



МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «ЛИПЕЦКИЙ
ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ»

Институт компьютерных наук
Кафедра автоматизированных систем управления

Лабораторная работа №1
по операционным системам

Студент АС-21-1 _____ Станиславчук С. М.
(подпись, дата)

Руководитель
Доцент, к.п.н. _____ Кургасов В. В.
(подпись, дата)

Липецк 2024

Содержание

1. Задание кафедры	3
2. Цель работы	4
3. Ход работы.....	5
3.1 Часть1	5
3.2 Часть 2	20
4. Контрольные вопросы	33
5. Вывод.....	45

1. Задание кафедры

Проанализировать три гипервизора второго типа по выбранным самостоятельно критериям.

1. Создать виртуальную машину.
2. Выполнить установку ОС. В качестве имени пользователя использовать свою фамилию (латиницей).
3. Зайти в терминал и продемонстрировать результат работы команды: `grep` (без использования и с использованием конвейеризации).
4. Дополнительно, продемонстрировать работу нескольких команд в терминале (количество определяется последней цифрой номера студенческого билета, увеличенной на 3) на выбор, отметив наличие команды в ОС Windows с аналогичным действием (если таковая имеется, сравнить результаты работы).
5. Подготовьте отчет о выполненной работе, включив в отчет описание настройки виртуальной машины, процесса установки ОС, сопровождая скриншотами, письменные ответы на поставленные в работе вопросы.
6. Настройте сетевой интерфейс виртуальной машины, чтобы она была видна в локальной сети. Продемонстрируйте доступность результатом выполнения команды `ping`. Опишите и проиллюстрируйте предпринятые действия пошагово.
7. Создайте общую папку гостевой ОС и ОС, установленной на вашей рабочей станции: опишите ваши шаги и продемонстрируйте доступность.

2. Цель работы

Ознакомиться с программными продуктами для виртуализации, научиться устанавливать на виртуальную машину различные ОС и получить навыки их настройки. Получить представление о технологиях и средствах разработки программного обеспечения в ОС Unix.

3. Ход работы

3.1 Часть 1

Проанализируем три гипервизора второго типа:

- 1) Oracle Virtual Box
- 2) VMware Workstation
- 3) QEMU

Сравним их по следующим критериям: лицензирование, с какими ОС работает, возможность создания снимков файловой системы, какой интерфейс использует.

Oracle Virtual Box и QEMU являются программами с открытым исходным кодом и доступны для скачивания бесплатно. Для того, чтобы пользоваться VMware Workstation нужно заплатить за лицензию, хотя и существует бесплатная версия VMware Player, но у нее очень ограниченный функционал.

Oracle Virtual Box и QEMU доступны для всех популярных ОС, в отличие от VMware Workstation, который доступен только для Windows и Linux.

Oracle Virtual Box и VMware Workstation имеют возможность создавать снимки файловой системы - фиксация определённых настроек и самой виртуальной машины, и операционной системы, и установленных программ в определённый момент. QEMU не имеет такой возможности.

Oracle Virtual Box и VMware Workstation имеют дружелюбный для пользователя графический интерфейс, с помощью которого можно выполнить всю настройку виртуальной машины. QEMU не имеет графического интерфейса, работа производится через командную строку.

Запустим Oracle VM VirtualBox. На рисунке 1 представлено окно после запуска.

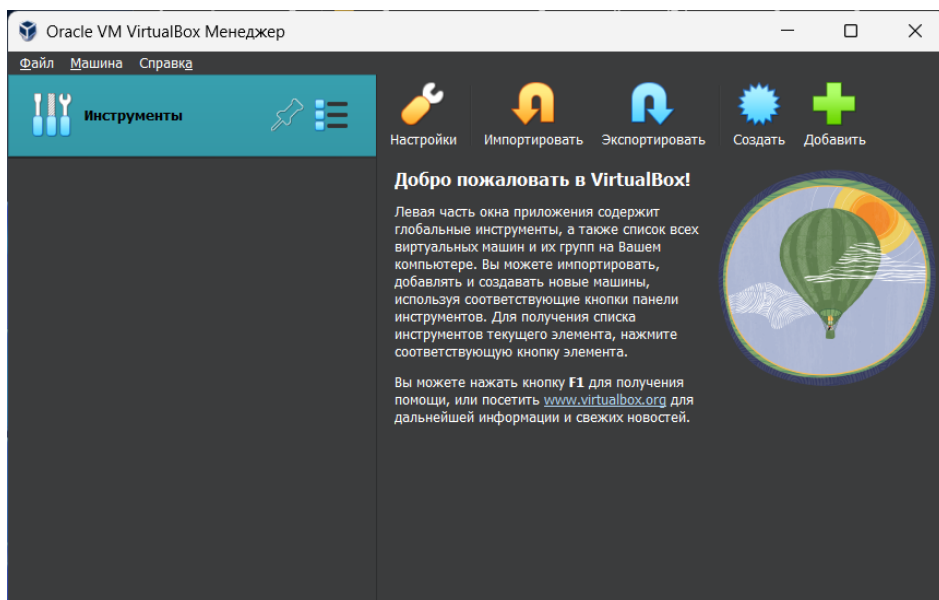


Рисунок 1 - Окно после запуска

Создадим новую виртуальную машину нажатием кнопки «Создать».

Укажем имя ВМ, папку, где будут храниться файлы ВМ, заранее скаченный образ AstraLinux и редакцию, тип, версию ОС.

На следующем этапе настроим оборудование. Выберем количество ОЗУ и виртуальных процессоров.

Создадим виртуальный жесткий диск размером 25 Гб.

После выполнения всех действий получим таблицу, которая подытоживает конфигурацию, выбранную нами.

После нажатия «Готово» запустим нашу виртуальную машину кнопкой «Запустить».

Произошла загрузка с установочного диска AstraLinux. Произведем установку данной ОС. Выберем язык и пункт «Графическая установка».

Принимаем лицензионное соглашение и переходим на следующий шаг.

Выберем удобный нам способ переключения раскладки клавиатуры и перейдем на следующий шаг.

Введем имя нашего компьютера и перейдем на следующий шаг.

Введем имя учетной записи администратора «antonova» и перейдем на следующий шаг.

Придумаем пароль для нашей учетной записи и перейдем на следующий шаг.

Выберем часовой пояс и нажмем «Продолжить». Произойдет определение дисков.

Выберем метод разметки и нажмем «Продолжить».

Выберем диск, на который установится ОС.

Выберем схему разметки.

Удостоверимся, что все выбрано правильно и нажмем «Продолжить».

Подтвердим запись изменений на диск. Произойдет установка базовой системы.

Выберем ядро для установки и нажмем «Продолжить».

Выберем нужное программное обеспечение и нажмем «Продолжить».

Произведется установка выбранного ПО.

По желанию можно выбрать дополнительные настройки ОС. Нажмем «Продолжить».

Установим загрузчик GRUB в главную загрузочную запись и нажмем «Продолжить».

Выберем жесткий диск, на который мы устанавливали ОС и нажмем «Продолжить».

После завершения установки нажмем «Продолжить» и подождем пока загрузится ОС. Состояние виртуальной машины после запуска ОС представлено на рисунке 2.

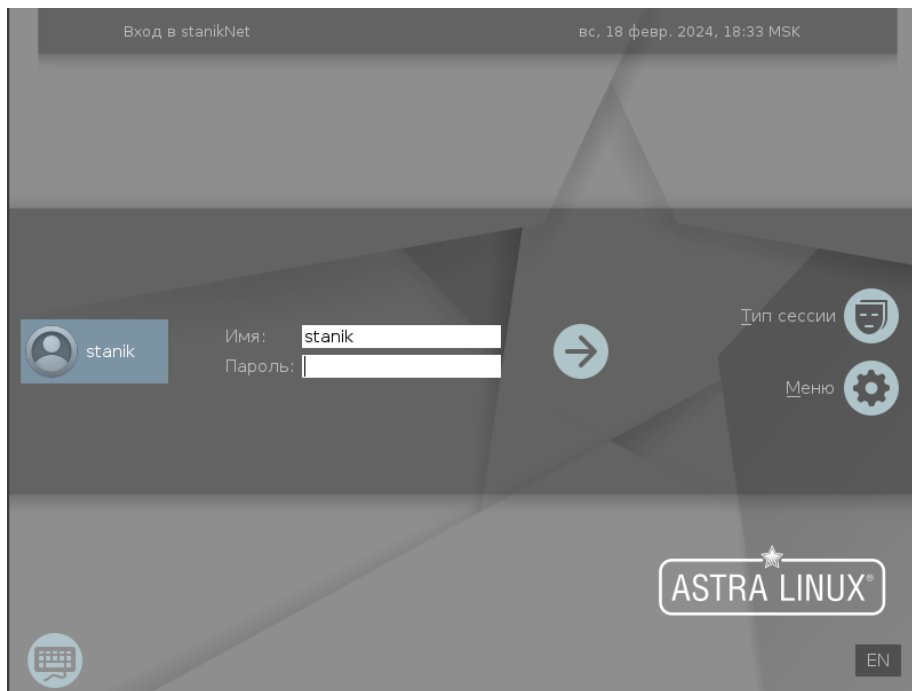


Рисунок 2 - Вход в astra

Войдем в нашу учетную запись, созданную при установке ОС.
Результат представлен на рисунке 3.

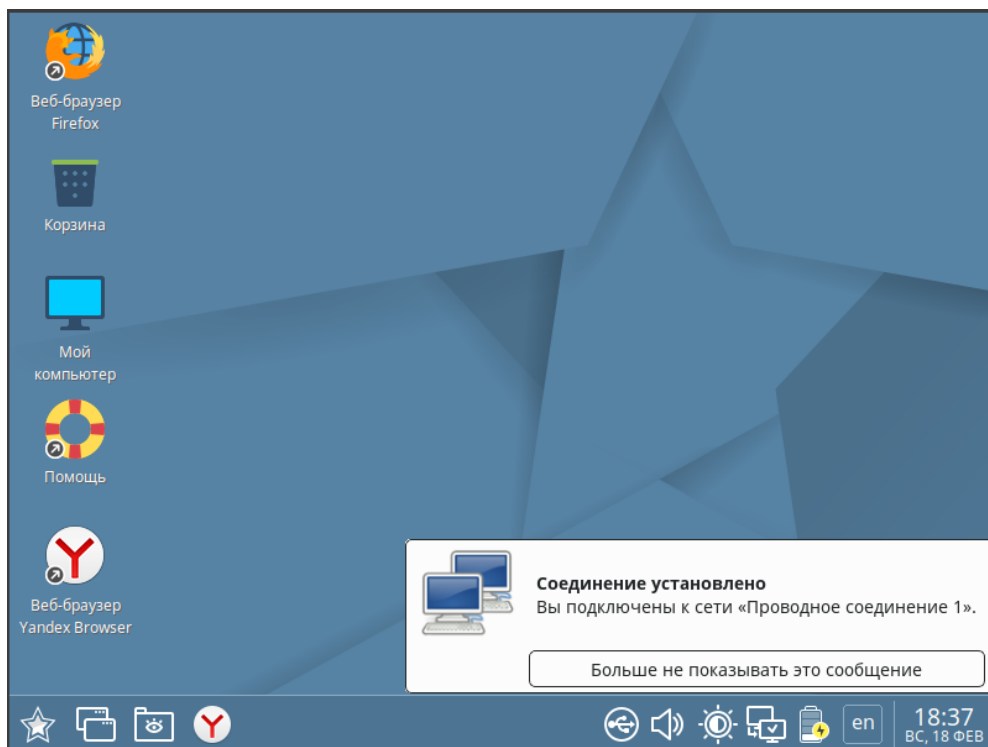


Рисунок 3 - Рабочий стол

Создадим тестовый файл для проверки команды `grep`, который
представлен на рисунке 4.

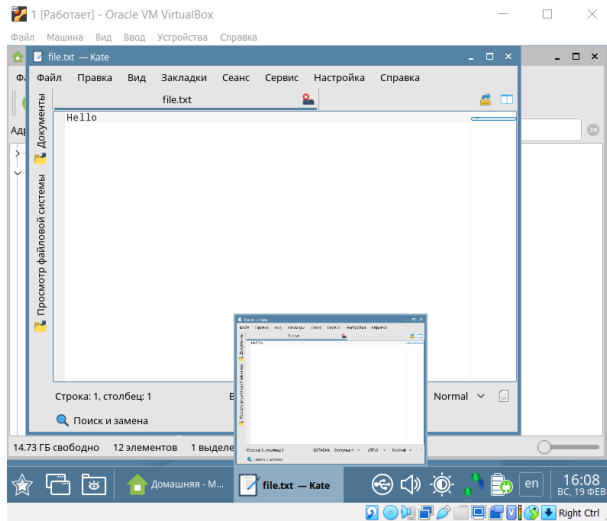


Рисунок 2 - Тестовый файл

Зайдем в терминал и выполним команду `grep file.txt`. Это пример использования команды `grep` без использования конвейеризации. Результат представлен на рисунке 5.

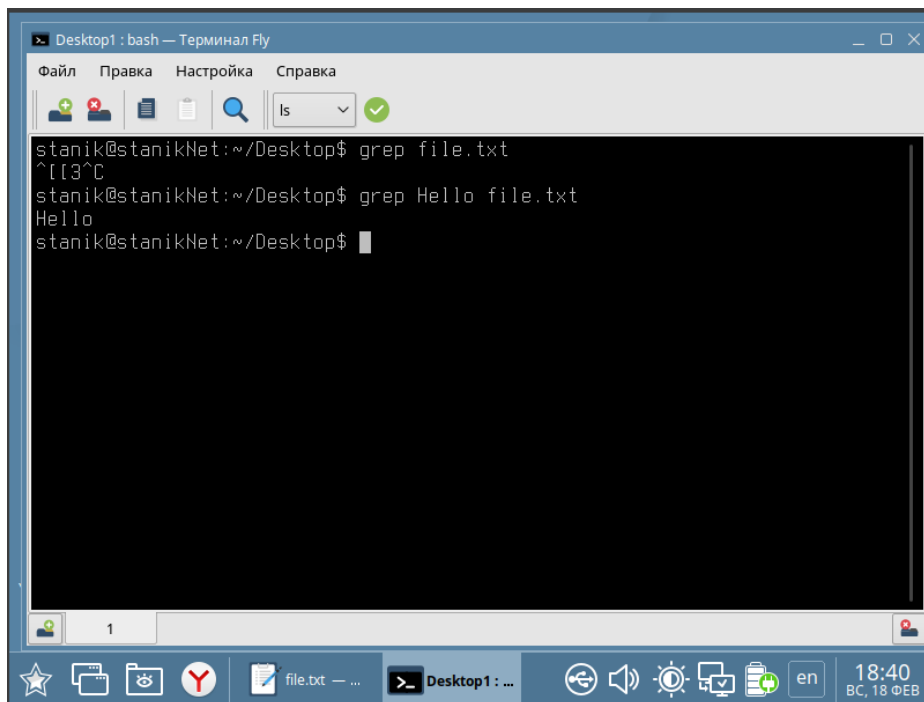


Рисунок 5 - Пример использования команды `grep` без использования конвейеризации

Создадим еще один тестовый файл, он представлен на рисунке 6.

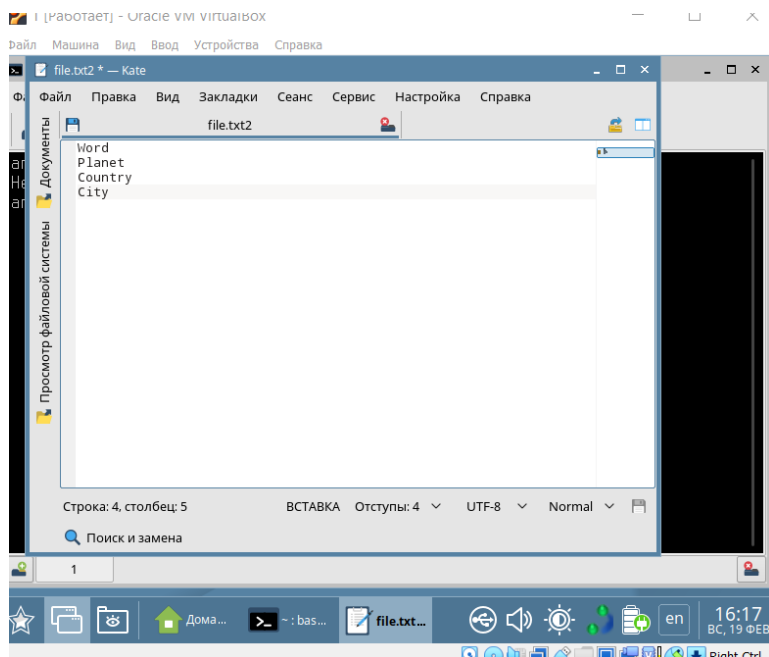


Рисунок 6 - Тестовый файл

Выполним команду `grep` с использованием конвейеризации.

Результат представлен на рисунке 7.

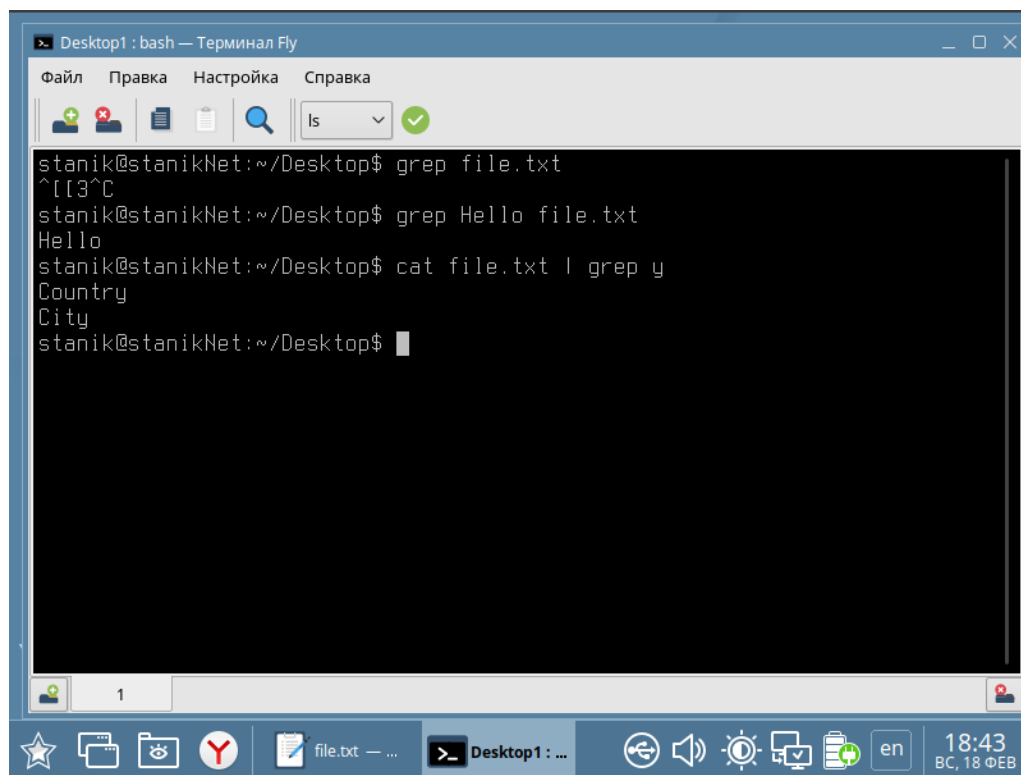


Рисунок 7 - Пример использования команды `grep` с конвейеризацией

Выполним несколько команд в терминале AstraLinux и

попробуем найти их эквиваленты в Windows.

1. Создание папки

Пример команды AstraLinux представлен на рисунке 8.

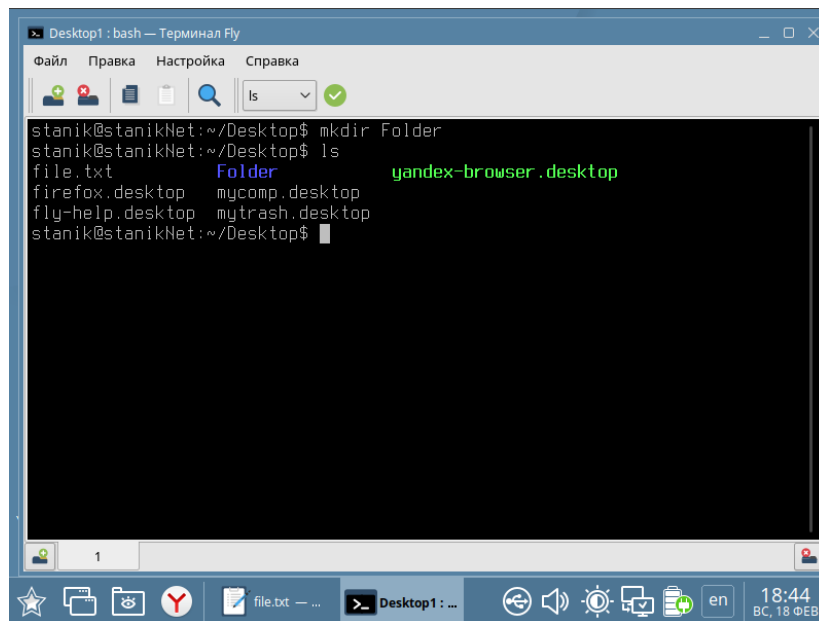


Рисунок 3 - Команда mkdir

Пример команды Windows представлен на рисунке 9.

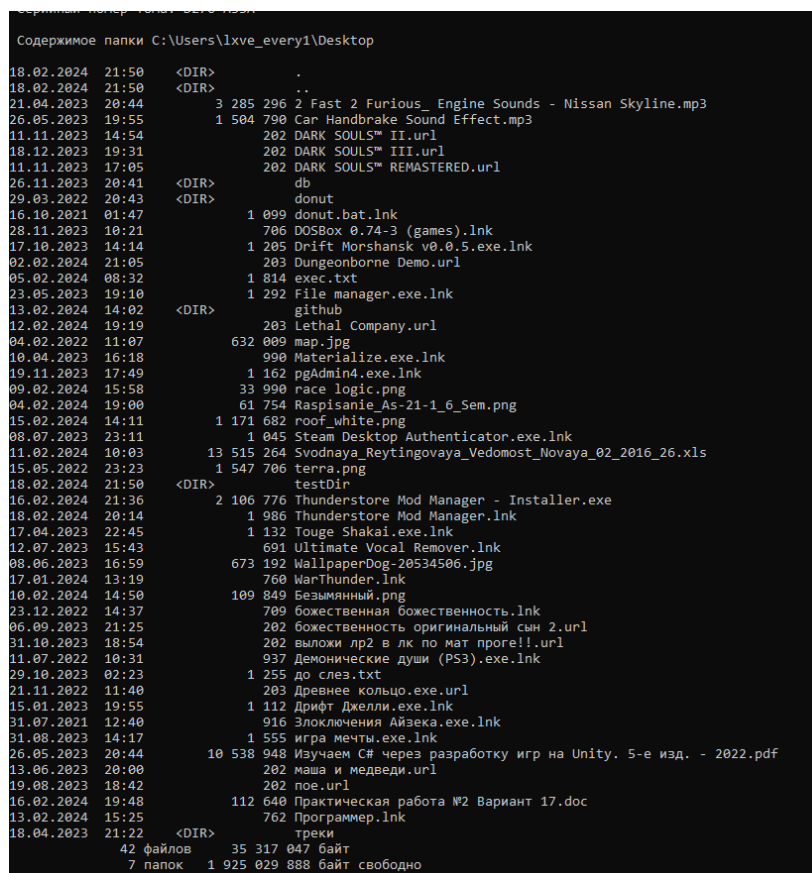


Рисунок 9 - Команда md

Команда «mkdir» в AstraLinux имеет эквивалент в Windows – «md».

Обе этих команды принимают в качестве аргумента название создаваемой папки.

2. Просмотр директории

Пример команды AstraLinux представлен на рисунке 10.

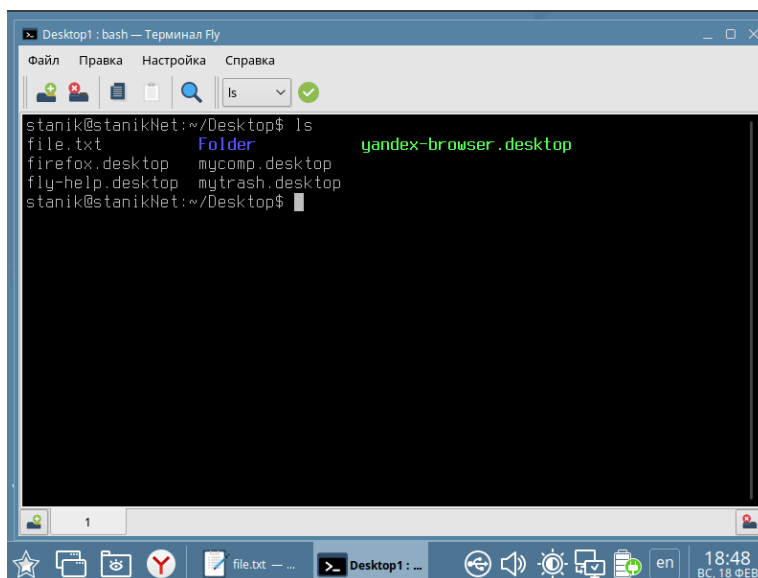


Рисунок 10 - Команда ls

Пример команды Windows представлен на рисунке 11.

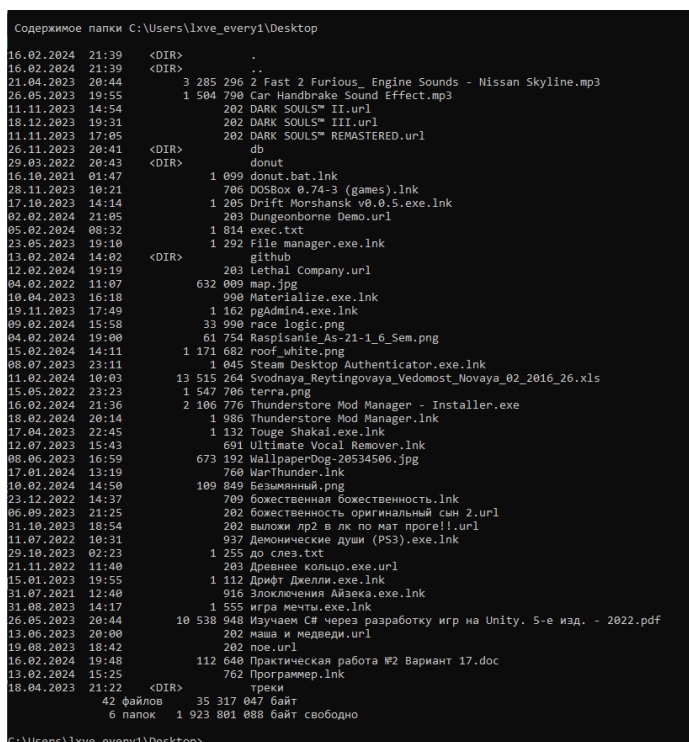
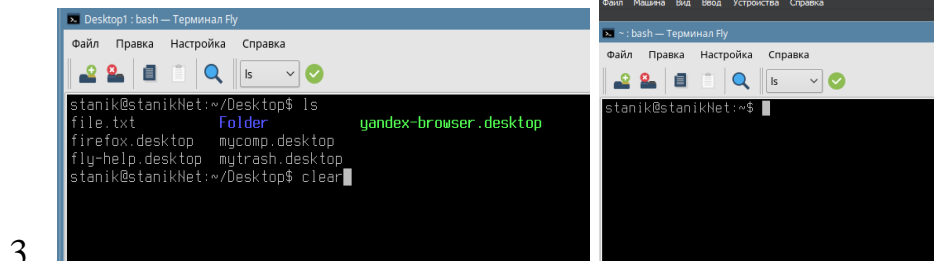


Рисунок 11 - Команда dir

Команда «ls» в AstraLinux имеет эквивалент в Windows – «dir».

По сравнению с AstraLinux, Windows выводит еще дополнительную информацию о серийном номере тома, дате и времени создания папки.



3.

Очистка экрана командной строки

Пример команды AstraLinux представлен на рисунке 12.

Рисунок 12 - Команда clear

Пример команды Windows представлен на рисунке 13.

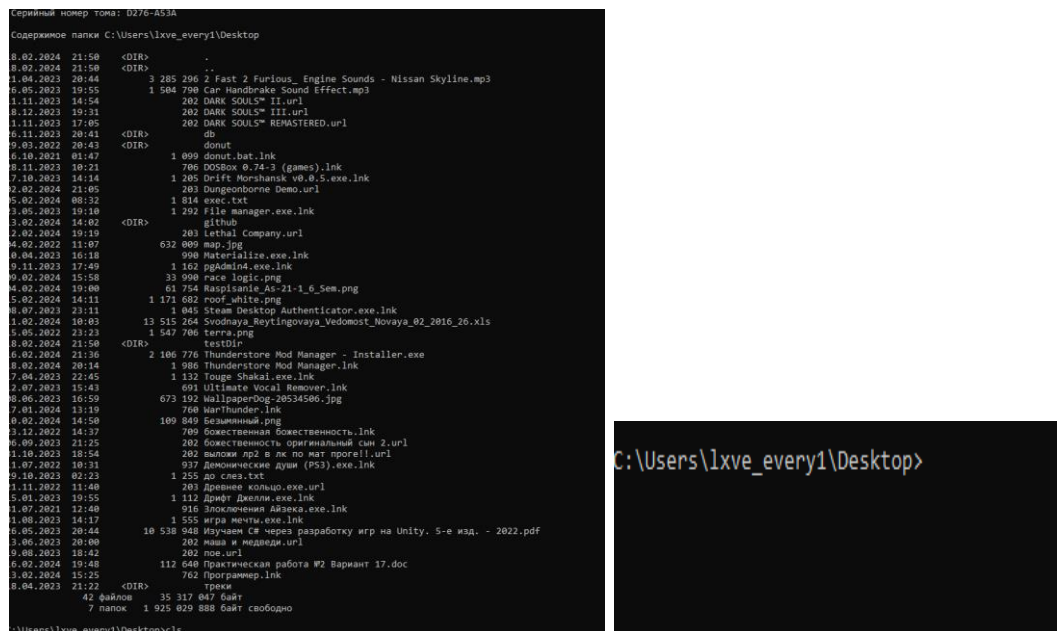


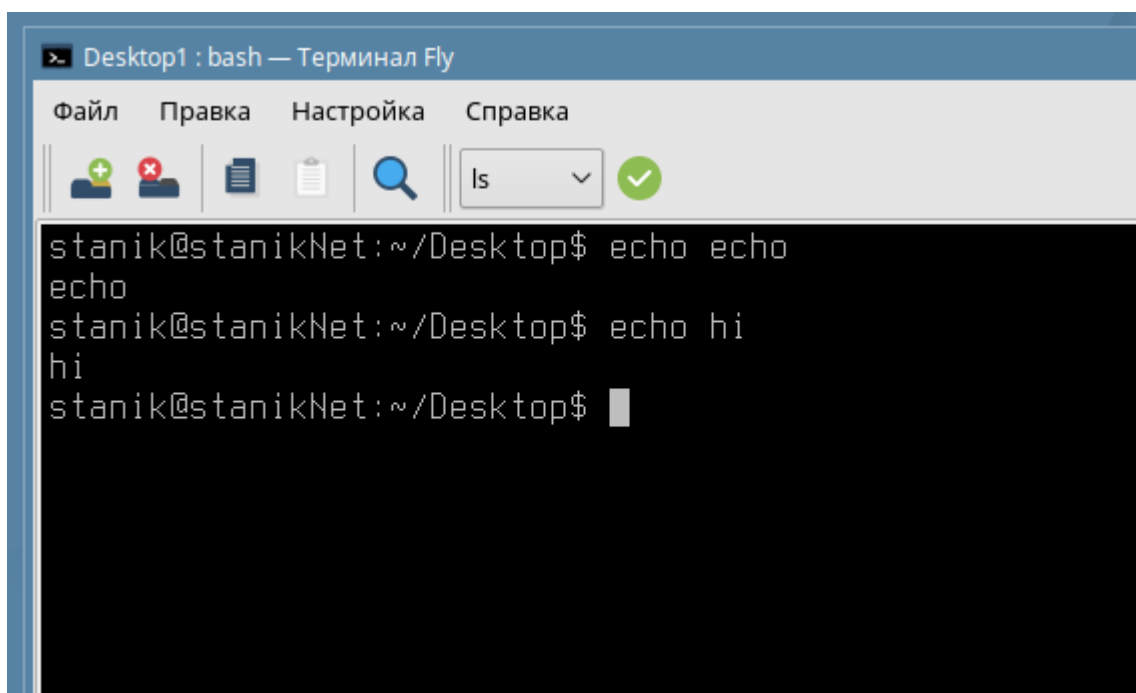
Рисунок 13 - Команда clr

Команда «clear» в AstraLinux имеет эквивалент в Windows – «cls».

Единственная разница в этих командах – это их название.

4. Вывод сообщения на экран

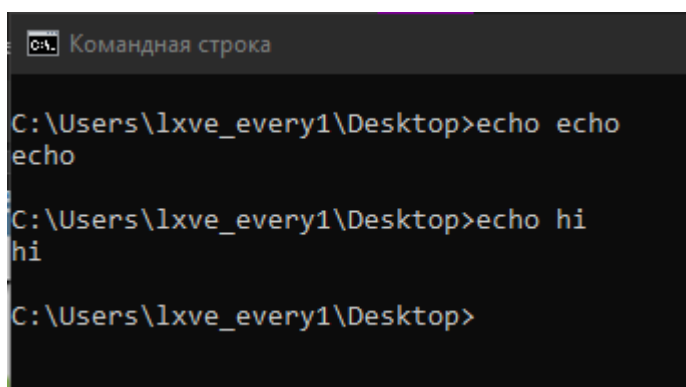
Пример команды AstraLinux представлен на рисунке 14.



```
Desktop1 : bash — Терминал Fly
Файл  Правка  Настройка  Справка
[Icons]  [ls]  [Checkmark]
stanik@stanikNet:~/Desktop$ echo echo
echo
stanik@stanikNet:~/Desktop$ echo hi
hi
stanik@stanikNet:~/Desktop$
```

Рисунок 14 - Команда echo

Пример команды Windows представлен на рисунке 15.



```
Командная строка
C:\Users\lxve_every1\Desktop>echo echo
echo
C:\Users\lxve_every1\Desktop>echo hi
hi
C:\Users\lxve_every1\Desktop>
```

Рисунок 15 - Команда echo

Команда «echo» в AstraLinux имеет эквивалент в Windows – «echo». Обе этих команды принимают в качестве аргумента строку, которая будет выведена в консоль.

Настроим сетевой интерфейс ВМ так, чтобы она была видна в локальной сети. Для этого зайдём в настройки ВМ и выберем «Сеть». Результат представлен на рисунке 16.

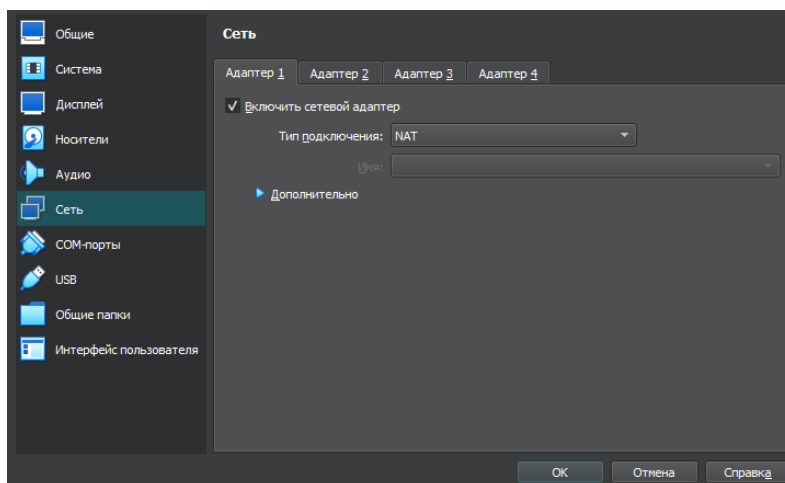


Рисунок 16 - Настройка сети

В типе подключения выберем «Сетевой мост». В этом случае виртуальная машина подключается напрямую к основной сети как полноценное устройство. Для подключения используется сетевая карта хост-системы. Результат представлен на рисунке 17.

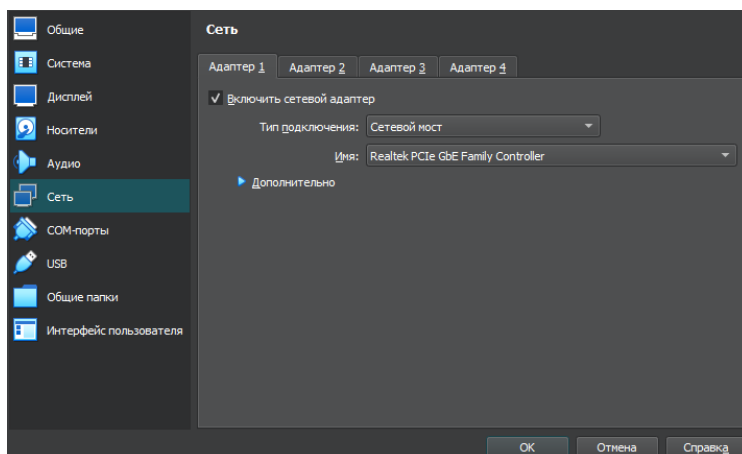
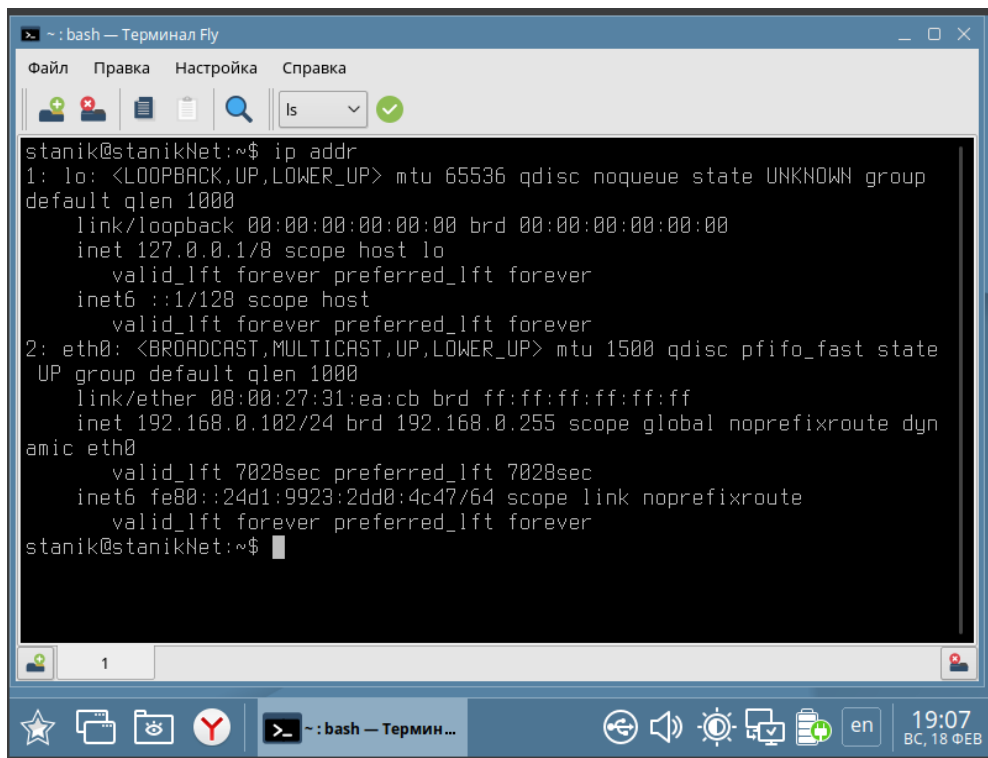


Рисунок 17 - Настройка сети

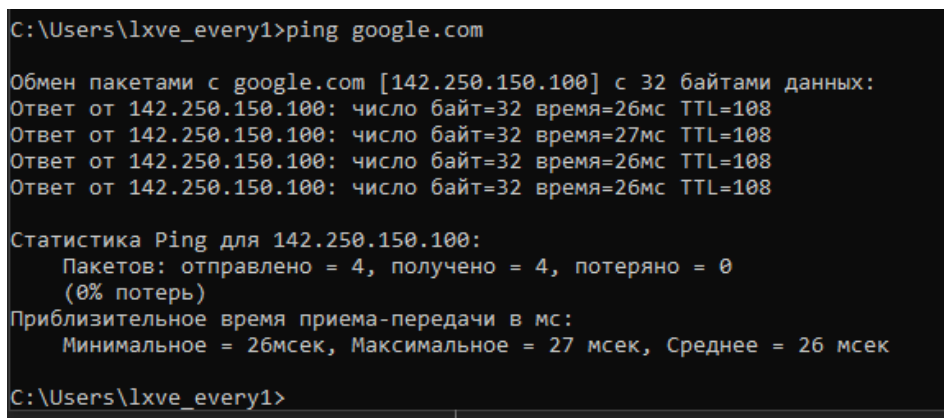
Запустим ВМ и узнаем IP – адрес ВМ через команду в терминале. Результат представлен на рисунке 18.



```
stanik@stanikNet:~$ ip addr
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN group
default qlen 1000
    link/loopback 00:00:00:00:00:00 brd 00:00:00:00:00:00
    inet 127.0.0.1/8 scope host lo
        valid_lft forever preferred_lft forever
    inet6 ::1/128 scope host
        valid_lft forever preferred_lft forever
2: eth0: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc pfifo_fast state
UP group default qlen 1000
    link/ether 08:00:27:31:ea:cb brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.102/24 brd 192.168.0.255 scope global noprefixroute dyn
amic eth0
        valid_lft 7028sec preferred_lft 7028sec
    inet6 fe80::24d1:9923:2dd0:4c47/64 scope link noprefixroute
        valid_lft forever preferred_lft forever
stanik@stanikNet:~$
```

Рисунок 18 - Команда ip addr

IP – адрес VM – 192.168.0.104. Проверим доступность данной VM в хост системе командой ping. Результат представлен на рисунке 19



```
C:\Users\lxve_every1>ping google.com

Обмен пакетами с google.com [142.250.150.100] с 32 байтами данных:
Ответ от 142.250.150.100: число байт=32 время=26мс TTL=108
Ответ от 142.250.150.100: число байт=32 время=27мс TTL=108
Ответ от 142.250.150.100: число байт=32 время=26мс TTL=108
Ответ от 142.250.150.100: число байт=32 время=26мс TTL=108

Статистика Ping для 142.250.150.100:
    Пакетов: отправлено = 4, получено = 4, потеряно = 0
    (0% потерь)
Приблизительное время приема-передачи в мс:
    Минимальное = 26мсек, Максимальное = 27 мсек, Среднее = 26 мсек

C:\Users\lxve_every1>
```

Рисунок 19 - Команда ping

Команда ping была выполнена успешно, следовательно VM теперь находится и видна в локальной сети.

Создадим общую папку гостевой ОС и ОС, установленной на рабочей станции. Для этого зайдём в настройки виртуальной машины и выберем вкладку «Общие папки». Результат представлен на рисунке 20.

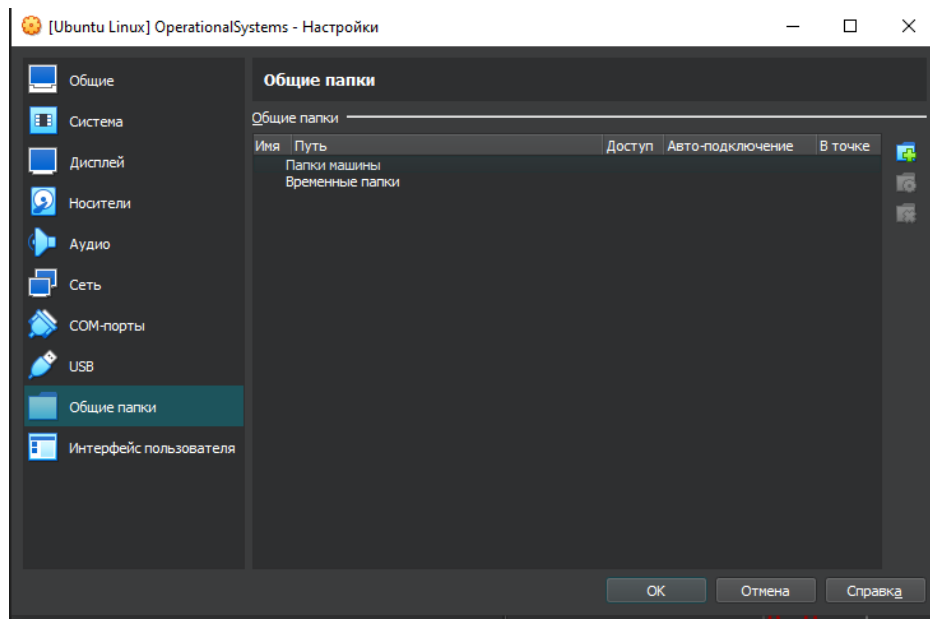


Рисунок 20 - Общие папки

Нажмем на кнопку «Добавить общую папку». В открывшемся окне выберем путь к папке, название папки и точку подключения. Результат представлен на рисунке 21.

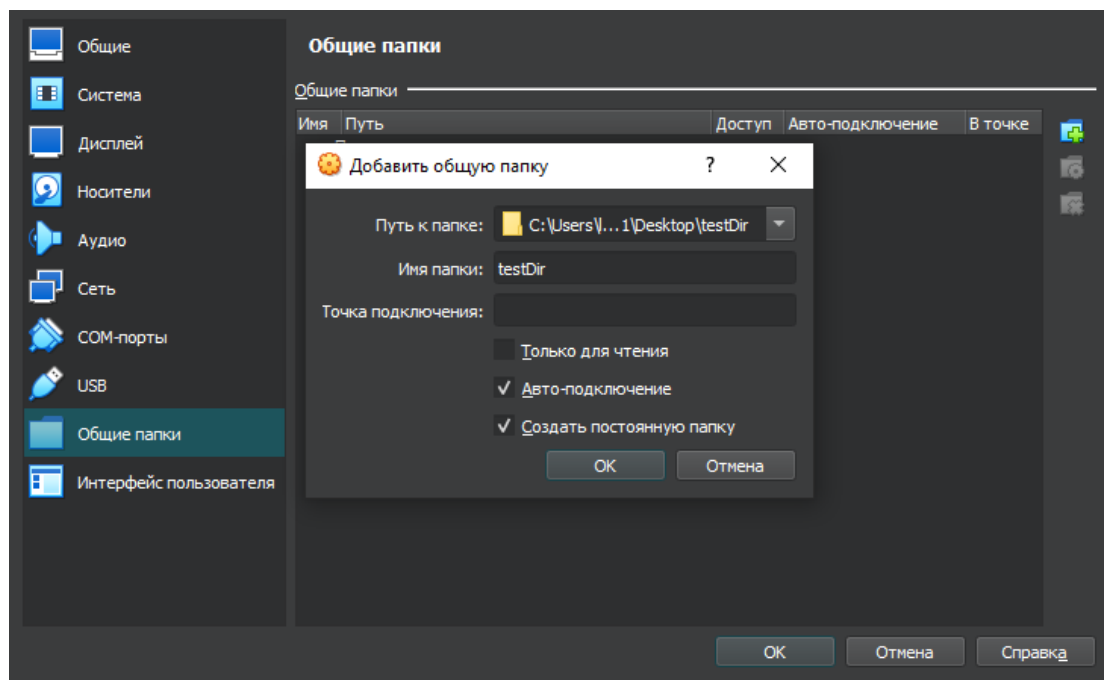


Рисунок 21 - Изменить общую папку

```

stanik@stanikNet:/media/cdrom$ sudo bash vboxWindowsAdditions.run
Мы полагаем, что Ваш системный администратор изложил Вам основы
безопасности. Как правило, Всё сводится к трём следующим правилам:

    №1) Уважайте частную жизнь других.
    №2) Думайте, прежде что-то Вводить.
    №3) С большой Властью приходит большая ответственность.

[sudo] пароль для stanik:
bash: vboxWindowsAdditions.run: Нет такого файла или каталога
stanik@stanikNet:/media/cdrom$ sudo bash VBoxWindowsAdditions.run
bash: VBoxWindowsAdditions.run: Нет такого файла или каталога
stanik@stanikNet:/media/cdrom$ sudo bash VBoxLinuxAdditions.run
Verifying archive integrity... 100% MD5 checksums are OK. All good.
Uncompressing VirtualBox 7.0.10 Guest Additions for Linux 100%
VirtualBox Guest Additions installer
Copying additional installer modules ...
Installing additional modules ...
VirtualBox Guest Additions: Starting.
VirtualBox Guest Additions: Setting up modules
VirtualBox Guest Additions: Building the VirtualBox Guest Additions kernel
modules. This may take a while.
VirtualBox Guest Additions: To build modules for other installed kernels,
run
VirtualBox Guest Additions: /sbin/rcvboxadd quicksetup <version>
VirtualBox Guest Additions: or
VirtualBox Guest Additions: /sbin/rcvboxadd quicksetup all
VirtualBox Guest Additions: Building the modules for kernel 5.4.0-110-gene
ric.
update-initramfs: Generating /boot/initrd.img-5.4.0-110-generic
VirtualBox Guest Additions: reloading kernel modules and services
VirtualBox Guest Additions: kernel modules and services 7.0.10 r158379 rel
oaded
VirtualBox Guest Additions: NOTE: you may still consider to re-login if so
me
user session specific services (Shared Clipboard, Drag and Drop, Seamless
or
Guest Screen Resize) were not restarted automatically
stanik@stanikNet:/media/cdrom$ █

```

Рисунок 22 – выполнение команда sudo bash

VBoxLinuxAdditions.run*

Для доступа в появившийся каталог нужно повысить права пользователя и перезапустить систему (sudo usermod-aG vboxsf stanik) (sudo reboot)

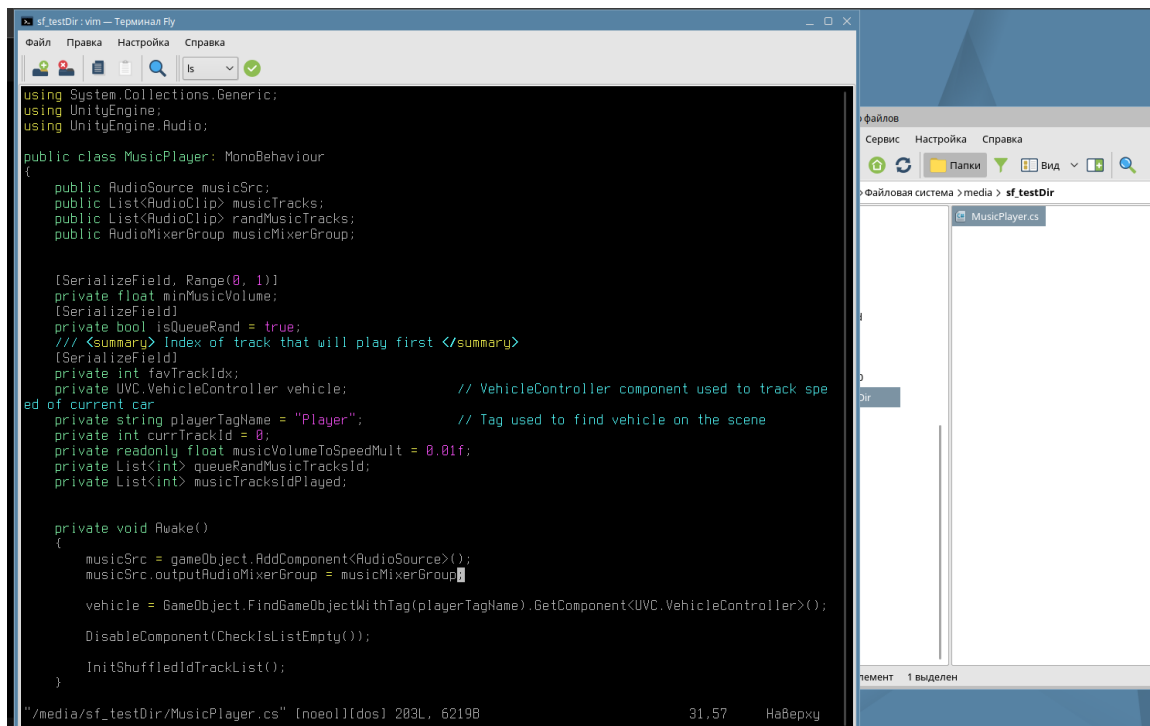


Рисунок 23 – Доступ к файлу из хостовой ОС в эмулированной системе

3.2 Часть 2

Сравним несколько популярных IDE доступных для Linux. В сравнении будут участвовать следующие IDE:

1) Code::Blocks (<http://www.codeblocks.org>) – свободная кросс-платформенная интегрированная среда разработки, написанная на C++ и поддерживаемая языки C, C++, D (с ограничениями), Fortran. Распространяется как бесплатное ПО под лицензией GNU GPLv3. Code::Blocks разрабатывается для Windows, Linux и macOS. Среда можно собрать из исходников практически под любую Unix-подобную систему, например FreeBSD, PC-BSD. Последняя версия доступная на данный момент – 20.03 от 29 марта 2020.

2) Netbeans for C/C++ Development (<https://netbeans.apache.org/>) – это свободная и популярная кроссплатформенная IDE с открытым исходным кодом для C/C++, Java, Python, PHP, JavaScript, Ада. Распространяется как бесплатное ПО под лицензией Apache License 2.0. Данная IDE разрабатывается для Windows, Linux, macOS, Solaris. Последняя версия доступная на данный момент – 20 от 1 декабря 2023.

3) Eclipse CDT (<http://www.eclipse.org/cdt/>) – это кроссплатформенная IDE с открытым исходным кодом. Изначально среда была разработана для языка Java, но в настоящее время существуют многочисленные расширения для поддержки и других языков. Таким образом, сейчас Eclipse поддерживает C/C++, Fortan, Perl, PHP, JavaScript, Python, Ruby, 1C V8. Распространяется как бесплатное ПО под лицензией Eclipse Public License. Данная IDE разрабатывается для Windows, Linux, macOS, Solaris. Последняя версия доступная на данный момент – 4.3.0 от 6 декабря 2023

4) CodeLite IDE (<http://codelite.org/>) – бесплатная кроссплатформенная IDE с открытым исходным кодом, разработанная и созданная специально для программирования на C/C++, JavaScript

(Node.js) и PHP. Распространяется по лицензии GNU General Public License v2. Данная IDE разрабатывается для Windows, Linux, macOS, Solaris. Последняя версия доступная на данный момент – 17.0.0 от 14 января 2023

5) Bluefish (<http://bluefish.openoffice.nl>) – предлагает функции IDE для разработки веб-сайтов, написания скриптов и программного кода. Поддерживаются языки ASP.NET и VBS, CFML, Clojure, CSS, D, gettext PO, Google Go, HTML, Java, JavaScript, jQuery, Lua, MATLAB, MediaWiki, NSIS, Pascal, Perl, PHP, Python, R, Ruby, SQL, Wordpress, XHTML и HTML5, XML, Ada, C. Данная IDE разрабатывается для Windows, Linux, macOS, Solaris. Последняя версия доступная на данный момент – 2.2.14 от 3 июня 2023

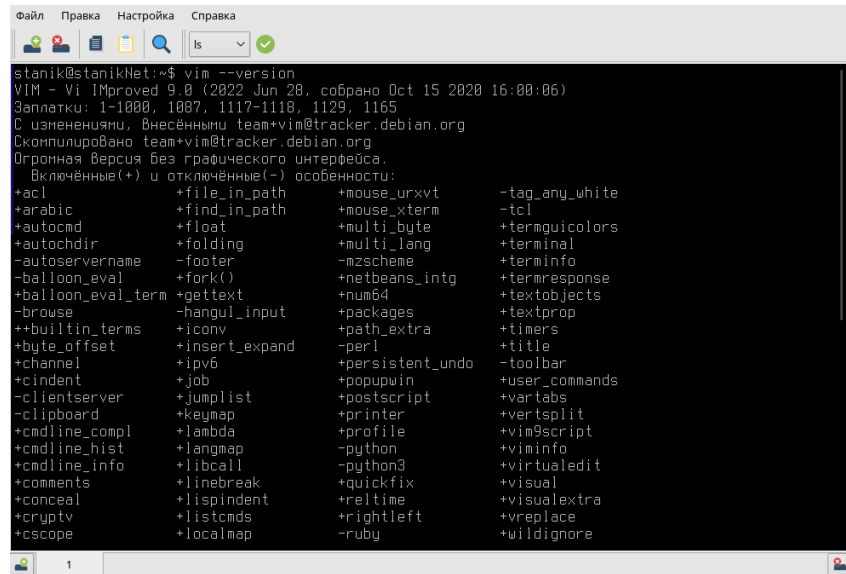
6) Brackets (<https://brackets.io/>) – текстовый редактор с открытым исходным кодом, разработанный специально для веб-дизайна и разработки. В основном нацелен на работу с HTML, CSS, JavaScript, но с помощью плагинов можно использовать язык C/C++. Данная IDE разрабатывается для Windows, Linux, macOS, Solaris. Последняя версия доступная на данный момент – 2.0.1 от 22 марта 2023.

7) Atom (<https://atom.io/>) – мультиплатформенный текстовый редактор с открытым исходным кодом, который может работать в Linux, Windows, MacOS. Поддерживаются языки PHP, HTML, Json, SQL, XML, CSS, CoffeeScript, JavaScript, Java, C/C++, Go. Последняя версия доступная на данный момент – 1.63.1 от 23 ноября 2022 (разработка прекращена).

Все выбранные для сравнения IDE являются кроссплатформенными, поэтому их можно использовать на любой ОС доступной пользователю. Самый богатый выбор для работы с языками программирования предлагает Bluefish, самый малый выбор – Code::Blocks. Большинство IDE активно разрабатываются и получают обновления и по сей день, кроме Atom, так как его разработка была прекращена.

Проверим наличие редактора Vim в AstraLinux командой `vim --version`.

Результат представлен на рисунке 22.

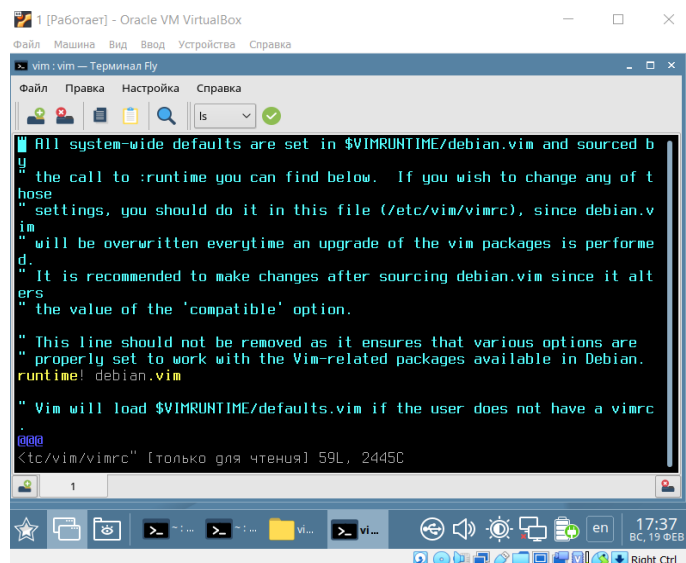


```
stanik@stanikNet:~$ vim --version
VIM - Vi IMproved 9.0 (2022 Jun 28, собрано Oct 15 2020 16:00:06)
Запатки: 1-1000, 1007, 1117-1118, 1129, 1165
С изменениями, Внесёнными team+vim@tracker.debian.org
Скомпилировано team+vim@tracker.debian.org
Огромная Версия без графического интерфейса.
Включённые(+) и отключённые(-) особенности:
+acl                +file_in_path      +mouse_urxvt       -tag_any_white
+arabic             +find_in_path      +mouse_xterm       -tcl
+autocmd            +float             +multi_byte        +termguicolors
+autocmdir          +floating           +multi_lang        +terminal
+autoservername     -footer            -mzscheme          +terminfo
+balloon_eval       +fork()            +netbeans_intg     +termresponse
+balloon_eval_term +gettext           +num64             +textobjects
-browse             -hangul_input      +packages          +textprop
++builtin_terms     +iconv             +path_extra        +timers
+byte_offset        +insert_expand     -perl              +title
+channel            +ipv6              +persistent_undo   -toolbar
+cindent            +job               +popupwin          +user_commands
-clientserver       +jumplist          +postscript        +vartabs
+clipboard          +keymap            +printer           +vertsplit
+cmdline_compl      +lambda            +profile           +vim9script
+cmdline_hist       +langmap           -python            +viminfo
+cmdline_info       +libcall           -python3           +virtualedit
+comments           +linebreak         +quickfix          +visual
+conceal            +lispindent        +reltime           +visualextra
+cryptv             +listcmds          +rightleft         +vreplace
+cscope             +localmap          -ruby              +wildignore
```

Рисунок 22 - Проверка версии vim

Мы получили информацию об установленной версии Vim, следовательно он у нас имеется в системе по умолчанию. Продолжим настройку.

Основной конфигурационный файл Vim находится в папке /etc/. Это файл /etc/vim/vimrc. Настроим Vim для стандарта Python PEP-8. Откроем файл в редакторе командой vim /etc/vim/vimrc. Результат представлен на рисунке 23.



```
vim: vimrc - Терминал Py
vim: vimrc - Терминал Py
" All system-wide defaults are set in $VIMRUNTIME/debian.vim and sourced b
u
" the call to :runtime you can find below. If you wish to change any of t
hose
" settings, you should do it in this file (/etc/vim/vimrc), since debian.v
im
" will be overwritten everytime an upgrade of the vim packages is performe
d.
" It is recommended to make changes after sourcing debian.vim since it alt
ers
" the value of the 'compatible' option.

" This line should not be removed as it ensures that various options are
" properly set to work with the Vim-related packages available in Debian.
runtime! debian.vim

" Vim will load $VIMRUNTIME/defaults.vim if the user does not have a vimrc
.
@@@
<tc/vim/vimrc" [только для чтения] 59L, 2445C
```

Рисунок 23 - Файл /etc/vim/vimrc

Изначально vimrc содержит множество технической информации

не нужной нам, поэтому откроем файл и перейдем в конец файла, используя визуальный режим и некоторые хоткеи обычного режима: перейдём в визуальный режим — v, теперь нажмём G (перевод курсора в конец документа). Результат представлен на рисунке 24.

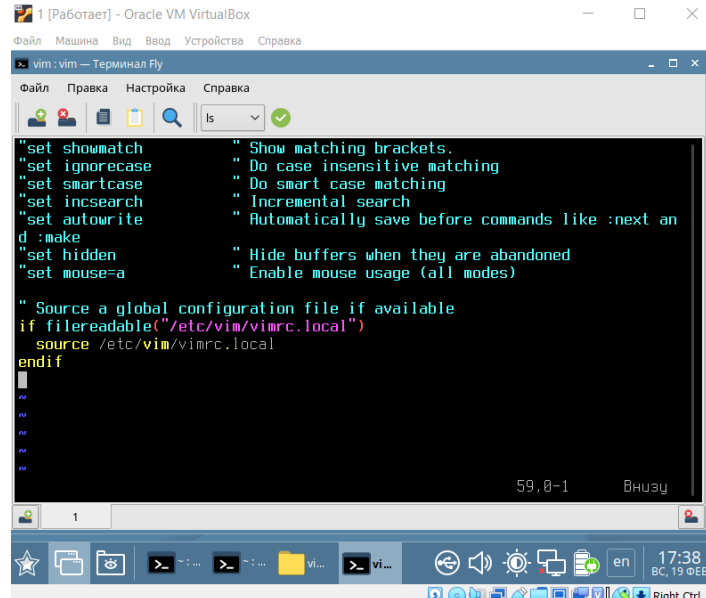


Рисунок 24 - Конец файла

Нажмем i для перехода в режим правки и добавим строки параметров табуляции. Результат представлен на рисунке 25.

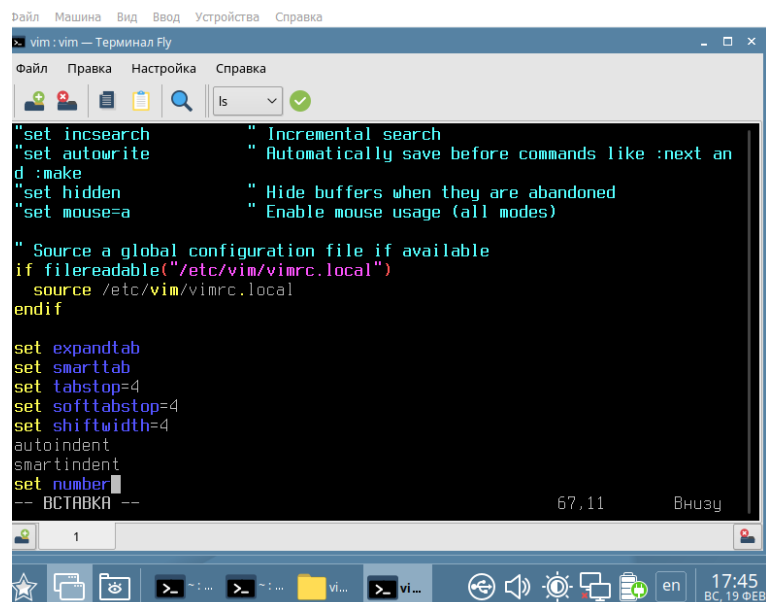


Рисунок 25 - Изменение параметров

С помощью команды :wq! запишем файл и выйдем. Результат

представлен на рисунке 26.

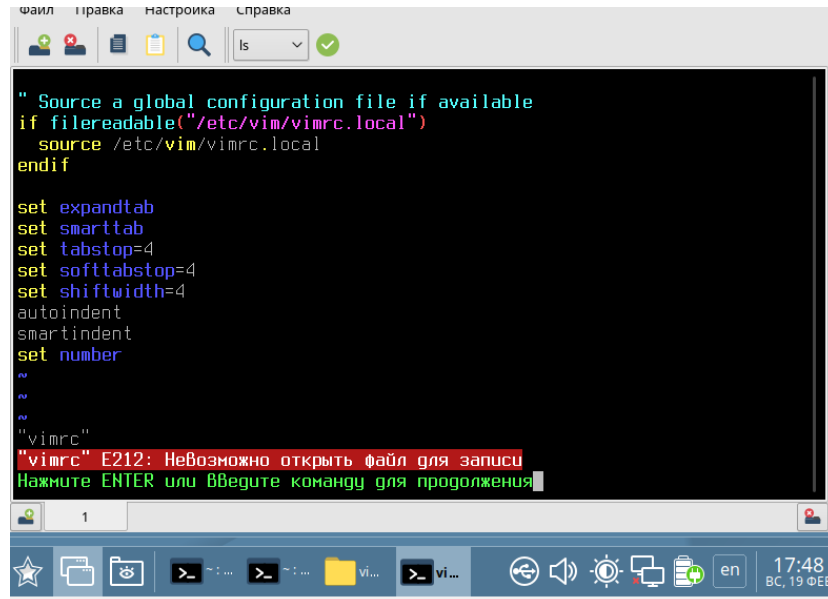


Рисунок 26 - Ошибка при сохранении

Была выведена ошибка. Перед тем как изменить файл, нужно изменить разрешения для чтения. С помощью команды `chmod ugo+rw` /etc/vim/vimrc. Результат представлен на рисунке 27.

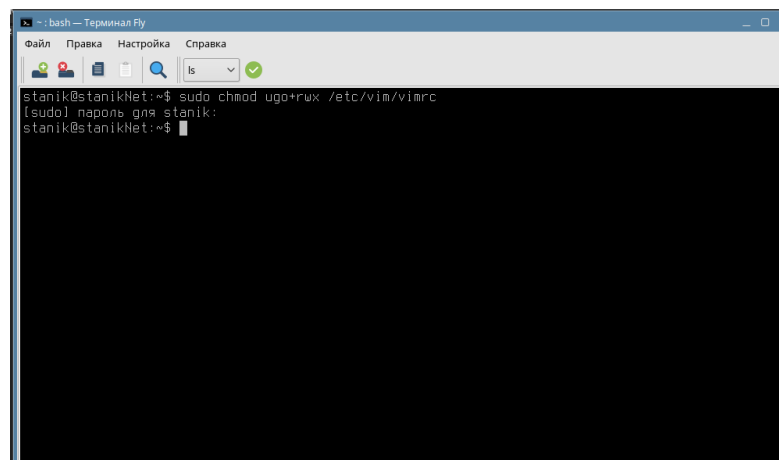


Рисунок 27- Изменение атрибутов

Попробуем теперь сохранить файл. Результат представлен на рисунке 28.


```
stanik@stanikNet:~$ sudo chmod ugo+rwX /etc/vim/vimrc
[sudo] пароль для stanik:
stanik@stanikNet:~$ vim /etc/vim/vimrc
stanik@stanikNet:~$
```

Рисунок 28 - Успешное выполнение операции

Операция прошла успешно.

Выберем цветовую схему для Vim. Для этого опять изменим конфигурационный файл и добавим в него строчку `colorscheme darkblue`. Результат представлен на рисунке 29.

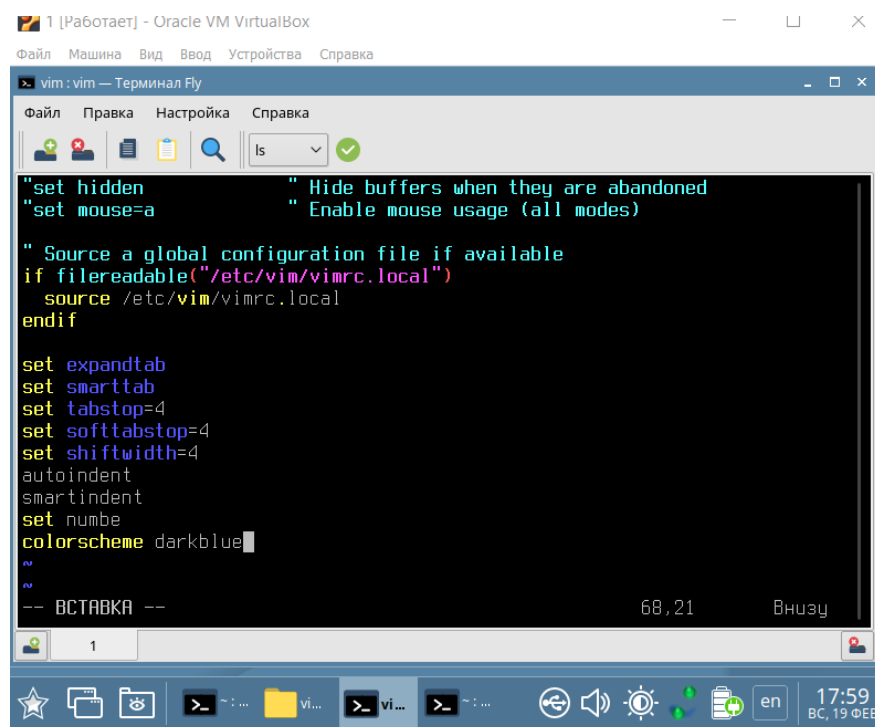


Рисунок 29 - Изменение конфигурационного файла

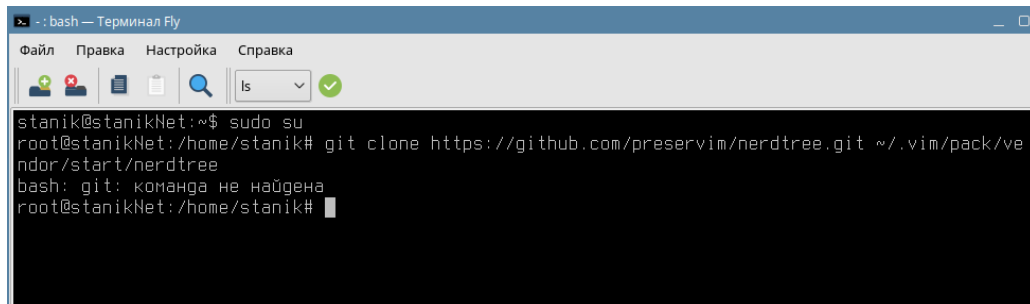
При нажатии неверной клавиши или ошибке в Vim проигрывается специальный звук. Отключим его, добавив в конфигурационный файл строки `set noerrorbells` и `set novisualbell`.

Также, добавим режим поддержки манипулятор типа «Мышь». Добавим в конфигурационный файл строку `set mouse=a`, т.е. включение поддержки мыши во всех режимах работы.

Сохраним файл командой `:wq!` и перейдем к установке плагинов.

Установим плагин NerdTree, который позволяет построить дерево

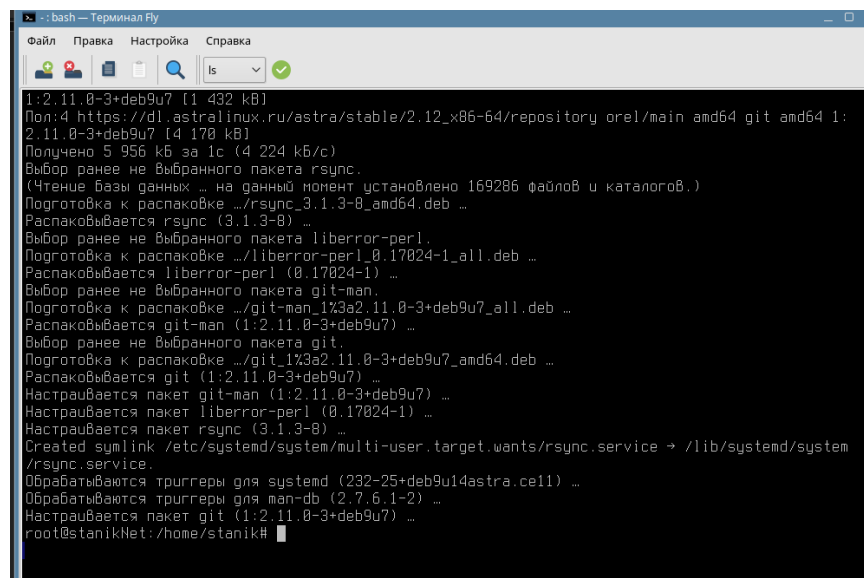
каталогов. Для этого склонируем его с GitHub командой `git clone https://github.com/preservim/nerdtree.git ~/.vim/pack/vendor/start/nerdtree`. Результат представлен на рисунке 30.



```
stanik@stanikNet:~$ sudo su
root@stanikNet:/home/stanik# git clone https://github.com/preservim/nerdtree.git ~/.vim/pack/vendor/start/nerdtree
bash: git: команда не найдена
root@stanikNet:/home/stanik#
```

Рисунок 30 - Ошибка при выполнении

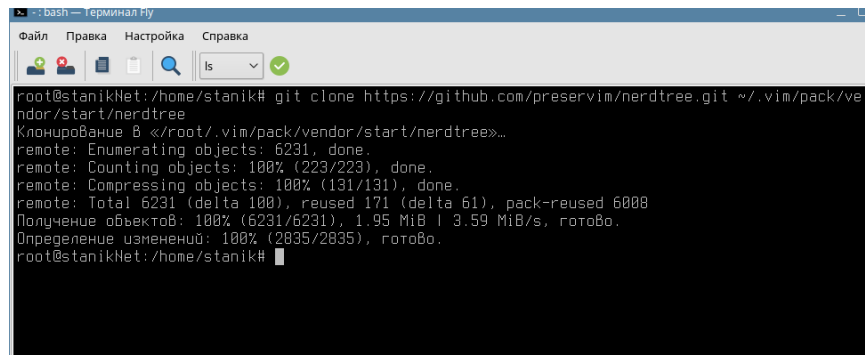
Для выполнения этой команды нужно установить Git. Сделаем это с помощью команды `sudo apt install git`. Результат представлен на рисунке 31.



```
1:2.11.0-3+deb9u7 [1 432 kB]
Пол:4 https://dl.astralinux.ru/astra/stable/2.12_x86-64/repository orcl/main amd64 git amd64 1:2.11.0-3+deb9u7 [4 170 kB]
Получено 5 956 kB за 1с (4 224 kB/c)
Выбор ранее не выбранного пакета rsync.
(Чтение базы данных ... на данный момент установлено 169286 файлов в каталогах.)
Подготовка к распаковке .../rsync_3.1.3-8_amd64.deb ...
Распаковывается rsync (3.1.3-8) ...
Выбор ранее не выбранного пакета liberror-perl.
Подготовка к распаковке .../liberror-perl_0.17024-1_all.deb ...
Распаковывается liberror-perl (0.17024-1) ...
Выбор ранее не выбранного пакета git-man.
Подготовка к распаковке .../git-man_1%3a2.11.0-3+deb9u7_all.deb ...
Распаковывается git-man (1:2.11.0-3+deb9u7) ...
Выбор ранее не выбранного пакета git.
Подготовка к распаковке .../git_1%3a2.11.0-3+deb9u7_amd64.deb ...
Распаковывается git (1:2.11.0-3+deb9u7) ...
Настраивается пакет git-man (1:2.11.0-3+deb9u7) ...
Настраивается пакет liberror-perl (0.17024-1) ...
Настраивается пакет rsync (3.1.3-8) ...
Created symlink /etc/systemd/system/multi-user.target.wants/rsync.service → /lib/systemd/system/rsync.service.
Обрабатываются триггеры для systemd (232-25+deb9u14astra.cell) ...
Обрабатываются триггеры для man-db (2.7.6.1-2) ...
Настраивается пакет git (1:2.11.0-3+deb9u7) ...
root@stanikNet:/home/stanik#
```

Рисунок 30 - Установка git

Попробуем выполнить команду. Результат представлен на рисунке 31.

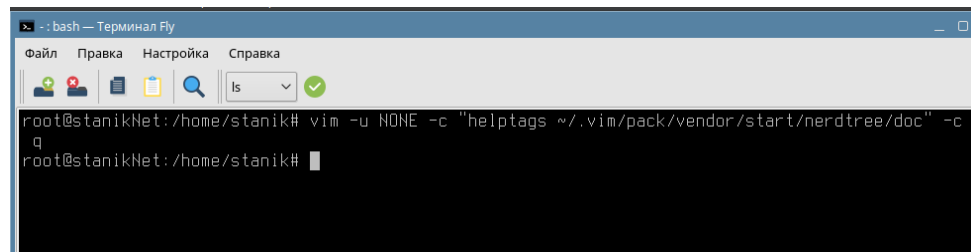


```
root@stanikNet:/home/stanik# git clone https://github.com/preservim/nerdtree.git ~/.vim/pack/vendor/start/nerdtree
Клонирование в «/root/.vim/pack/vendor/start/nerdtree»...
remote: Enumerating objects: 6231, done.
remote: Counting objects: 100% (223/223), done.
remote: Compressing objects: 100% (131/131), done.
remote: Total 6231 (delta 100), reused 171 (delta 61), pack-reused 6008
Получение объектов: 100% (6231/6231), 1.95 MiB | 3.59 MiB/s, готово.
Определение изменений: 100% (2835/2835), готово.
root@stanikNet:/home/stanik#
```

Рисунок 31 - Успешное выполнение операции

Операция выполнена успешно.

После этого, по рекомендации разработчиков, выполним команду `vim -u NONE -c "helptags ~/.vim/pack/vendor/start/nerdtree/doc" -c q`. Результат представлен на рисунке 32.



```
root@stanikNet:/home/stanik# vim -u NONE -c "helptags ~/.vim/pack/vendor/start/nerdtree/doc" -c q
root@stanikNet:/home/stanik#
```

Рисунок 32 - Успешное выполнение операции

Настроим Vim в качестве C/C++ IDE. Для начала, необходимо иметь место, где можно будет сохранять настройки для своего проекта. Для этого в конфигурационный файл добавим строчки `set exrc` и `set secure`. Результат представлен на рисунке 32.

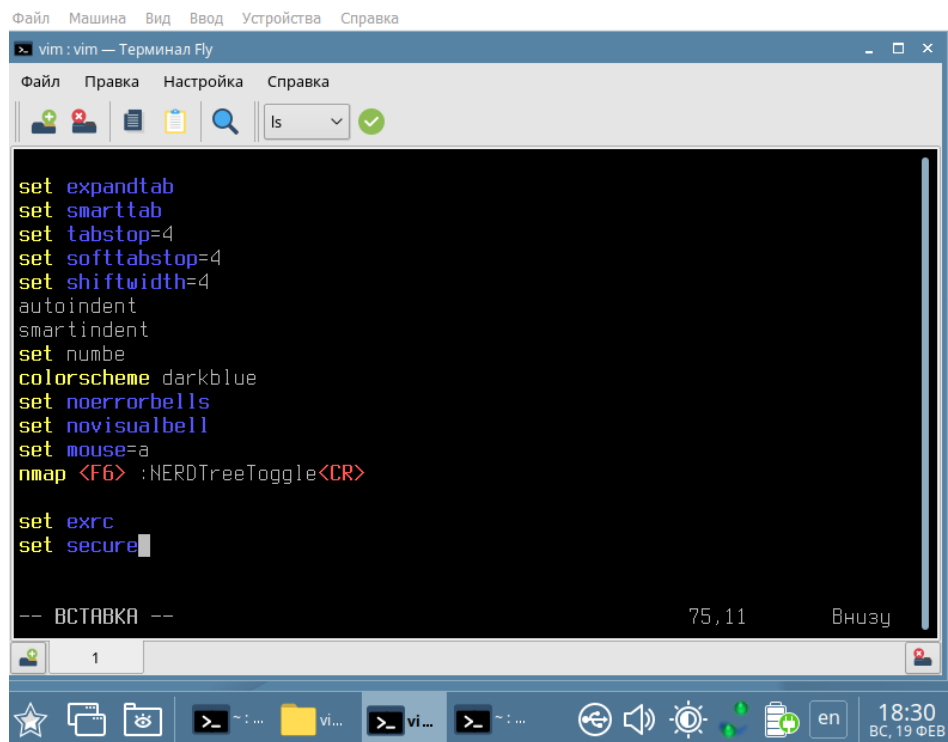


Рисунок 32 - Изменение конфигурационного файла

Используем возможность сохранять опции для каждого проекта в отдельности, т.к. у всех проектов есть какие-то свои правила касательно тех же отступов. Для этого создадим папку для проектов и создадим для проекта файл `.vimrc` со следующими строками:

```
set tabstop=4
set softtabstop=4
set shiftwidth=4
set noexpandtab
```

Результат представлен на рисунке 33.

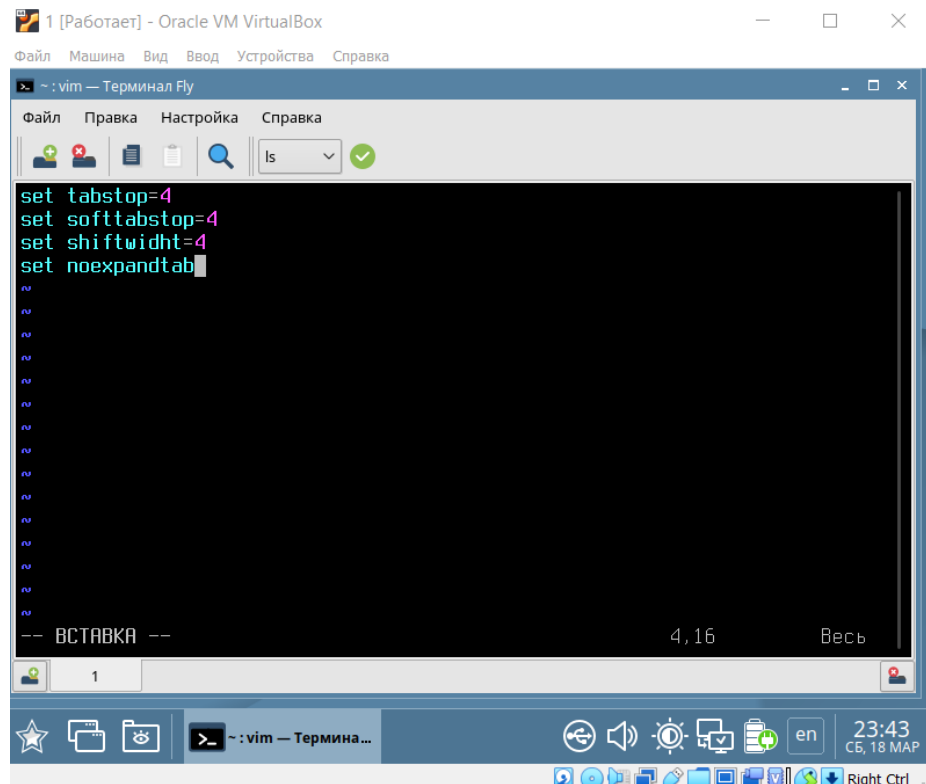


Рисунок 33 - Изменение локального конфигурационного файла

По умолчанию Vim предполагает, что все .h файлы – это C++. Однако, часть проектов могут быть написаны на чистом C. Для исправления этого добавим следующие строки в локальный .vimrc

```
augroup project
```

```
autocmd!
```

```
autocmd BufRead, BufNewFile *.h, *.c set filetype=c.doxxygen
```

```
augroup END
```

Результат представлен на рисунке 34.

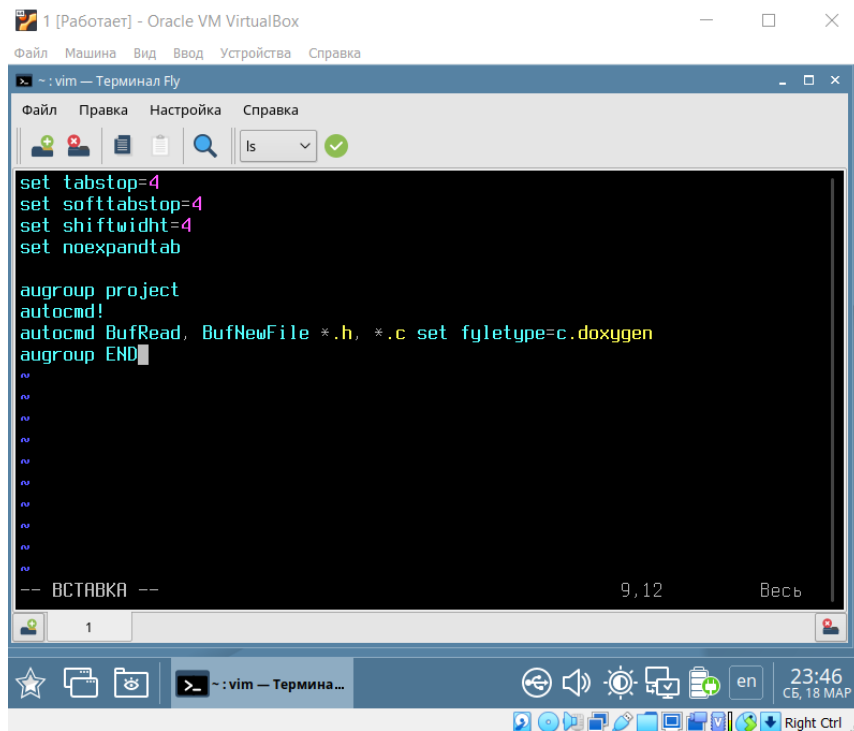


Рисунок 34 - Изменение локального конфигурационного файла

Установим переменную `path`. По умолчанию Vim смотрит файлы в текущей папке, а также в `/usr/include`. Однако почти все проекты имеют заголовочные файлы, которые хранятся в других папках. Поэтому, нужно установить переменную `path`, которая содержит список папок для поиска, разделенный запятыми. Добавим строчку в конфигурационный файл:

```
let &path.='src/include,/usr/include/AL,'
```

Результат представлен на рисунке 35.

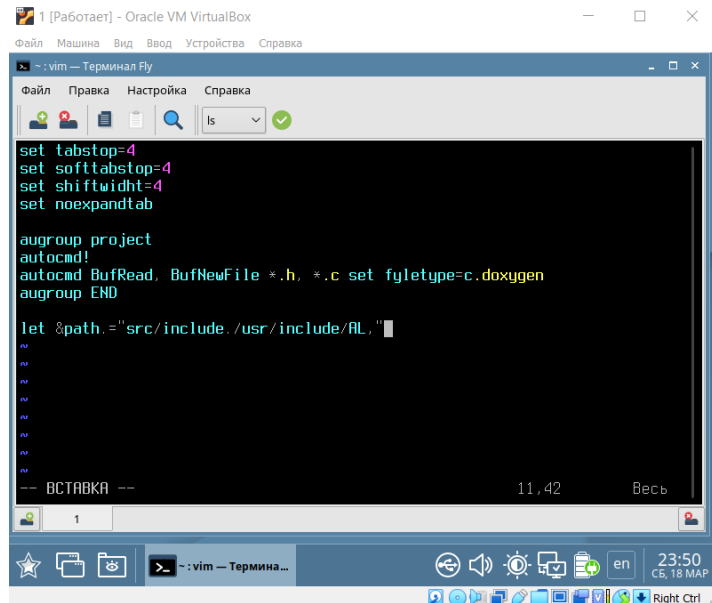


Рисунок 35 - Изменение локального конфигурационного файла

YouCompleteMe установить не удалось. Для этого необходим Vim скомпилированный с помощью Python. На AstraLinux произвести это не удалось.

Воспользуемся gcc для компиляции и запуска первой программы. Программа представлена на рисунке 36.

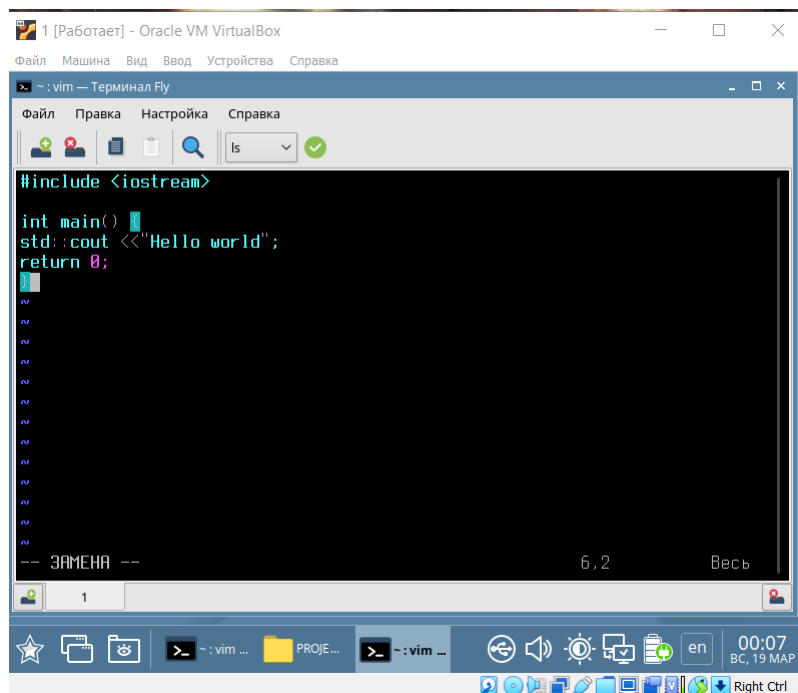


Рисунок 36 - Тестовая программа

Для быстрого доступа к команде добавим строчки в конфигурационный файл. Результат представлен на рисунке 37.

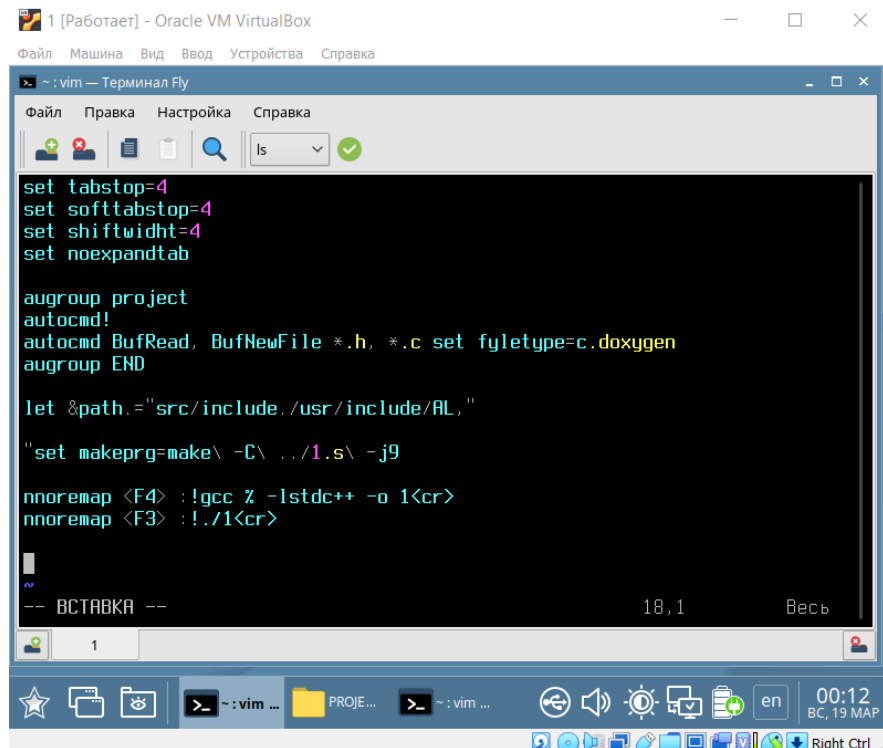


Рисунок 37 - Изменение локального конфигурационного файла
Нажмем F4 для компиляции. Результат представлен на рисунке 38.

В результате запуска компиляции кнопкой F4 программа перестала работать.

4. Контрольные вопросы

1. Что такое операционная система? Назовите основные компоненты ОС.

Операционная система — программное обеспечение, управляющее аппаратным обеспечением, предоставляющее абстрактный программный интерфейс для взаимодействия с ним и занимающееся распределением предоставляемых ресурсов, в том числе между прикладными программами. В широком смысле под операционной системой понимается совокупность ядра операционной системы и работающих поверх него программ и утилит, предоставляющих интерфейс для взаимодействия пользователя с компьютером.

В составе операционной системы различают три группы компонентов:

- ядро, содержащее планировщик; драйверы устройств, непосредственно управляющие оборудованием; сетевая подсистема, файловая система;
- системные библиотеки;
- оболочка с утилитами.

2. Дайте определение понятию виртуализация.

Виртуализация — это изоляция вычислительных процессов и ресурсов друг от друга. Это новый виртуальный взгляд на ресурсы составных частей, не ограниченных реализацией, физической конфигурацией или географическим положением. Обычно виртуализированные ресурсы включают в себя вычислительные мощности и хранилище данных. В широком смысле, понятие виртуализации представляет собой сокрытие настоящей реализации какого-либо процесса или объекта от истинного его представления для того, кто им пользуется. В компьютерных технологиях под термином «виртуализация» обычно понимается абстракция вычислительных ресурсов и предоставление пользователю системы, которая «инкапсулирует» (скрывает в себе) собственную реализацию. Проще говоря, пользователь работает с удобным

для себя представлением объекта, и для него не имеет значения, как объект устроен в действительности.

3. Какие есть виды виртуализации? Охарактеризуйте каждый вид.

Понятие виртуализации условно можно разделить на две фундаментально различающиеся категории:

1. Виртуализация платформ

Продуктом этого вида виртуализации являются виртуальные машины - программные абстракции, запускаемые на платформе реальных аппаратно- программных систем.

2. Виртуализация ресурсов

Данный вид виртуализации преследует своей целью комбинирование или упрощение представления аппаратных ресурсов для пользователя и получение неких пользовательских абстракций оборудования, пространств имен, сетей и т.п.

Под виртуализацией платформ понимают создание программных систем на основе существующих аппаратно-программных комплексов, зависящих или независящих от них. Система, предоставляющая аппаратные ресурсы и программное обеспечение, называется хостовой (host), а симулируемые ей системы – гостевыми (guest). Чтобы гостевые системы могли стабильно функционировать на платформе хостовой системы, необходимо, чтобы программное и аппаратное обеспечение хоста было достаточно надежным и предоставляло необходимый набор интерфейсов для доступа к его ресурсам.

Виртуальная машина (virtual machine):

- программная и/или аппаратная система, эмулирующая аппаратное обеспечение некоторой платформы (target — целевая, или гостевая платформа) и исполняющая программы для target-платформы на host-платформе (host — хост-платформа, платформа-хозяин);

- или виртуализирующая некоторую платформу и создающая на ней среды, изолирующие друг от друга программы и даже операционные системы (песочница, sandbox).

4. На какие виды подразделяется виртуализация платформ?

Есть несколько видов виртуализации платформ, в каждом из которых осуществляется свой подход к понятию «виртуализация».

Полная эмуляция (симуляция)

При таком виде виртуализации виртуальная машина полностью виртуализует все аппаратное обеспечение при сохранении гостевой операционной системы в неизменном виде. Такой подход позволяет эмулировать различные аппаратные архитектуры. Основным минус данного подхода заключается в том, что эмулируемое аппаратное обеспечение весьма и весьма существенно замедляет быстроедействие гостевой системы, что делает работу с ней очень неудобно.

Частичная эмуляция (нативная виртуализация)

В этом случае виртуальная машина виртуализирует лишь необходимое количество аппаратного обеспечения, чтобы она могла быть запущена изолированно. Такой подход позволяет запускать гостевые операционные системы, разработанные только для той же архитектуры, что и у хоста. Таким образом, несколько экземпляров гостевых систем могут быть запущены одновременно. Этот вид виртуализации позволяет существенно увеличить быстроедействие гостевых систем по сравнению с полной эмуляцией и широко используется в настоящее время. Также, в целях повышения быстроедействия, в платформах виртуализации, использующих данный подход, применяется специальная «прослойка» между гостевой операционной системой и оборудованием (гипервизор), позволяющая гостевой системе напрямую обращаться к ресурсам аппаратного обеспечения. Гипервизор, называемый также «Монитор виртуальных машин» (Virtual Machine Monitor) - одно из ключевых понятий в мире

виртуализации.

Частичная виртуализация, а также «виртуализация адресного пространства» При таком подходе, виртуальная машина симулирует несколько экземпляров аппаратного окружения (но не всего), в частности, пространства адресов. Такой вид виртуализации позволяет совместно использовать ресурсы и изолировать процессы, но не позволяет разделять экземпляры гостевых операционных систем. Строго говоря, при таком виде виртуализации пользователем не создаются виртуальные машины, а происходит изоляция каких-либо процессов на уровне операционной системы.

Паравиртуализация. При применении паравиртуализации нет необходимости симулировать аппаратное обеспечение, однако, вместо этого (или в дополнение к этому), используется специальный программный интерфейс (API) для взаимодействия с гостевой операционной системой.

Виртуализация уровня операционной системы. Сутью данного вида виртуализации является виртуализация физического сервера на уровне операционной системы в целях создания нескольких защищенных виртуализованных серверов на одном физическом. Гостевая система, в данном случае, разделяет использование одного ядра хостовой операционной системы с другими гостевыми системами. Виртуальная машина представляет собой окружение для приложений, запускаемых изолированно. Данный тип виртуализации применяется при организации систем хостинга, когда в рамках одного экземпляра ядра требуется поддерживать несколько виртуальных серверов клиентов.

Виртуализация уровня приложений. Этот вид виртуализации не похож на все остальные: если в предыдущих случаях создаются виртуальные среды или виртуальные машины, использующиеся для

изоляции приложений, то в данном случае само приложение помещается в контейнер с необходимыми элементами для своей работы: файлами реестра, конфигурационными файлами, пользовательскими и системными объектами. В результате получается приложение, не требующее установки на аналогичной платформе. При переносе такого приложения на другую машину и его запуске, виртуальное окружение, созданное для программы, разрешает конфликты между ней и операционной системой, а также другими приложениями. Такой способ виртуализации похож на поведение интерпретаторов различных языков программирования (недаром интерпретатор, Виртуальная Машина Java (JVM), тоже попадает в эту категорию).

5. Что такое гипервизор?

Гипервизор — технология развертывания программного обеспечения на физическом оборудовании с использованием виртуализации.

Инструмент ускоряет и упрощает разработку, тестирование и поддержку программного обеспечения, а также экономит ресурсы на развертывании дорогостоящих серверных систем.

6. Что такое аппаратная виртуализация?

Аппаратная виртуализация — виртуализация с поддержкой специальной процессорной архитектуры. В отличие от программной виртуализации с помощью данной техники возможно использование изолированных гостевых операционных систем, управляемых гипервизором напрямую.

7. Что такое «виртуальная машина»? Назначение виртуальной машины.

Виртуальная машина - программная и/или аппаратная система, эмулирующая аппаратное обеспечение некоторой платформы (target — целевая, или гостевая платформа) и исполняющая программы для target-платформы на host-платформе (host — хост-платформа, платформа-хозяин).

8. Что такое хост-платформа?

Хост – платформа – платформа, на которой запускается гостевая ОС на виртуальной машине

9. Дайте определение гостевой ОС.

Гостевая ОС – ОС, которая запускается через гипервизор второго типа на хост – платформе.

10. Дайте определение понятию песочницы («sandbox»).

Sandbox – это среда безопасного тестирования. Решение изолирует непроверенные изменения в коде и эксперименты от производственной среды и хранилища в контексте разработки программного обеспечения.

11. Какие продукты для виртуализации вы знаете?

Oracle VM VirtualBox, VMWare Workstation, Hyper-V.

12. Можно ли запустить несколько гостевых ОС на одном хосте?

Можно, но с каждой запущенной гостевой ОС аппаратные ресурсы будут занимать гостевыми, в следствие чего может наблюдаться медленная работа хостовой ОС.

13. Какие системы относятся к Unix, а какие системы относятся к Unix- подобным?

Unix-подобная операционная система — операционная система, которая образовалась под влиянием Unix. Термин включает свободные/открытые операционные системы, образованные от Unix компании Bell Labs или эмулирующие его возможности, коммерческие и запатентованные разработки, а также версии, основанные на исходном коде Unix.

Unix — семейство переносимых, многозадачных и многопользовательских операционных систем, которые основаны на идеях оригинального проекта AT&T Unix, разработанного в 1970-х

годах в исследовательском центре Bell Labs Кеном Томпсоном, Деннисом Ритчи и другими.

Операционные системы семейства Unix характеризуются модульным дизайном, в котором каждая задача выполняется отдельной утилитой, взаимодействие осуществляется через единую файловую систему, а для работы с утилитами используется командная оболочка.

14. Что означает GNU GPL?

GNU General Public License (переводят как Универсальная общественная лицензия GNU, Универсальная общедоступная лицензия GNU или Открытое лицензионное соглашение GNU) — лицензия на свободное программное обеспечение, созданная в рамках проекта GNU в 1988 г., по которой автор передаёт программное обеспечение в общественную собственность.

15. Какие системы относятся к Windows-подобным?

Системы семейства Microsoft Windows (Windows XP, Windows 7, Windows 10).

16. Какие современные операционные системы относятся к Unix-подобным операционным системам?

Есть множество запатентованных Unix-подобий, таких, как: AIX, HP-UX, IRIX, macOS, LynxOS, QNX, SCO OpenServer, Solaris, Tru64 UNIX, UnixWare, Xenix и VxWorks.

17. Какие современные операционные системы относятся к Windows-подобным операционным системам?

Windows XP, Windows NT, Windows 7 и т.д.

18. Кто является создателем ядра Linux?

Автор ядра Linux – Линус Торвальдс

19. Что такое ISO-образ?

ISO-образ — это неформальный термин для обозначения образа оптического диска, содержащего файловую систему стандарта ISO 9660. В

более общем смысле, термин относится к образу любого оптического диска. Этот образ представляет собой файл с расширением .iso. Его можно использовать (в совокупности со специальными программными средствами) вместо компакт-диска.

20. Что такое виртуальный жесткий диск? Какие типы жестких дисков поддерживает выбранный вами для выполнения данной работы гипервизор второго типа. Кратко охарактеризуйте каждый тип.

Виртуальные жесткие диски — это файлы виртуальных жестких дисков, которые при подключении отображаются и функционируют практически идентично физическому жесткому диску. В основном они используются в сочетании с виртуальными машинами Hyper-V.

VBox поддерживает типы виртуальных носителей:

- 1) VDI. Собственный тип, формат контейнера жёсткого диска от VBox
- 2) VMDK. Популярный открытый формат контейнера, используется многими другими продуктами виртуализации
- 3) VHD. Тип виртуального носителя, который использует Microsoft
- 4) HDD. Также поддерживаются файлы изображений Parallels версии 2

21. Какие типы архитектур файловых систем поддерживает выбранный вами для установки в качестве выполнения данной работы дистрибутив ОС?

Linux поддерживает множество разных файловых систем, включая FAT, FAT32, NTFS из Windows, а также Ext3, Ext4, ReiserFS, XFS, Btrfs и пр.

22. Какие типы архитектур файловых систем поддерживает ОС Ubuntu актуальной версии? Кратко охарактеризуйте каждую из них (преимущества/недостатки).

Пользователи могут использовать одну или несколько файловых систем, зашитых в линуксовый дистрибутив. Так как от них зависит метод работы с файлами, конфигурацией ядра и особенности хранения информации в оперативной памяти, то выбирать file system нужно исходя из текущих задач, учитывая сильные и слабые стороны доступных ФС.

Выделяют два типа таких систем, журналируемые и не журналируемые. Определить тип можно при помощи команды `file -s`. ФС первого типа ведут логи, фиксируя в отдельном файле информацию о действиях пользователя и план проверки системы. За счёт логирования система получается более устойчивой к сбоям. ФС второго типа не имеют логов. Они обладают хорошим быстродействием, но более уязвимы, так как не обеспечивают сохранность данных.

Часть 2

1. Что такое IDE?

Интегрированная среда разработки (IDE) – это программное приложение, которое помогает программистам эффективно разрабатывать программный код. Оно повышает производительность разработчиков, объединяя такие возможности, как редактирование, создание, тестирование и упаковка программного обеспечения в простом для использования приложении. Так же как писатели используют текстовые редакторы, а бухгалтеры – электронные таблицы, разработчики программного обеспечения применяют IDE, чтобы упростить свою работу.

2. Что такое API?

API – это механизмы, которые позволяют двум программным компонентам взаимодействовать друг с другом, используя набор определений и протоколов. Например, система ПО метеослужбы содержит ежедневные данные о погоде. Приложение погоды на телефоне «общается» с этой системой через API и показывает ежедневные обновления погоды на телефоне.

3. Что такое библиотека в программировании?

Библиотека (от англ. library) в программировании — сборник подпрограмм или объектов, используемых для разработки программного обеспечения (ПО). С точки зрения операционной системы (ОС) и прикладного ПО, библиотеки разделяются на динамические и статические.

4. Понятия Статической и Динамической библиотек.

Динамическая библиотека — файл, содержащий машинный код. Загружается в память процесса загрузчиком программ операционной системы либо при создании процесса, либо по запросу уже работающего процесса, то есть динамически

Статическая библиотека — объектный файл в виде файла (нередко может быть поставлен вместе с исходным кодом), код из которого выборочно или полностью вставляется в программу на этапе компоновки.

5. Что такое плагин?

Плагин — независимо компилируемый программный модуль, динамически подключаемый к основной программе и предназначенный для расширения и/или использования её возможностей. Плагины обычно выполняются в виде библиотек общего пользования.

6. Назовите несколько консольных текстовых редакторов для Linux.

Vim, Nano, vi.

7. Что делает команда gcc?

Команда gcc предназначена для компиляции с помощью компилятора GCC кода на языке C. Данная команда похожа на команду g++, используемую для компиляции кода на языке C++. В качестве имен файлов могут использоваться как имена файлов исходного кода на языке C с расширением.

8. Что делает команда `make`?

`make` — утилита предназначенная для автоматизации преобразования файлов из одной формы в другую. Правила преобразования задаются в скрипте с именем `Makefile`, который должен находиться в корне рабочей директории проекта.

9. Что делает команда `gdb`?

Программа `gdb` - это популярный отладчик для программ, написанных на языке программирования Си и C++. Отладчик позволяет выполнить программу пошагово, посмотреть значения всех переменных на каждом из этапов выполнения, а если это необходимо, то и дизассемблировать код и посмотреть состояние регистров процессора.

10. Дайте определение заголовочного файла и файла реализации. Приведите пример.

В C / C++ текст программы разделяется на множество клочков, каждый из которых хранится в отдельном текстовом файле. На первом этапе сборки препроцессор начинает читать это отдельные файлы заменяя директивы `#include` на содержимое соответствующих текстовых файлов и получая на выходе длинный текстовый документ единицы трансляции, который затем передается компилятору.

Те текстовые файлы, которые включаются в другие текстовые файлы при помощи директивы `#include` обычно называют заголовочными файлами и используют для них расширения `.h` `.hpp` `.hxx` `.inl` и подобные (многие заголовочные файлы из стандартной библиотеки вообще не имеют расширения). А те текстовые файлы, с которых начинается построение единицы трансляции (соответственно эти файлы обычно передаются среди аргументов командной строки компилятору) обычно имеют расширение `.c` `.cpp` `.cc` или `.cxx`.

11. Дайте краткую характеристику каждому этапу трансляции программ, написанных на Си.

Рассмотрим схему трансляции программы на языке Си, которая традиционно используется в системах Unix.

Трансляция программы состоит из следующих этапов: препроцессирование (1); трансляция в ассемблер (2); ассемблирование (3); компоновка (4).

Традиционно исходные файлы программы на языке Си имеют суффикс имени файла .c, заголовочные файлы для программы на Си имеют суффикс .h. В файловых системах Unix регистр букв значим, и если, например, имя файла имеет суффикс .C, такой файл считается содержащим текст программы на языке Си++, и будет компилироваться компилятором языка Си++, а не Си.

Препроцессирование (1). Препроцессор просматривает входной .c файл, исполняет содержащиеся в нём директивы препроцессора, в частности, включает в него содержимое других файлов, указанных в директивах `#include`.

Файл-результат препроцессирования не содержит директив препроцессора, не раскрытых макросов, вместо директив `#include` в файл-результат подставлено содержимое соответствующих заголовочных файлов. Файл с результатом препроцессирования обычно имеет суффикс .i, однако после завершения трансляции все промежуточные временные файлы по умолчанию удаляются, поэтому чтобы увидеть результат препроцессирования (что бывает полезно при отладке ошибок, связанных с небрежным использованием макросов) нужно использовать опцию -E командной строки gcc. Результат препроцессирования называется единицей трансляции (англ., translation unit) или единицей компиляции (англ., compilation unit).

Трансляция в ассемблер (2). На вход подаётся одна единица трансляции, а на выходе (при отсутствии синтаксических и семантических ошибок) выдаётся файл на языке ассемблера для (как

правило) машины, на которой ведётся трансляция. Файл с оттранслированной программой на языке ассемблера имеет суффикс имени .s, но точно так же, как и результат работы препроцессора, он по умолчанию удаляется.

Ассемблирование (3). На этой стадии работает ассемблер. Он получает на входе результат работы предыдущей стадии и генерирует на выходе объектный файл. Объектные файлы в UNIX имеют суффикс .o.

Компоновка (4). Компоновщик получает на вход набор объектных файлов, соответствующим единицам трансляции, составляющим программу, подключает к ним стандартную библиотеку языка Си и библиотеки, указанные пользователем, и на выходе получает исполняемую программу.

5. Вывод

В ходе выполнения данной лабораторной работы я ознакомился с программными продуктами для виртуализации, научился устанавливать на виртуальную машину ОС и получил навыки её настройки на примере ОС Astra Linux.