
file was locked
unlocking
file was locked
unlocking
file was locked
unlocking
file was locked
stounlocking
file was locked
stop
unlocking
file was locked
cleaunlocking
file was locked
```

Рис.2. Результат выполнения программы

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. 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.3,4).

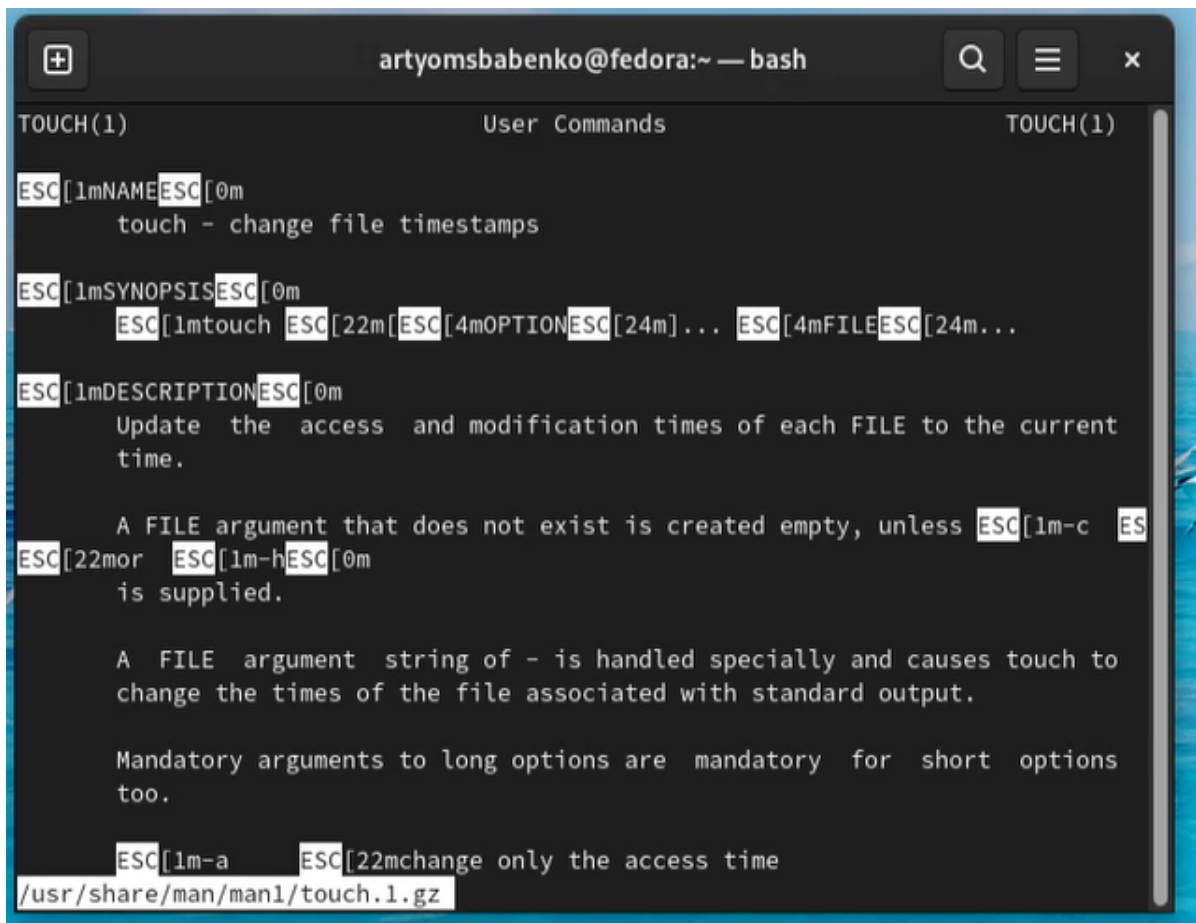
```
lab12_02      [-M--]  2  L: [  1+13  14/ 14]  *(204 / 204b)  <EOF>  [*] [X]
command*""

while getopts :n: opt
do
case $opt in
n)command*"$OPTARG";;
esac
done

if test -f "/usr/share/man/man1/$command.1.gz"
then less /usr/share/man/man1/$command.1.gz
else
echo "No such command"
fi
```

1Помощь 2Сохранить 3Блок 4Замена 5Копия 6Перейти 7Поиск 8Удалить 9МенюМС10Выход

Рис.3. Листинг № 2



```
artyomsbabenko@fedora:~ — bash
TOUCH(1)                                User Commands                                TOUCH(1)

touch - change file timestamps

Update the access and modification times of each FILE to the current
time.

A FILE argument that does not exist is created empty, unless -c or
-h is supplied.

A FILE argument string of - is handled specially and causes touch to
change the times of the file associated with standard output.

Mandatory arguments to long options are mandatory for short options
too.

-a      change only the access time
/usr/share/man/man1/touch.1.gz
```

Рис.4. Результат выполнения программы

- Используя встроенную переменную \$RANDOM, написал командный файл, генерирующий случайную последовательность букв латинского алфавита (рис.3,4).

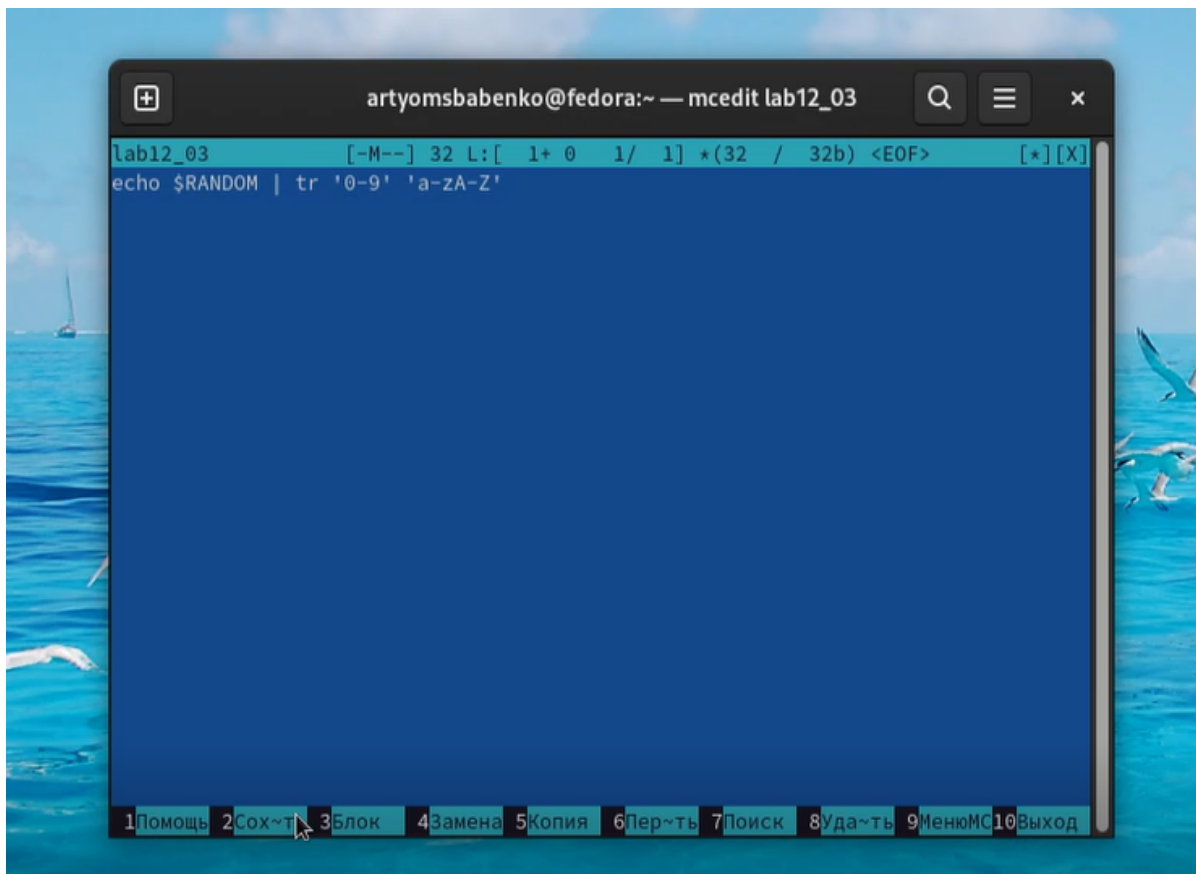
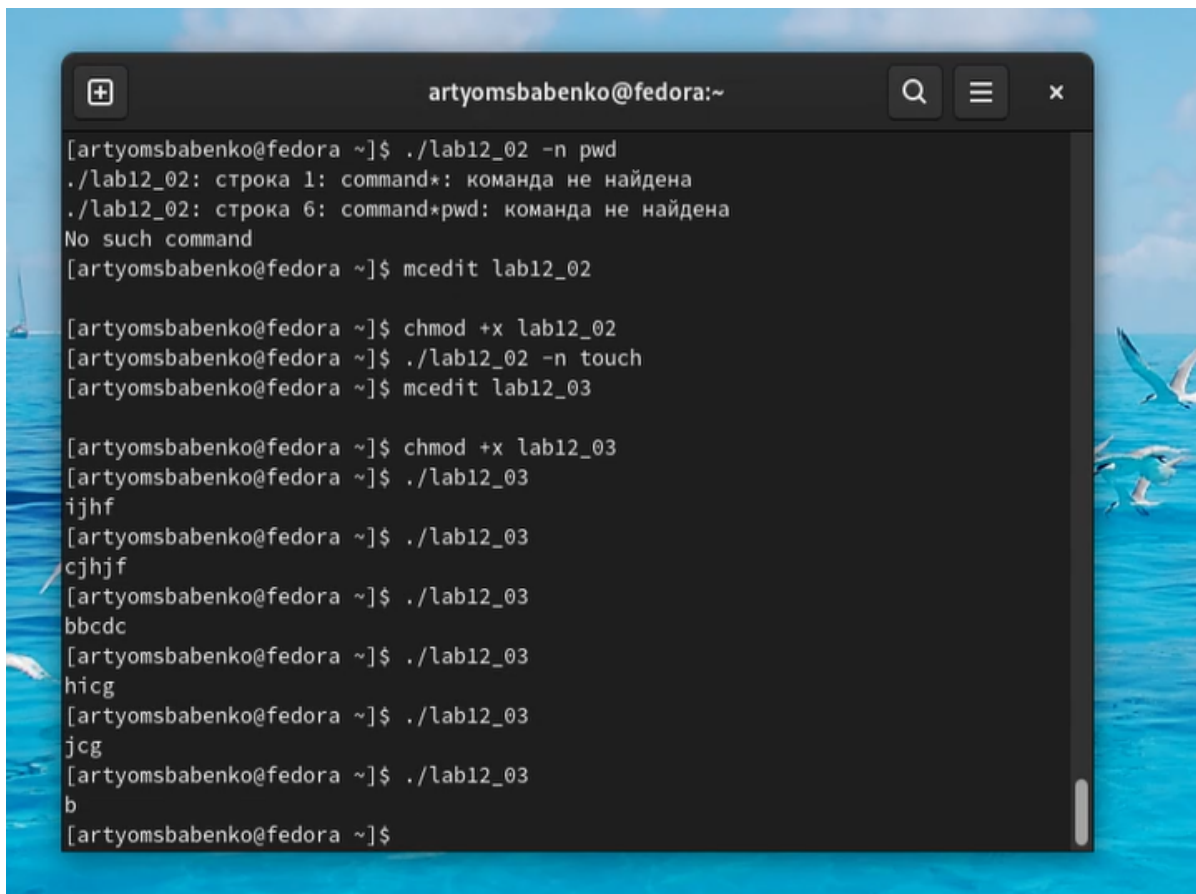


Рис.5. Листинг № 3

A terminal window titled 'artyomsbabenko@fedora:~' with search, menu, and close icons. It shows the execution of a script './lab12_02' which reports errors for 'command*' and 'command*pwd'. Subsequent commands use 'mcedit' to edit 'lab12_02' and 'lab12_03', and 'chmod +x' to make them executable. The script './lab12_03' is then run multiple times, outputting the strings 'ijhfh', 'cjhjfh', 'bbcdc', 'hicg', 'jcg', and 'b' respectively.

```
[artyomsbabenko@fedora ~]$ ./lab12_02 -n pwd
./lab12_02: строка 1: command*: команда не найдена
./lab12_02: строка 6: command*pwd: команда не найдена
No such command
[artyomsbabenko@fedora ~]$ mcedit lab12_02

[artyomsbabenko@fedora ~]$ chmod +x lab12_02
[artyomsbabenko@fedora ~]$ ./lab12_02 -n touch
[artyomsbabenko@fedora ~]$ mcedit lab12_03

[artyomsbabenko@fedora ~]$ chmod +x lab12_03
[artyomsbabenko@fedora ~]$ ./lab12_03
ijhfh
[artyomsbabenko@fedora ~]$ ./lab12_03
cjhjfh
[artyomsbabenko@fedora ~]$ ./lab12_03
bbcdc
[artyomsbabenko@fedora ~]$ ./lab12_03
hicg
[artyomsbabenko@fedora ~]$ ./lab12_03
jcg
[artyomsbabenko@fedora ~]$ ./lab12_03
b
[artyomsbabenko@fedora ~]$
```

Рис.6. Результат выполнения программы

Выводы

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

Контрольные вопросы

2. Путём добавления одной строки в конец другой.
3. Команда `seq` в Linux используется для генерации чисел от первого до последнего шага `increment`. Это очень полезная команда, в которой нам пришлось генерировать список чисел в цикле `while`, `for`, `before`.
4. 3
5. У ZSH много функций, часть из них — лишь незначительные улучшения в Bash, но вот некоторые из основных:

автоматический `cd`: просто введите имя каталога;

рекурсивное расширение пути: например, «`/u/lo/b`» заменяется на «`/usr/local/bin`»;

исправление орфографии и приблизительное завершение: если вы допустили незначительную ошибку при вводе имени каталога, ZSH исправит её за вас;

поддержка плагинов и тем: ZSH включает множество различных фреймворков плагинов;

поддержка плагинов и тем, вероятно, самая крутая функция ZSH.