



## Урок 1

# Знакомство. Цель и содержание курса

Знакомство с Emacs. Минимально жизнеспособный продукт. Ценность MVP. Рынок и конкуренты. Команда. Жизненный цикл ПО.

[Введение](#)

[Установка Emacs](#)

[Open Source](#)

[Команда](#)

[Психологические отношения и ответственность](#)

[Минимальный рабочий прототип — MVP](#)

[Рынок, конкуренты](#)

[Жизненный цикл ПО, проекта, продукта](#)

[Практическое задание](#)

[Дополнительные материалы](#)

[Используемая литература](#)

# Введение

Технически операционная система Linux — это ядро Linux, созданное Линусом Торвальдсом в 1991 году, и окружения операционной системы GNU, разработка которого стартовала в 1983.

Автором GNU был Ричард Столлман, и первой программой из состава этой ОС стал текстовый редактор Emacs. Он ведет историю с времен, когда с компьютерами общались посредством телетайпов, а потом терминалов. Во время перехода на терминалы стал популярен командный редактор TECO. В 1972 Ричард Столлман добавил в TECO возможность подключения макросов, и эта программа стала прототипом Emacs. В 1978 году был создан Multics Emacs на MacLisp, а в 1981 — Gosling Emacs для UNIX. Еще три года спустя Ричард Столлман начал разработку новой реализации Emacs как свободной альтернативы проприетарному Gosling Emacs.

Emacs, созданный Ричардом Столлманом, был написан на C, а плагины — на диалекте языка Lisp — Mocklisp.

На стажировке мы будем разрабатывать клон редактора Emacs, но стек технологий будет немного другим.

Объектно-ориентированный C++ более подходит для функционального текстового редактора. В перспективе планируем добавить поддержку плагинов на Python — это более современно и доступно.

Emacs работает в текстовом и графическом режимах. Наш редактор тоже сможет поддерживать их — если программа запущена в графическом режиме или по **ssh** с туннелированием X11-трафика. Для реализации графического режима будем использовать **Qt**.

Научимся работать над серьезным open source приложением уровня компонента пользовательского окружения ОС. Надеемся, что в перспективе наша реализация Emacs с плагинами на Python войдет в состав современных операционных систем: Linux, FreeBSD, MacOS X, — а также будет доступна для использования в Windows.

## Установка Emacs

Прежде чем разрабатывать клон Emacs, ознакомимся с существующей реализацией в GNU/Linux. Установим Emacs в соответствии с [PPA description](#).

Добавляем **ppa** в систему:

```
sudo add-apt-repository ppa:ubuntu-elisp/ppa
sudo apt-get update
```

```

root@comp2: ~
root@comp2:~# add-apt-repository ppa:ubuntu-elisp/ppa
The Official ;-) Ubuntu Emacs Daily Snapshot PPA.

The packaging metadata used here was written completely from scratch in 2013 by Robert Bruce Park, completely
discarding all legacy cruft associated with the official Emacs package for debian/ubuntu. This means:

* No distropatches. Many of the existing distropatches were backported from trunk, so this shouldn't be a big
deal, however there is a chance that your favorite bug was only ever fixed in a distropatch, and that has no
w regressed in these snapshots.

* Little/no support for installing debian packages of elisp modules. Much of that support exists only in the
form of a distropatch, which is not included here. So for example if you were to `apt-get install yaml-mode`,
you would have to include "(require 'yaml-mode)" in your init, but you wouldn't be able to autoload yaml-mod
e like you may be accustomed to with the stable series. I am working on enabling this, but be warned that it
is currently broken.

* On the plus side, package.el works *excellently* and most of the packages you might want to install should
be available either from MELPA or Marmalade or similar. In general, if you are using these snapshots you will
want to `M-x package-install foo` rather than `apt-get install foo`.
More info: https://launchpad.net/~ubuntu-elisp/+archive/ubuntu/ppa
Press [ENTER] to continue or ctrl-c to cancel adding it

gpg: keyring `/tmp/tmporzv6inlu/secring.gpg' created
gpg: keyring `/tmp/tmporzv6inlu/pubring.gpg' created
gpg: requesting key D62FCE72 from hkp server keyserver.ubuntu.com
gpg: /tmp/tmporzv6inlu/trustdb.gpg: trustdb created
gpg: key D62FCE72: public key "Launchpad PPA for Ubuntu Emacs Lisp" imported
gpg: Total number processed: 1
gpg:             imported: 1   (RSA: 1)
OK
root@comp2:~# apt-get update

```

```

OK
root@comp2:~# apt-get update
Get:1 http://security.ubuntu.com/ubuntu xenial-security InRelease [107 kB]
Get:2 http://security.ubuntu.com/ubuntu xenial-security/main amd64 DEP-11 Metadata [67,7 kB]
Hit:3 https://apt.dockerproject.org/repo ubuntu-xenial InRelease
Get:4 http://security.ubuntu.com/ubuntu xenial-security/main DEP-11 64x64 Icons [68,0 kB]
Get:5 http://security.ubuntu.com/ubuntu xenial-security/universe amd64 DEP-11 Metadata [109 kB]
Get:6 http://security.ubuntu.com/ubuntu xenial-security/universe DEP-11 64x64 Icons [149 kB]
Hit:7 http://ppa.launchpad.net/mrazavi/openvas/ubuntu xenial InRelease
Get:8 http://ppa.launchpad.net/ubuntu-elisp/ppa/ubuntu xenial InRelease [17,5 kB]
Get:9 http://ppa.launchpad.net/ubuntu-elisp/ppa/ubuntu xenial/main amd64 Packages [2 252 B]
Get:10 http://ppa.launchpad.net/ubuntu-elisp/ppa/ubuntu xenial/main i386 Packages [2 248 B]
Get:11 http://ppa.launchpad.net/ubuntu-elisp/ppa/ubuntu xenial/main Translation-en [600 B]
Hit:12 http://ru.archive.ubuntu.com/ubuntu xenial InRelease
Get:13 http://ru.archive.ubuntu.com/ubuntu xenial-updates InRelease [109 kB]
Get:14 http://ru.archive.ubuntu.com/ubuntu xenial-backports InRelease [107 kB]
Get:15 http://ru.archive.ubuntu.com/ubuntu xenial-updates/main amd64 Packages [884 kB]
Get:16 http://ru.archive.ubuntu.com/ubuntu xenial-updates/main i386 Packages [786 kB]
Get:17 http://ru.archive.ubuntu.com/ubuntu xenial-updates/main amd64 DEP-11 Metadata [320 kB]
Get:18 http://ru.archive.ubuntu.com/ubuntu xenial-updates/main DEP-11 64x64 Icons [227 kB]
Get:19 http://ru.archive.ubuntu.com/ubuntu xenial-updates/universe amd64 Packages [707 kB]
Get:20 http://ru.archive.ubuntu.com/ubuntu xenial-updates/universe i386 Packages [649 kB]
Get:21 http://ru.archive.ubuntu.com/ubuntu xenial-updates/universe amd64 DEP-11 Metadata [248 kB]
Get:22 http://ru.archive.ubuntu.com/ubuntu xenial-updates/universe DEP-11 64x64 Icons [337 kB]
Get:23 http://ru.archive.ubuntu.com/ubuntu xenial-updates/multiverse amd64 DEP-11 Metadata [5 968 B]
Get:24 http://ru.archive.ubuntu.com/ubuntu xenial-updates/multiverse DEP-11 64x64 Icons [14,3 kB]
Get:25 http://ru.archive.ubuntu.com/ubuntu xenial-backports/main amd64 DEP-11 Metadata [3 328 B]
Get:26 http://ru.archive.ubuntu.com/ubuntu xenial-backports/universe amd64 DEP-11 Metadata [5 104 B]
Fetched 4 925 kB in 11s (444 kB/s)
AppStream cache update completed, but some metadata was ignored due to errors.
Reading package lists... Done
root@comp2:~#

```

Выбираем версию операционной системы:

#### ▼ Technical details about this PPA

This PPA can be added to your system manually by copying the lines below and adding them to your system's software sources.

Display sources.list entries for:

```
deb http://ppa.launchpad.net/ubuntu-elisp/ppa/ubuntu xenial main
deb-src http://ppa.launchpad.net/ubuntu-elisp/ppa/ubuntu xenial main
```

#### Signing key:

1024R/0D7BAE435ADBC6C3E4918A74

#### Fingerprint:

0D7BAE435ADBC6C3E4918A74

For questions and bugs with software

- Xenial (16.04) ▼
- Choose your Ubuntu version
- Cosmic (18.10)
- Bionic (18.04)
- Artful (17.10)
- Zesty (17.04)
- Yakkety (16.10)
- Xenial (16.04)
- Wily (15.10)
- Vivid (15.04)
- Utopic (14.10)
- Trusty (14.04)
- Saucy (13.10)
- Raring (13.04)
- Quantal (12.10)
- Precise (12.04)

xenial main  
ubuntu xenial main

...this?)

Ubuntu Emacs Lisp.

Сгенерируются строки для **sources.list**:

```
deb http://ppa.launchpad.net/ubuntu-elisp/ppa/ubuntu xenial main
deb-src http://ppa.launchpad.net/ubuntu-elisp/ppa/ubuntu xenial main
```

Добавим эти строки в **/etc/apt/sources.list**:

```
sudo -i
echo deb http://ppa.launchpad.net/ubuntu-elisp/ppa/ubuntu xenial main >>
/etc/apt/sources.list
echo deb-src http://ppa.launchpad.net/ubuntu-elisp/ppa/ubuntu xenial main >>
/etc/apt/sources.list
```

Проверяем:

```
tailf -2 /etc/apt/sources.list
```

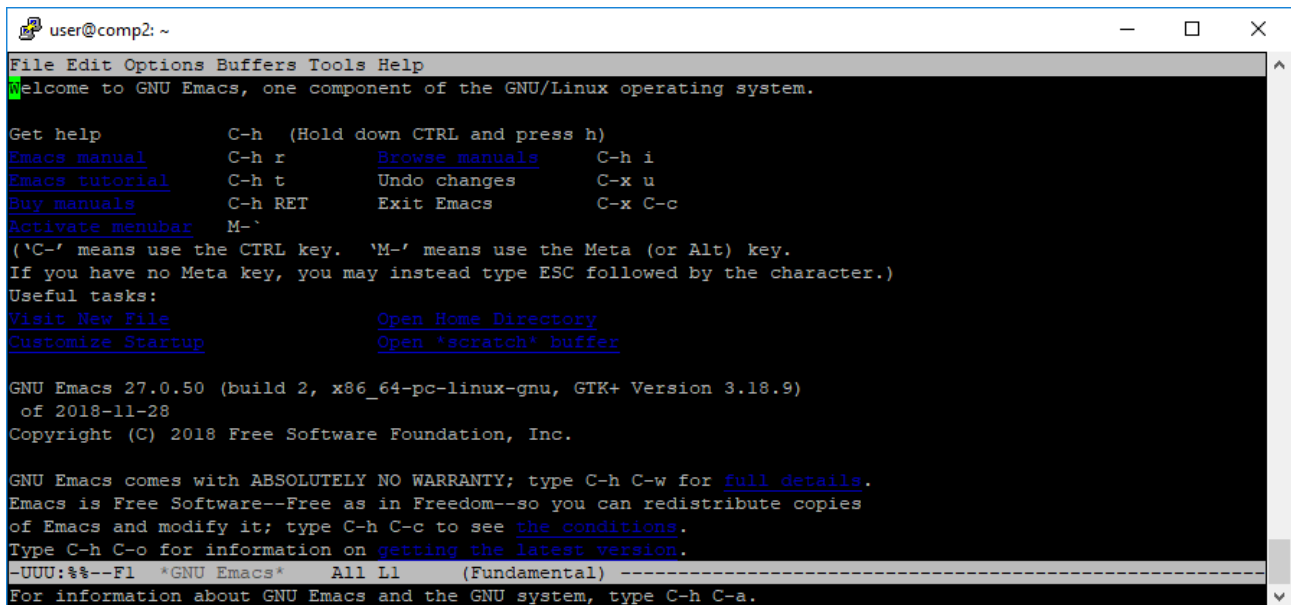
```
root@comp2:~# tailf -2 /etc/apt/sources.list
deb http://ppa.launchpad.net/ubuntu-elisp/ppa/ubuntu xenial main
deb-src http://ppa.launchpad.net/ubuntu-elisp/ppa/ubuntu xenial main
```

Далее устанавливаем:

```
install emacs-snapshot
```

После этого можно запустить Emacs в терминале:

```
emacs
```



```
user@comp2: ~
File Edit Options Buffers Tools Help
Welcome to GNU Emacs, one component of the GNU/Linux operating system.

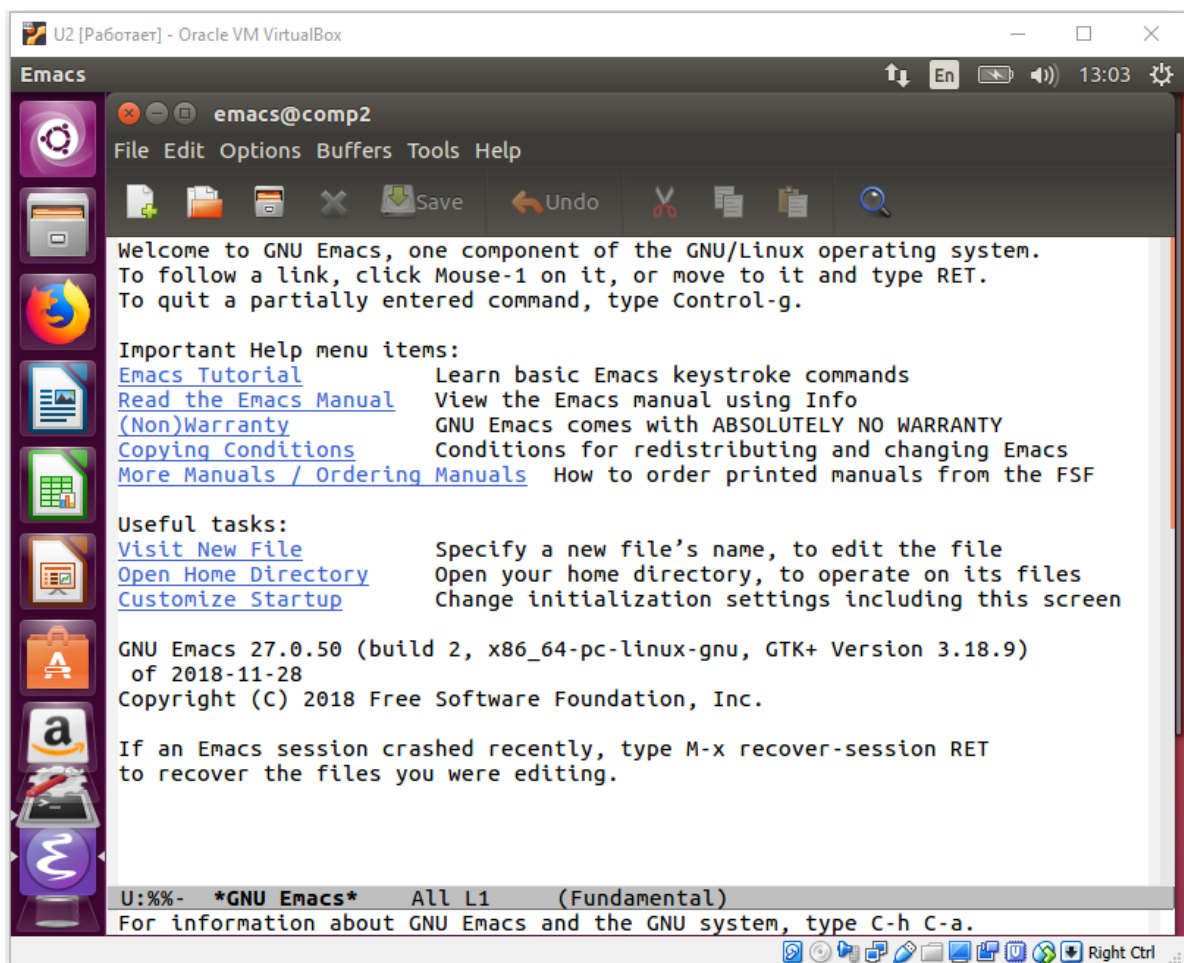
Get help          C-h (Hold down CTRL and press h)
Emacs manual      C-h r          Browse manuals    C-h i
Emacs tutorial    C-h t          Undo changes     C-x u
Buy manuals      C-h RET        Exit Emacs       C-x C-c
Activate menubar M-`

('C-' means use the CTRL key. 'M-' means use the Meta (or Alt) key.
If you have no Meta key, you may instead type ESC followed by the character.)
Useful tasks:
Visit New File           Open Home Directory
Customize Startup        Open *scratch* buffer

GNU Emacs 27.0.50 (build 2, x86_64-pc-linux-gnu, GTK+ Version 3.18.9)
of 2018-11-28
Copyright (C) 2018 Free Software Foundation, Inc.

GNU Emacs comes with ABSOLUTELY NO WARRANTY; type C-h C-w for full details.
Emacs is Free Software--Free as in Freedom--so you can redistribute copies
of Emacs and modify it; type C-h C-c to see the conditions.
Type C-h C-o for information on getting the latest version.
-UUU:%%-F1 *GNU Emacs* All L1 (Fundamental) -----
For information about GNU Emacs and the GNU system, type C-h C-a.
```

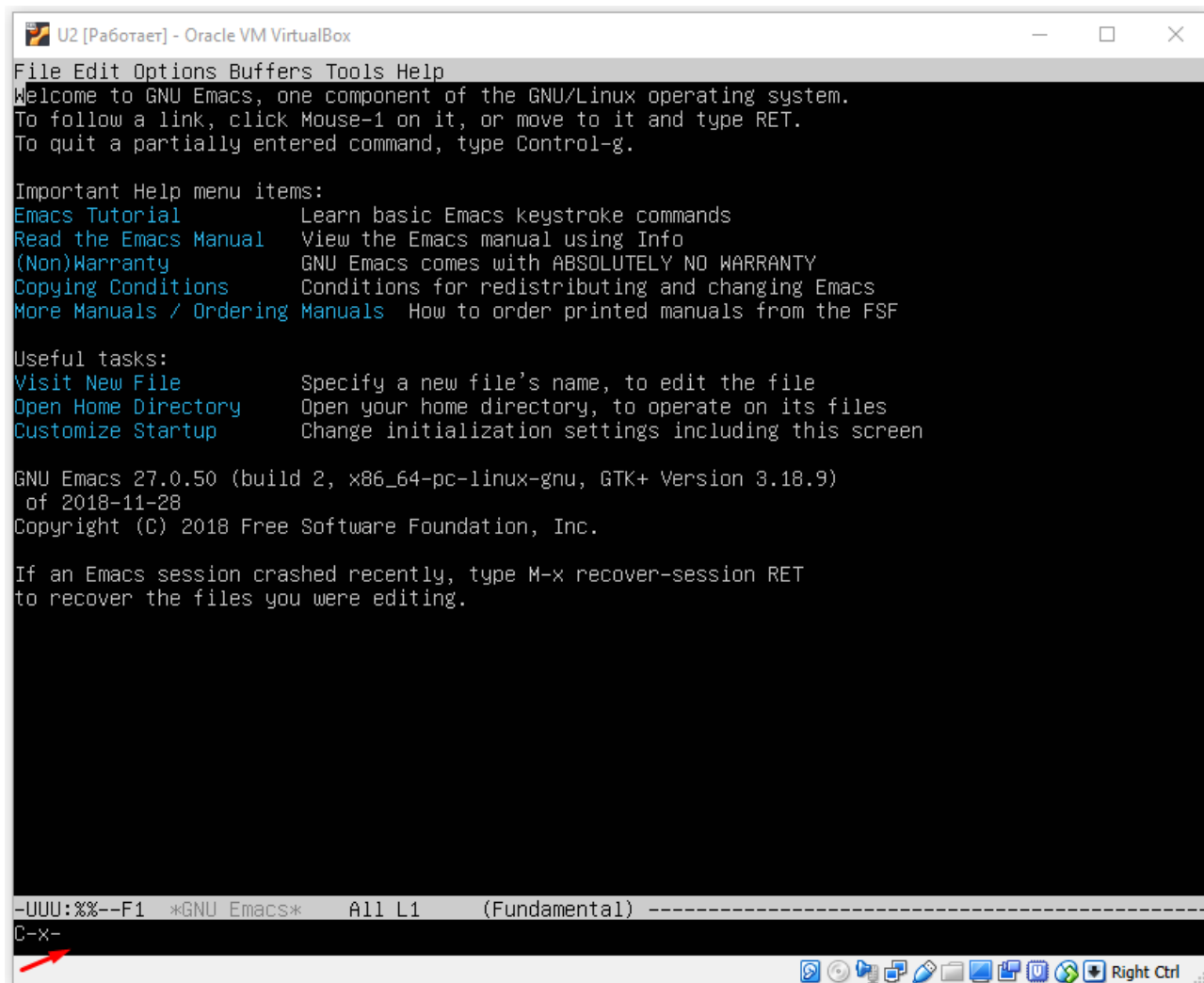
Так же командой в режиме **X-server** (или при подключении по **ssh** с туннелированием X11-трафика) запустится графическая версия программы:



В текстовом и графическом режимах редактор одинаковый, но в графическом есть управление мышкой.

Гиперссылки активны: можно зайти в **Emacs Tutorial** или почитать **Emacs Manual**.

Разберемся, как отсюда выбраться.



Ctrl-x (в документации Emacs отображается как C-x) переведет в режим команд.

Теперь можно нажать Ctrl-c.

Чтобы не выходить, а ознакомиться с tutorialом на русском языке, нажимаем Ctrl-u, а после этого — Ctrl-h.

```
U2 [Пагодаер] - Oracle VM VirtualBox
File Edit Options Buffers Tools Help
Welcome to GNU Emacs, one component of the GNU/Linux operating system.
To follow a link, click Mouse-1 on it, or move to it and type RET.
To quit a partially entered command, type Control-g.

Important Help menu items:
Emacs Tutorial      Learn basic Emacs keystroke commands
Read the Emacs Manual  View the Emacs manual using Info
(Non)Warranty       GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Conditions   Conditions for redistributing and changing Emacs
More Manuals / Ordering Manuals  How to order printed manuals from the FSF

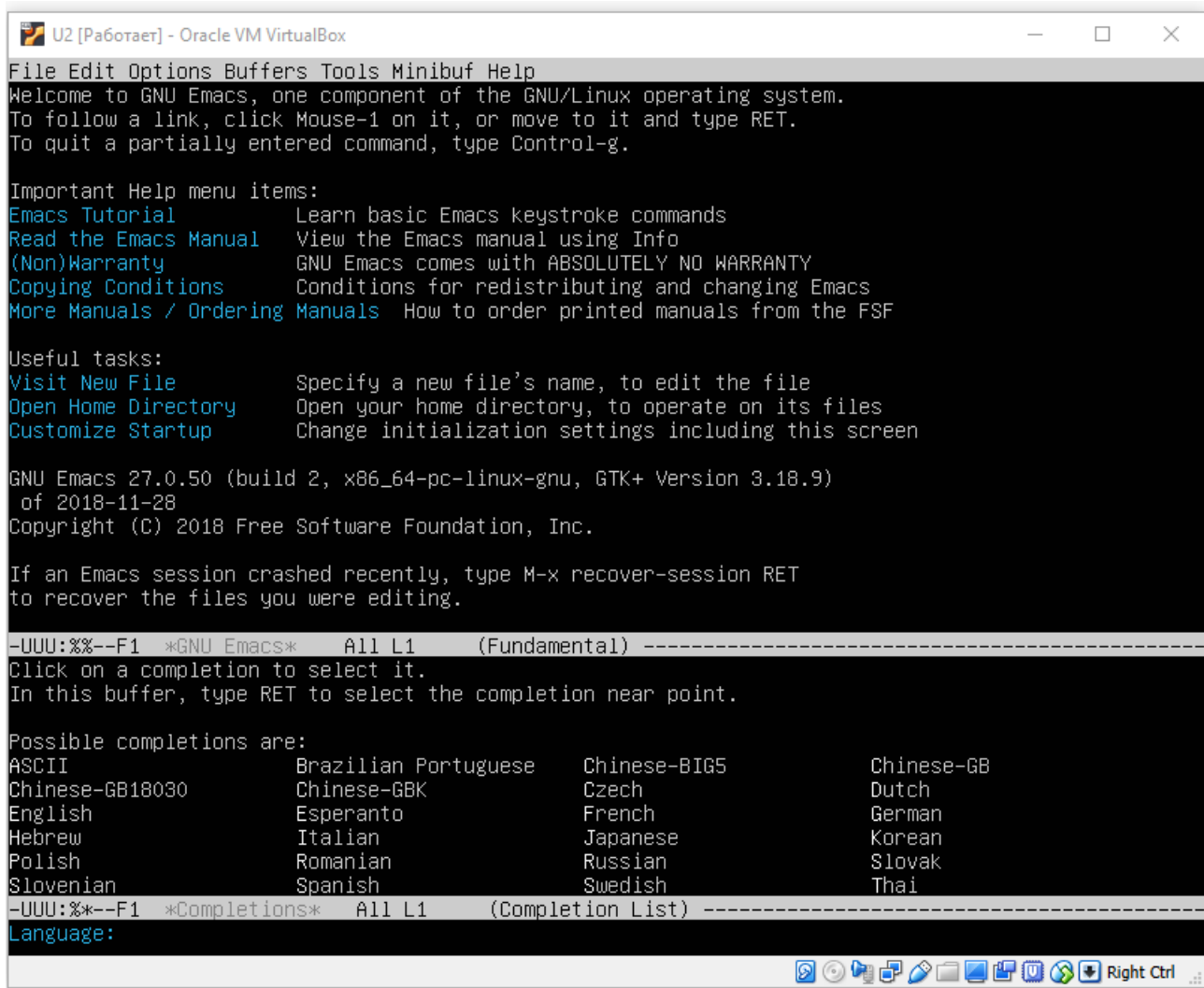
Useful tasks:
Visit New File       Specify a new file's name, to edit the file
Open Home Directory  Open your home directory, to operate on its files
Customize Startup     Change initialization settings including this screen

GNU Emacs 27.0.50 (build 2, x86_64-pc-linux-gnu, GTK+ Version 3.18.9)
of 2018-11-28
Copyright (C) 2018 Free Software Foundation, Inc.

If an Emacs session crashed recently, type M-x recover-session RET
to recover the files you were editing.

-UUU:%%--F1 *GNU Emacs* All L1 (Fundamental) -----
C-u C-h-
```

После этого нажмем **t**. Появится выбор языка:



```
U2 [Работает] - Oracle VM VirtualBox
File Edit Options Buffers Tools Minibuf Help
Welcome to GNU Emacs, one component of the GNU/Linux operating system.
To follow a link, click Mouse-1 on it, or move to it and type RET.
To quit a partially entered command, type Control-g.

Important Help menu items:
Emacs Tutorial      Learn basic Emacs keystroke commands
Read the Emacs Manual  View the Emacs manual using Info
(Non)Warranty       GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Conditions   Conditions for redistributing and changing Emacs
More Manuals / Ordering Manuals  How to order printed manuals from the FSF

Useful tasks:
Visit New File       Specify a new file's name, to edit the file
Open Home Directory  Open your home directory, to operate on its files
Customize Startup     Change initialization settings including this screen

GNU Emacs 27.0.50 (build 2, x86_64-pc-linux-gnu, GTK+ Version 3.18.9)
  of 2018-11-28
Copyright (C) 2018 Free Software Foundation, Inc.

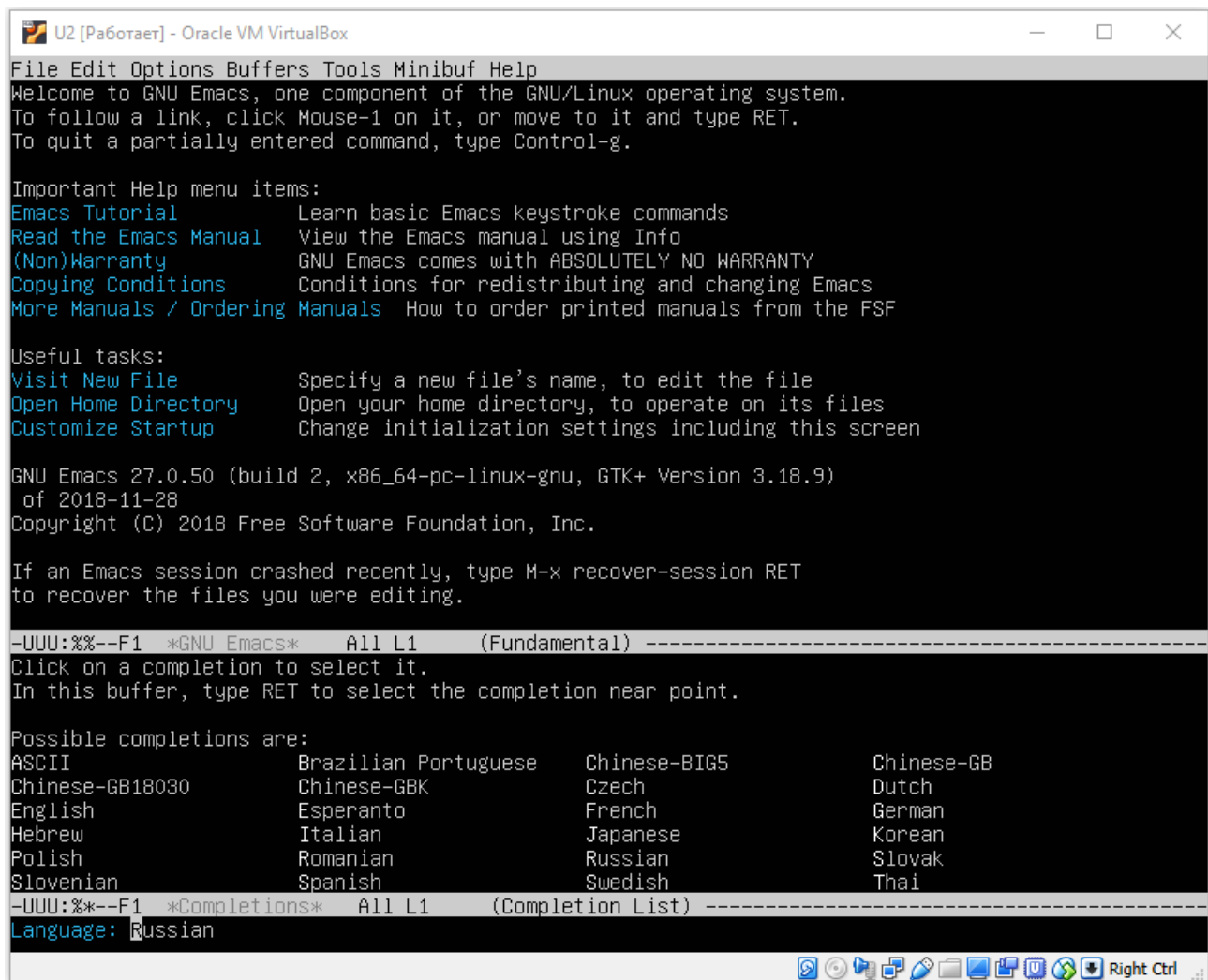
If an Emacs session crashed recently, type M-x recover-session RET
to recover the files you were editing.

-UUU:%%--F1 *GNU Emacs*  All L1  (Fundamental) -----
Click on a completion to select it.
In this buffer, type RET to select the completion near point.

Possible completions are:
ASCII      Brazilian Portuguese  Chinese-BIG5      Chinese-GB
Chinese-GB18030  Chinese-GBK      Czech             Dutch
English     Esperanto          French            German
Hebrew      Italian            Japanese          Korean
Polish      Romanian           Russian           Slovak
Slovenian   Spanish            Swedish           Thai
-UUU:%*--F1 *Completions*  All L1  (Completion List) -----
Language:
```



Кнопками «Вниз»/«Вверх» выбираем язык и жмем **Enter**:



```
U2 [Работаer] - Oracle VM VirtualBox
File Edit Options Buffers Tools Minibuf Help
Welcome to GNU Emacs, one component of the GNU/Linux operating system.
To follow a link, click Mouse-1 on it, or move to it and type RET.
To quit a partially entered command, type Control-g.

Important Help menu items:
Emacs Tutorial      Learn basic Emacs keystroke commands
Read the Emacs Manual  View the Emacs manual using Info
(Non)Warranty       GNU Emacs comes with ABSOLUTELY NO WARRANTY
Copying Conditions   Conditions for redistributing and changing Emacs
More Manuals / Ordering Manuals  How to order printed manuals from the FSF

Useful tasks:
Visit New File      Specify a new file's name, to edit the file
Open Home Directory Open your home directory, to operate on its files
Customize Startup   Change initialization settings including this screen

GNU Emacs 27.0.50 (build 2, x86_64-pc-linux-gnu, GTK+ Version 3.18.9)
of 2018-11-28
Copyright (C) 2018 Free Software Foundation, Inc.

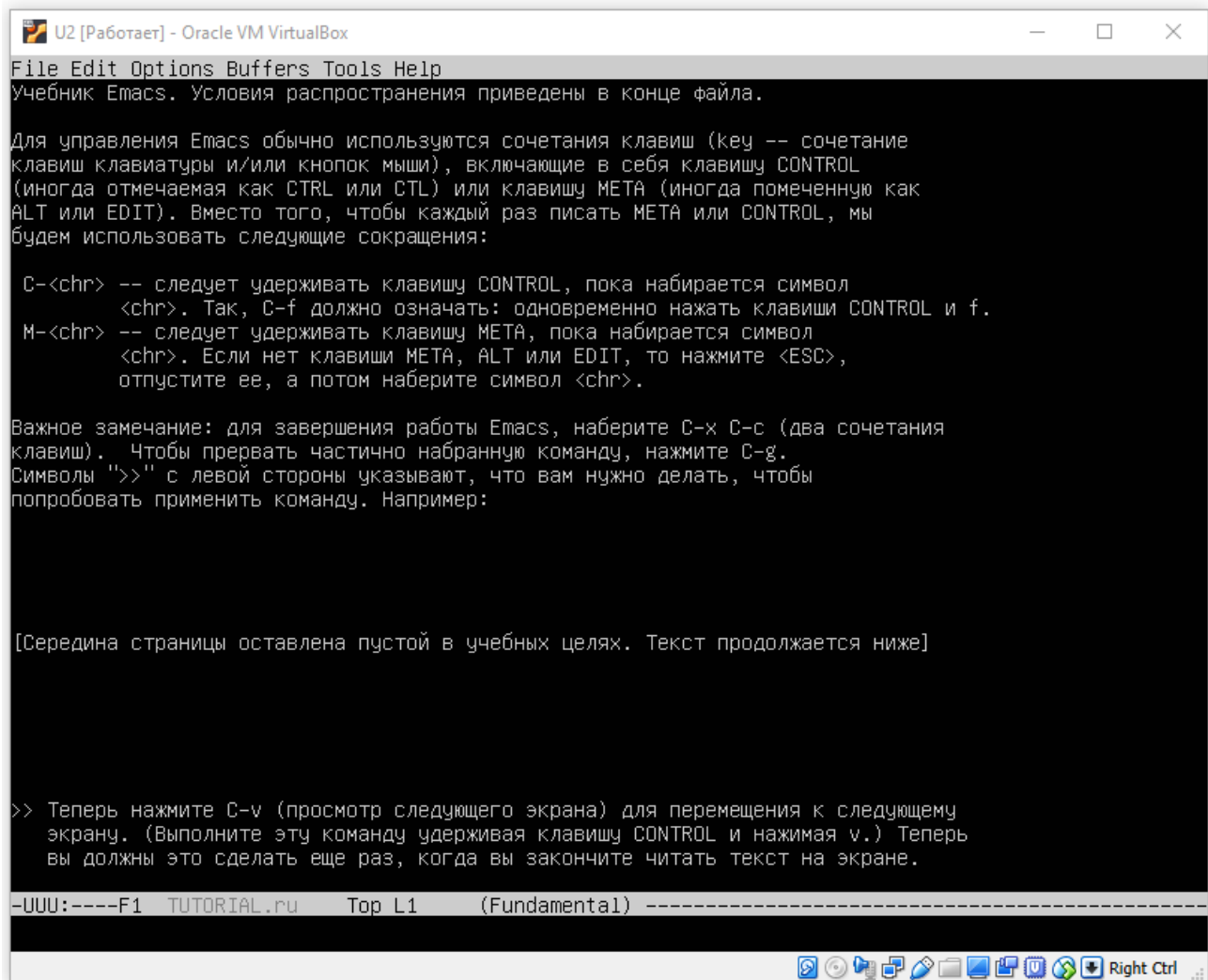
If an Emacs session crashed recently, type M-x recover-session RET
to recover the files you were editing.

-UUU:%%--F1 *GNU Emacs* All L1 (Fundamental) -----
Click on a completion to select it.
In this buffer, type RET to select the completion near point.

Possible completions are:
ASCII      Brazilian Portuguese  Chinese-BIG5      Chinese-GB
Chinese-GB18030  Chinese-GBK          Czech             Dutch
English     Esperanto            French            German
Hebrew      Italian              Japanese          Korean
Polish      Romanian             Russian           Slovak
Slovenian   Spanish              Swedish           Thai

-UUU:%%*--F1 *Completions* All L1 (Completion List) -----
Language: Russian
```

И получаем tutorial на русском:



```
File Edit Options Buffers Tools Help
Учебник Emacs. Условия распространения приведены в конце файла.

Для управления Emacs обычно используются сочетания клавиш (key -- сочетание
клавиш клавиатуры и/или кнопок мыши), включающие в себя клавишу CONTROL
(иногда отмечаемая как CTRL или CTL) или клавишу META (иногда помеченную как
ALT или EDIT). Вместо того, чтобы каждый раз писать META или CONTROL, мы
будем использовать следующие сокращения:

C-<chr> -- следует удерживать клавишу CONTROL, пока набирается символ
<chr>. Так, C-f должно означать: одновременно нажать клавиши CONTROL и f.
M-<chr> -- следует удерживать клавишу META, пока набирается символ
<chr>. Если нет клавиши META, ALT или EDIT, то нажмите <ESC>,
отпустите ее, а потом наберите символ <chr>.

Важное замечание: для завершения работы Emacs, наберите C-x C-s (два сочетания
клавиш). Чтобы прервать частично набранную команду, нажмите C-g.
Символы ">>" с левой стороны указывают, что вам нужно делать, чтобы
попытаться применить команду. Например:

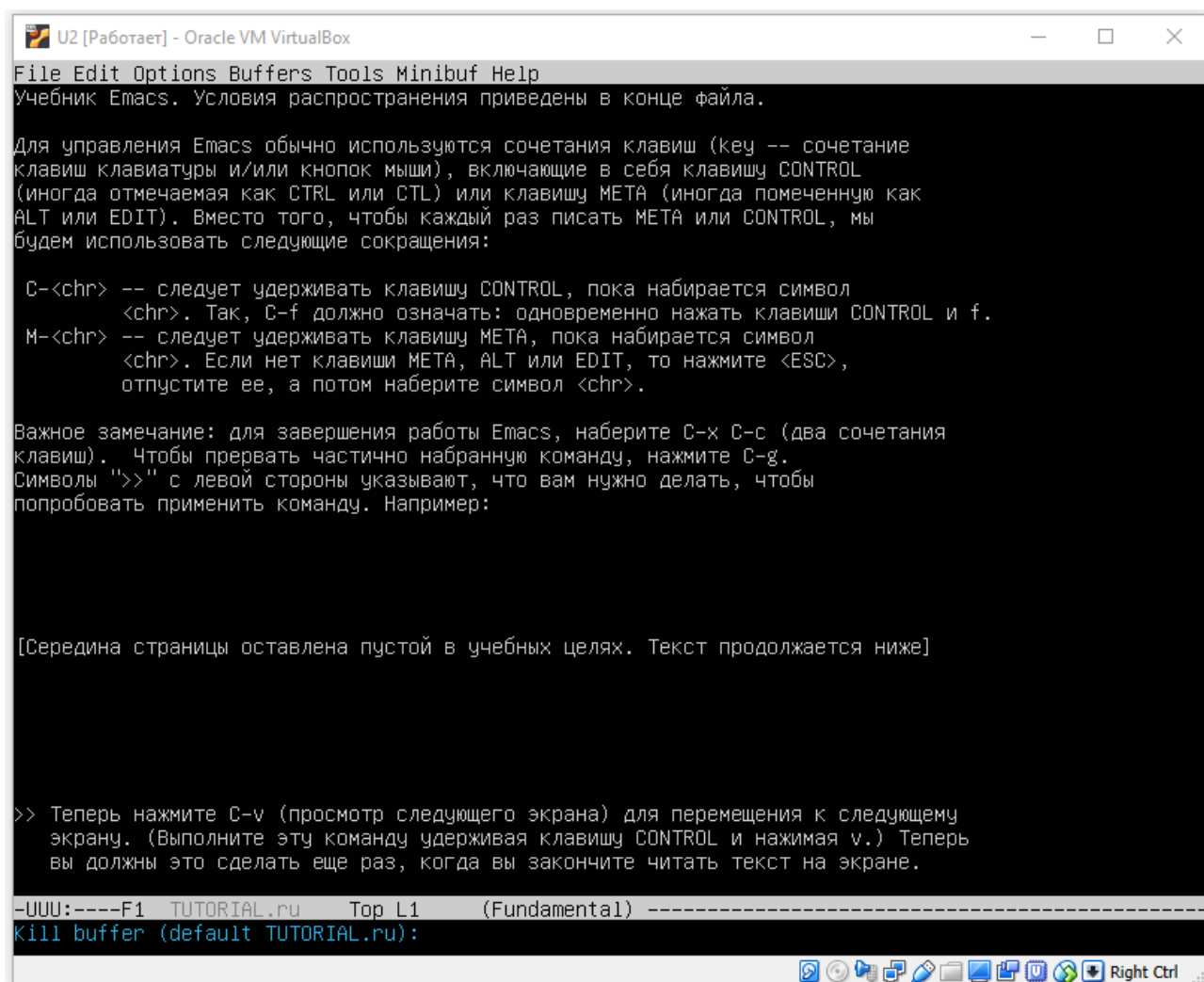
[Середина страницы оставлена пустой в учебных целях. Текст продолжается ниже]

>> Теперь нажмите C-v (просмотр следующего экрана) для перемещения к следующему
экрану. (Выполните эту команду удерживая клавишу CONTROL и нажимая v.) Теперь
вы должны это сделать еще раз, когда вы закончите читать текст на экране.

-UUU:----F1 TUTORIAL.ru Top L1 (Fundamental) -----
```

Обязательно пройдите его.

Теперь создадим новый файл. Нажмем **Ctrl-C** и следом **k** — закроем данный файл (открытые файлы называются буферами). Не



```
File Edit Options Buffers Tools Minibuf Help
Учебник Emacs. Условия распространения приведены в конце файла.

Для управления Emacs обычно используются сочетания клавиш (key -- сочетание
клавиш клавиатуры и/или кнопок мыши), включающие в себя клавишу CONTROL
(иногда отмечаемая как CTRL или CTL) или клавишу META (иногда помеченную как
ALT или EDIT). Вместо того, чтобы каждый раз писать META или CONTROL, мы
будем использовать следующие сокращения:

C-<chr> -- следует удерживать клавишу CONTROL, пока набирается символ
<chr>. Так, C-f должно означать: одновременно нажать клавиши CONTROL и f.
M-<chr> -- следует удерживать клавишу META, пока набирается символ
<chr>. Если нет клавиши META, ALT или EDIT, то нажмите <ESC>,
отпустите ее, а потом наберите символ <chr>.

Важное замечание: для завершения работы Emacs, наберите C-x C-c (два сочетания
клавиш). Чтобы прервать частично набранную команду, нажмите C-g.
Символы ">>" с левой стороны указывают, что вам нужно делать, чтобы
попробовать применить команду. Например:

[Середина страницы оставлена пустой в учебных целях. Текст продолжается ниже]

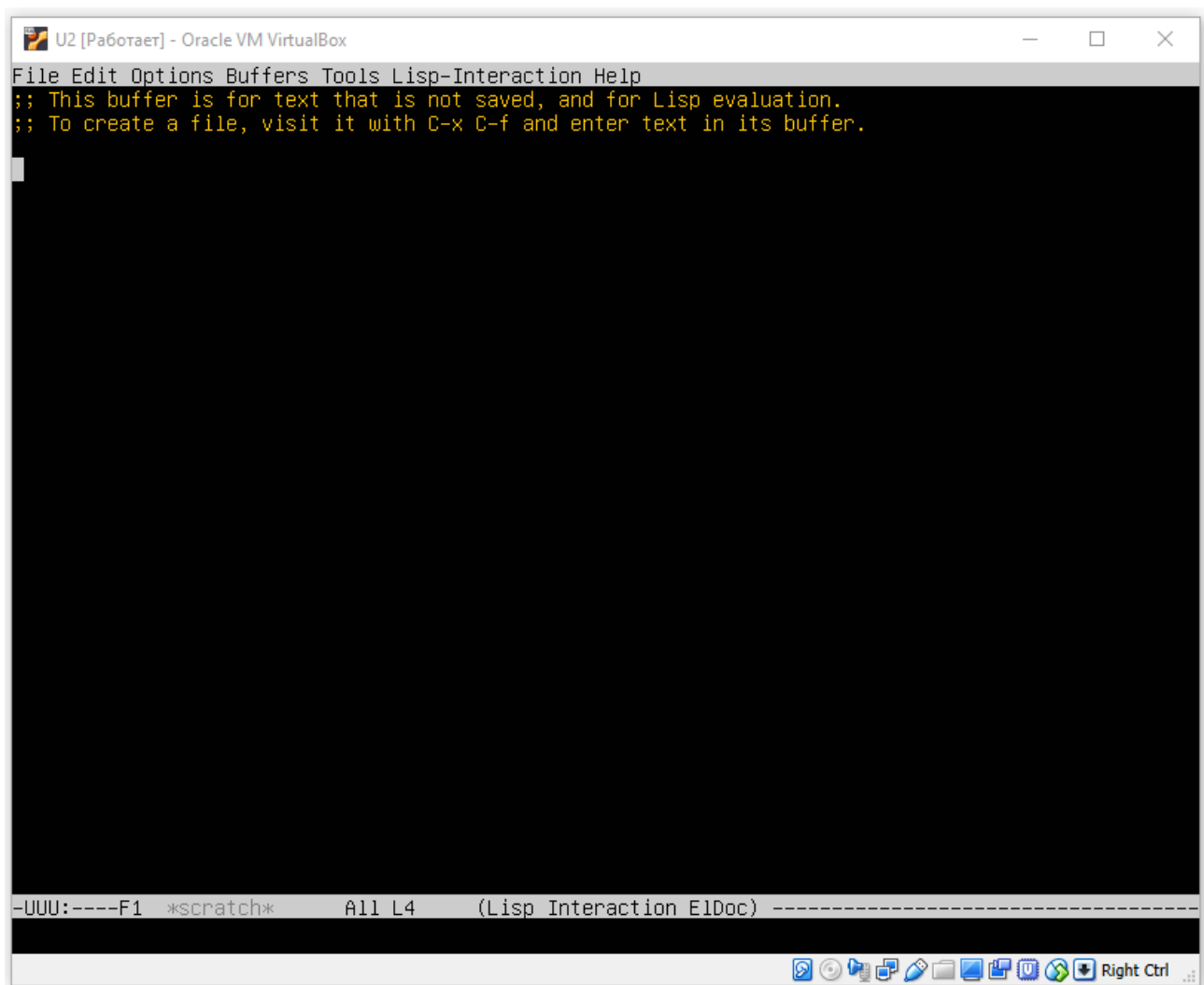
>> Теперь нажмите C-v (просмотр следующего экрана) для перемещения к следующему
экрану. (Выполните эту команду удерживая клавишу CONTROL и нажимая v.) Теперь
вы должны это сделать еще раз, когда вы закончите читать текст на экране.

-UUU:----F1 TUTORIAL.ru Top L1 (Fundamental) -----
Kill buffer (default TUTORIAL.ru):
```

Жмем **Enter-y**.

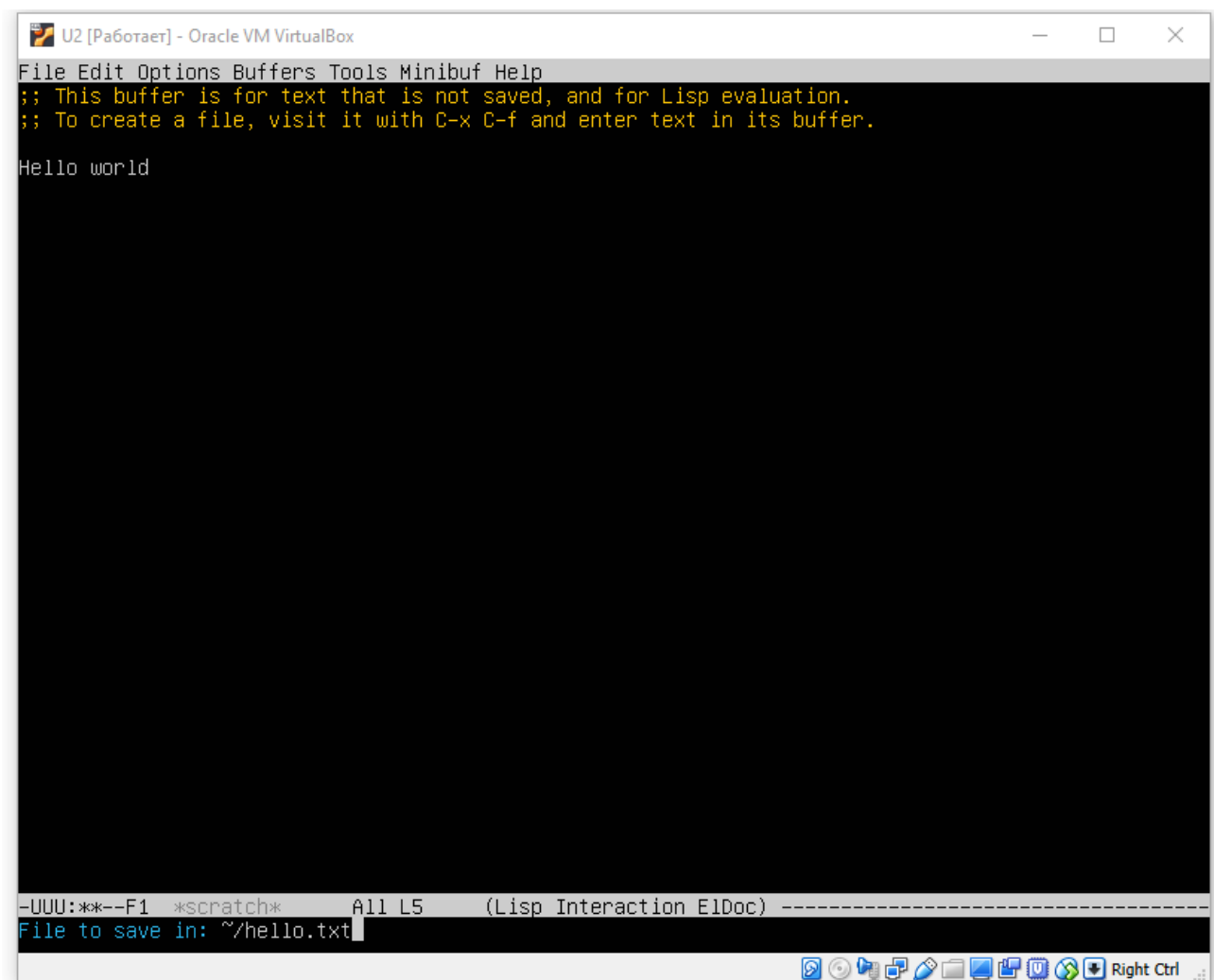
Так же закрываем файл с приветствием — **Ctrl-X k**.

Попадаем в редактирование нового Scratch-файла. Его можно использовать как временный буфер для заметок. Такой файл создается при открытии редактора и удаляется при закрытии, если не был явно сохранен на диск. Аналог папки **/tmp**:



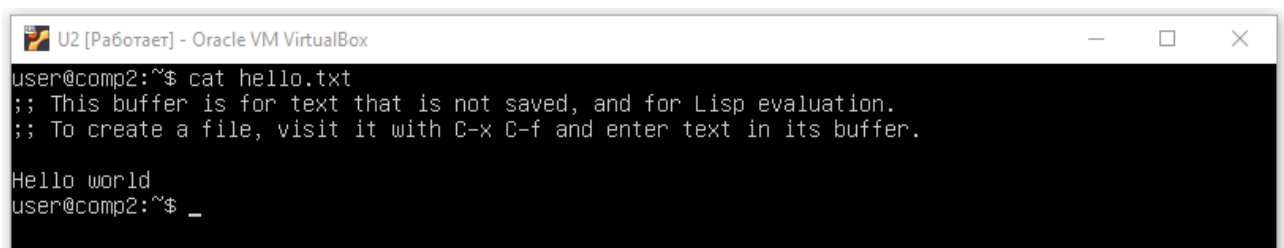
Пишем: «Hello world».

Жмем **Ctrl-X** и **Ctrl-S**. Пишем имя файла — например, **hello.txt** — и жмем **Enter**:



После этого можно выйти: **Ctrl-X Ctrl-C**.

Проверяем:



Файл существует!

Более подробно о логике Emacs можно почитать в статье [«Emacs для начинающих»](#).

## Open Source

Почему Open Source — это хорошо?

Open Source — программное обеспечение с открытым исходным кодом. Его код можно изучать, предлагать пулл-реквесты или открывать issue на исправление найденных багов и уязвимостей, можно делать форки и развивать независимо.

Open Source ПО идеально для обучения и создания некоммерческих систем, но его идея не противоречит и построению коммерческих продуктов. Многие из них используют Open Source или работают на нем, и даже делают вклад в сообщество Open Source.

Разрабатывать Open Source — престижно. Участие в таких проектах — достойная часть портфолио и резюме.

Концепция Open Source не подразумевает, что можно делать с ПО что угодно. Например, многие открытые лицензии не позволяют взять исходный код, модифицировать его и закрыть. По-разному лицензии относятся к тому, можно ли использовать свободное ПО в качестве компонентов несвободного.

Подробнее о лицензиях — в статье [«Лицензия для вашего open-source проекта»](#).

Мы будем использовать лицензию GNU GPL 3.

## Команда

Это самое важное в проекте: не продукт, не руководитель, а команда. Если она хорошая и сработанная — то справится даже с неудачным продуктом или разработает новый. Продукт может «провалиться в прокате» или поменяться, а слаженный коллектив — нет. Если вы основатель стартапа или хотя бы руководитель IT-отдела и в ваших силах сформировать климат в команде — сделайте все, чтобы она могла работать. Тогда и продукт не заставит себя ждать.

В команде работать сложно. Команда может многое, но разработка в одиночку очень сильно отличается от совместной работы. Если наедине с собой человек знает свои мысли, планы, проблемы, задержки в сроках реализации, то читать чужие мысли он не умеет. И начинает додумывать — как правило, ошибочно. Поэтому работать в команде — синхронизироваться, договариваться — надо учиться. В этом мы тоже будем практиковаться.

## Психологические отношения и ответственность

Разработка — это не только инструменты и программирование, но во многом и человеческие отношения. Участники команды должны воспринимать друг друга как равных партнеров, которые вместе ставят единую цель и делают общее дело. В этом помогут прописные истины:

- Уметь слушать других. Не умеешь — учись.
- Не самоутверждаться за счет других. В этом нет смысла.
- Ответственность — коллективная, так как вы вместе работаете над продуктом. Если хочется подумать о своей ответственности — это хорошо. Но не надо перекладывать свою ответственность на другого, искать и назначать виноватых! Успех — результат коллективной работы, а не деятельности одного гения (вариант — руководителя команды).
- Позитивно относиться ко вкладу других участников разработки. Если похвала уместна, не стоит на нее скупиться.

Разработка должна приносить радость и удовлетворение. Вы вместе занимаетесь интересным делом!

Нужно понимать, что цель — не только продукт, но и обмен знаниями, получение нового опыта.

Главное — командная сплоченность. Если пренебрежительно относиться к коллегам, но говорить о командном духе, эффекта не будет.

Отдельно стоит сказать об инициативе. Нередко на словах она декларируется, а на практике все наоборот. Если так работать, мотивация сотрудников будет падать. На одной зарплате, без энтузиазма, мотивация сойдет на нет. Поэтому важно поощрять инициативу и позволять коллегам проявлять себя. У продукта, сделанного с любовью и вниманием, больше шансов стать успешным, чем у разработанного «из-под палки».

В удаленной команде — все то же самое!

Главное — понимать: самая важная ценность — люди, и не только члены вашей команды, но и будущие пользователи продукта.

Если вы сработаетесь как команда, можно разработать не один успешный продукт. Если проект провалился, можно с учетом опыта разработать еще один.

## Минимальный рабочий прототип — MVP

Как разработать продукт, чтобы он «выстрелил»? Все большое начинается с малого. Разработать «убийцу» продукта, который обслуживает штат в сотни, а то и тысячи человек, вряд ли получится. Но если посмотреть историю такого конкурента, можно увидеть, что и у него было скромное начало.

Поэтому цель разработки — продукт с минимальными функциями, но уже работающий и решающий одну небольшую, но важную задачу.

Это MVP (Minimum Viable Product) — минимально жизнеспособный продукт. Можно назвать его минимальным рабочим прототипом, но между ними есть небольшое отличие. MVP — уже завершённый результат труда, пусть маленький и с небольшим набором функций, но уже способный продаваться, выполнять задачу, заинтересовать свою аудиторию, скачиваться.

Важно: небольшие цели приводят к таким же ошибкам. Минимальный рабочий продукт проще отладить, потому что неприятные баги программного обеспечения на старте могут стоить репутации и поставить крест на разработке. Этого не надо бояться, на ошибках учатся. Но лучше воспользоваться чужим негативным опытом и избежать неприятностей на старте.

Потом можно будет добавлять модули, расширяться. Главная задача на данном этапе — сделать минимальный жизнеспособный продукт. Такой подход хорош по трем причинам:

- MVP реально сделать, не растеряв энтузиазма. Долгая разработка при отсутствии финансовой отдачи может очень быстро свести на нет мотивацию команды.
- Если продукт не затребуют, то трудозатраты будут меньше, чем если стремиться сделать сразу все фичи.
- Возможность быстро получить обратную связь от аудитории и переориентироваться при необходимости, совершить пивот — решительный разворот, смену бизнес-модели или переориентацию на не самые очевидные в начале фичи.

## Рынок, конкуренты

Идеи для продукта появляются не на пустом месте, а на основе анализа. А чаще всего хорошие продукты — это не новая сногшибательная идея, а удачное улучшение или повторная реализация того, что уже существует.

И в этом помогает анализ конкурентов и аналогов. В свободном ПО тоже есть конкуренция. Какой продукт установят и будут использовать — ваш или другого автора? Но это не значит, что с конкурентами надо бороться.

Их надо благодарить за то, что они есть! Нет конкурентов — нет аналогов. Значит и пользователя не будет.

Если сервис предлагает нечто совершенно новое и непонятное, он выйдет на рынок только при основательной рекламной и финансовой поддержке. Только крупные игроки могут создавать спрос с нуля. Бывает, что одиночка придумывает новинку, которая распространяется благодаря сарафанному радио и маркетинговому эффекту — но на это не стоит надеяться. Если чувствуете в себе потенциал — попробуйте. Если не уверены — лучше воспользоваться более традиционным и проверенным путем.

Почему на рынках продавцы с похожим ассортиментом торгуют рядом — они же конкуренты? Их суммарное присутствие создает рекламу, привлекает внимание, имеет кумулятивный эффект. Так и с IT-рынком: конкуренты бесплатно прививают вашим будущим клиентам практику использования продукта, рекламируют потребность в нем.

Мы выбрали для реализации продукт с достаточно сильными конкурентами. Мало того, что с Emacs конкурирует vi/vim, но и мы планируем конкурировать с оригинальным Emacs. Так-то.

## Жизненный цикл ПО, проекта, продукта

Классические подходы:

- **Цикл Деминга** — PDCA: *plan* — *do* — *check* — *action*, то есть *планируй* — *действуй* — *проверяй* — *меняй*. Применяется в стандарте менеджмента качества ISO 9001:2008, в разработке ПО это итеративная модель.
- **Каскадная модель**, или «Водопад»: выработка требований — анализ — проектирование — кодирование — тестирование — эксплуатация.
- **Спиральная модель** — разработка через прототипирование с многократным анализом, проектированием и кодированием; с тестированием с учетом полученного результата; с внесением изменений в требования и учетом недостатков при следующей итерации.

Методологии разработки часто фактически сами являются моделью жизненного цикла ПО или содержат ее.

Жизненный цикл проекта, с одной стороны, связан с продуктом. С другой, проект — это процесс, а продукт — результат.

В целом жизненный цикл проекта можно описать такт:

1. **Первичный сбор требований.** Идея обретает первоначальные формулировки. Определяется цель проекта и бюджет.
2. **Планирование.** Оценка рисков, сроков, формирование команды.
3. **Выполнение работ.** Собственно разработка.
4. **Завершение проекта.** Передача проекта заказчику, обучение, вывод на рынок.
5. Иногда добавляется еще одна стадия — **поддержка и сопровождение**.

Жизненный цикл продукта оказывается обширнее, чем цикл ПО — которым продукт может являться в узком смысле — и проекта.



С жизненным циклом продукта мы познакомимся на практике, итеративно разрабатывая Open Source ПО.

## Практическое задание

1. Установить Emacs, научиться в нем работать.
2. Сформулировать, что нужно сделать, чтобы приблизиться к нашей цели.
3. Предложить, какую задачу по Open Source редактору сможете взять.

## Дополнительные материалы

1. [Launchpad.net — PPA description](#).
2. [Emacs для начинающих](#).
3. [Emacs для начинающих: HOW TO](#).
4. [Emacs на Wiki](#).
5. [«Открытое программное обеспечение» на Wiki](#)/
6. [Лицензия для вашего open-source проекта](#).
7. [GNU General Public License на Wiki](#).

## Используемая литература

Для подготовки данного методического пособия были использованы следующие ресурсы:

1. [Ubuntu Emacs Lisp — PPA description](#).
2. [Как перейти на новейшие Emacs?](#)
3. [Вход и выход из Emacs](#).
4. [Введение в Emacs](#).
5. [Emacs на Wiki](#).