Руководство

по настройке компьютера инженера

Денис Тельнов

Содержание

Version: 0.5.7

Этап 1: Установка базового набора рабочих программ

> *Preface: без использования SSD нормальной работы разработчика не получится, даже если удастся (хватит терпения) установить все необходимые программы :(*

1 Установка Windows 11 и MSOffice

1.1 Информация о продуктах Microsoft

1.1.1 Лицензия

В последние годы стало возможным приобрести недорогие OEM лицензии на продукты Microsoft, на Озоне или Вайлдбериз, поэтому у пользователей появился выбор между покупкой “коробочных” (Retail) лицензий, и более дешёвых OEM лицензий.

Лицензии OEM — это лицензии на ПО, поставляемые производителями и сборщиками техники вместе с новыми ноутбуками и компьютерами. Microsoft выделяет производителям техники целые пулы OEM лицензий по небольшой цене. И, по Европейским законам, производители могут продавать не только продукт целиком, но и комплектующие к нему, и, в данном случае, они могут продавать эти лицензии отдельно от компьютерной техники. Каждая OEM лицензия входит в какой-то конкретный пул лицензий, который закреплён за конкретным производителем, т.е., OEM, это не какая-то абстрактная безликая лицензия, а выпущенная определённым производителем. В целом, покупка OEM лицензии достаточно безопасна, случаев отзыва пулов лицензий очень и очень немного. Но, как и любую другую электронную продукцию, OEM лицензии необходимо покупать только у проверенных продавцов со множеством отзывов, но это обычные издержки электронной торговли. Из особенностей OEM лицензии следует отметить, что она, в отличии от Retail или Volume лицензий, “одноразовая”, то есть, активировав её один раз, *ключ* нельзя применить второй раз, и программу нельзя будет переустановить или перенести на другой компьютер.

Подробнее про лицензии Microsoft в Приложении.

1.1.2 Выбор версий продуктов Microsoft

Поддержка Windows 10 закончится осенью 2025, после чего, скорее всего, интернет заполонят вирусы для этой версии Windows из-за отсутствия обновлений безопасности. Поэтому рекомендуется изначально устанавливать Windows 11.

Для русскоговорящих пользователей следует устанавливать следует *En* версию Windows 11 Pro. А русский язык можно будет добавить после установки, даже полностью сменить интерфейс на русский. Если устанавливать сразу *Ru* версию, то в процессе установки будут созданы папки с русскими буквами в путях (*Документы* и т.д.), из-за чего не будут работать некоторые инженерные (*HPC*) программы. По этой же причине, в случае установки Windows версии Home, также следует устанавливать English версию, даже не смотря на то, что в Home-версии Windows язык интерфейса сменить не удастся. Так же, не следует использовать кириллицу в имени пользователя — это гарантированный способ сломать работоспособность инженерных программ, компиляторов; это настолько катастрофично, что, рано или поздно, но систему придётся переустановить.

Для установки рекомендуется использовать версию *build 22631.3296* (*23H2*). Это версия, с одной стороны, достаточно свежая, чтобы были драйвера для новых моделей ноутбуков, а с другой стороны, эта версия всё ещё позволяет при установке создать локальную учётную запись, и провести установку при отсутствии интернета.

Note: Для создания локальной учётной записи можно использовать и более новые версии Windows, при создании загрузочной флешки при помощи утилиты *Rufus*, см. следующий пункт.

Выбор версии и дистрибутива MSOffice:

- если не используется MS Access, то Office Home или Student *абсолютно достаточно*;
- если уже установлен офис версии 2016 (или новее) и он устраивает, то обновляться не требуется;
- если офис не установлен, то следует устанавливать самую последнюю версию (2024) с канала *Current*.

Подробности в *Приложении Выбор версии MSOffice*.

1.1.3 Как получить дистрибутивы Windows и MSOffice

На данный момент все инсталяторы для продуктов Microsoft загружаются через интернет, это официальный рекомендуемый способ установки — диски отошли в прошлое. Ссылки на загрузку Windows, загрузку MS Office.

На самом деле, приведённые выше ссылки не работают. Во-первых, из-за санкций, которые ограничивают загрузку напрямую, непосредственно с сайта Microsoft (хотя никакого запрета на продажу или использование лицензий Microsoft не существует). Во-вторых, из-за того, что Microsoft продвигает только самую последнюю версию *Windows 11* (глючный *24H2*?), и только онлайн-офис *Office 365*, то, нормальную версию *Windows*, и обычный автономный *MSOffice* не загрузить напрямую со страниц сайта Microsoft.

Поэтому, в данном руководстве в *Приложении* приводятся специальные процедуры загрузки продуктов с сайта Microsoft:

- UUP (Unified Update Platform) — загрузка MS Windows;
- ODT (Office Deployment Tool) — загрузка MS Office.

Подготовить загрузочную флешку из полученного iso-образа можно с помощью утилиты **Rufus** под Windows, либо **mkusb** под Linux.

Rufus позволяет при создании загрузочной флешки сделать *Customize Windows Installation*, а именно *запрет* на: проверку ограничений, сбор данных (телеметрию), онлайн регистрацию. При выборе типа загрузки следует выбирать *GPT/SecureBoot*, а не *MBR/Legacy* (который предназначен для очень старых компьютеров). *Note:* Запрет на онлайн регистрацию означает, что будет создана локальная учётная запись, и что установку Windows возможно провести без подключения к сети интернет.

1.2 Установка и настройка Windows

Note: BitLocker — это встроенный в Windows механизм шифрования дисков для обеспечения безопасности информации. По умолчанию, при использовании учётной записи Microsoft, диски были автоматически зашифрованы. И, в случае переустановки Windows важно убедиться, что ключи шифрования сохранены и доступны для использования. При установке Windows на новые диски это не актуально. Подробности в *Приложении BitLocker*.

Для начала установки необходимо загрузиться с загрузочной флешки с инсталятором Windows. Для установки потребуется подключение к интернет для входа в учётную запись Microsoft (или создания новой), если только заранее не были предприняты специальные действия.

<*Upd 2025*>: Создать локальную учётную запись в процессе установки можно следующими командами: во время запроса на ввод учётной записи нажать Shift + F10, открыть командную строку, набрать и выполнить команду start ms-cxh:localonly. Нет гарантий, что этот метод в будущем не перестанет работать.

После установки потребуется единоразово понастраивать свойства и интерфейс Windows:

- Разрешить *Windows Development Mode*: (~~*Settings > Update & Security > For Developers:*~~) *Settings > System > For Developers: Developer Mode -> On*
- Также, для разработчиков, для поддержки длинных имён файлов выполнить в PowerShell(Admin):

```
Set-ItemProperty 'HKLM:\SYSTEM\CurrentControlSet\Control\FileSystem' -Name 'LongPathsEnabled' -Value 1
```

- После активации появится возможность переместить меню Старт влево: *Setting > Personalization > TaskBar > Taskbar behaviors > Taskbar alignment*.
- Отключить режим гибернации (hibernate): в Cmd/PowerShell(Admin) выполнить:

```
powercfg /H off
```

- После установки региона *Россия* Windows настроит все соответствующие параметры, в том числе, установит запятую в качестве десятичного разделителя. Поправить на точку можно в *ControlPannel*, раздел *Region* > *Additional Settings...* > *Decimal Symbol*.
- Настройка действия на складывание ноутбука — поиском в меню Start слова *Lid*. Рекомендуется настроить везде Sleep, и при работе от аккумулятора, и при работе от сети. Современный ноутбук при постоянном использовании можно вообще никогда не выключать, а только переводить в сон.
- Поведение <Alt+Tab> — показывать ли отдельные *Табы* в списке окон по <Alt+Tab>: *Settings* > *System* > *Mutitasking: Show tabs from apps...*, установить в *Don't show tabs*, тогда по <Alt+Tab> будут показываться только окна, без составляющих их табов.
- Включить встроенный многокарманный буфер обмена (Clipboard) в Windows — Win+V (*Note:* этот буфер очищается при перезагрузке!, поэтому, лучше использовать сторонний буфер обмена CopyQ).
- *Windows Update* > *Advanced Options* > включить *Receive Updates for other Microsoft products*. Это чтобы MSOffice тоже обновлялся.
- После установки всех программ отредактировать список программ запускаемых при загрузке: *Settings* > *Apps* > *StartUp*.
- Правой клавишей по трею > *Taskbar settings* > *Other system tray icons*, выбрать постоянно показываемые иконки:
  Safely Remove Hardware...
  Windows Update Status
  Плюс ещё какие нужные программы, а остальные спрятать чтобы не мешали.
- Настройка вида меню Start: *Settings* > *Personalization* > *Start*.
- В меню Start в нижней строке можно расположить быстрый доступ к Папкам и *Settings*: *Personalization* > *Start* > *Folders*.
- Если Edge не используется, то есть пара утилит на случай, если Edge станет совсем занудным своими напоминаниями: GoAwayEdge и MSEdgeRedirect, которые отключают Edge и регулярно проверяют, чтобы Edge больше не просыпался.
- Высокое время задержки перед повтором нажатия клавиш (если не помогает стандартный метод *ControlPanel* > *Keyboard: Repeat delay*) правится в реестре:
  regedit: HKEY_CURRENT_USER>Control Panel>Accessibility>Keyboard Response:

  ```
  AutoRepeatDelay 200
  AutoRepeatRate 5
  BounceTime 0
  DelayBeforeAcceptance 10
  Flags 59
  ```

  Эти настройки могут слетать (при каждом обновлении Windows11?), поэтому, если ситуация будет повторяться, то надо сделать *.reg* файл для этого раздела регистра, и применять

его, когда задержка перед повтором опять станет длительной. См. также Настройки. В общем, с этой задержкой перед повтором нажатий клавиш в Windows 11 какой-то баг, и, я полагаю, проблема разрешится с очередным обновлением.

- После полной переустановки Windows, если были данные на диске D: и т.д., то может возникнуть проблема с доступом к старым данным, которые обусловлены несовпадением новых идентификаторов пользователя со старыми идентификаторами. В этом случае необходимо установить права доступа по умолчанию (подробности *icacls*), для этого запустить с правами Администратора команду:

```
icacls D:\ /reset /t /c /l
```

- Для продвинутых пользователей есть утилита Winutil с широкими возможностями по установке и удаления утилит и программ, по тюнингу и настройке Windows, отключения телеметрии, и прочему. Установка в PowerShell от Администратора (Admin!):

```
Set-ExecutionPolicy Unrestricted -Scope Process -Force
irm https://github.com/ChrisTitusTech/winutil/releases/latest/download/winutil.ps1 | iex
```

  Запуск через меню *Пуск*: Winutil.
  Перед применением рекомендуется сделать бекап системы.

- Для продвинутых пользователей есть утилита Win11Debloat для отключения телеметрии и удаления ненужных предустановленных программ. Этот скрипт имеет три режима работы, в одном из которых (*3-ем*) предлагается список предустановленного софта, где можно выбрать галочками что удалить а что оставить. Перед применением *крайне* рекомендуется сделать бекап системы.

2 Справочные материалы по системе Windows

Windows 10/11 Guide. Including Windows Security tools, Encryption, Nextcloud, Graphics, Gaming, Virtualization, Windows Subsystem for Linux (WSL 2), Software Apps, and Resources.

Awesome list dedicated to Windows Subsystem for Linux.

2.1 Клавиши быстрого доступа (shortcut keys) в Windows

Полный список комбинаций клавиш быстрого доступа Windows, ещё один, и ещё один.

2.1.1 Переключение окон

Win+#Digit: Win+1, Win+2, и т.д. — быстрое переключение на приложение номер *#Digit* в *Панели Задач*/*Taskbar*. Если в *Панели Задач* закреплено (*pin*) несколько приложений, например: *FileExplorer*, *Edge* и *Far*, то, нажатие на Win+3 переключит на первый открытый *Far*, сколько бы ни было открыто слева окон проводников или браузеров. Повторное нажатие на Win+3 переключит на следующий открытый *Far*, и т.д.

То есть, закрепив в *Панели Задач* наиболее часто используемые приложения, их всегда можно сходу вызвать по их порядковому номеру соответствующей горячей клавишей Win+Номер.

2.1.2 Переключение вкладок

В Windows во всех приложениях быстрый доступ к конкретной *Вкладке*/*Табу* (*Tab*) проводится по комбинации Ctrl+#Digit, т.е. Ctrl+1, Ctrl+2 и т.д. (кроме *Windows Terminal*, где Ctrl+Alt+1)

По Ctrl+Tab — листание *Вкладок*/*Табов*. Ctrl+Shift+Tab — листание *Вкладок* в обратном порядке.

2.1.3 Остальные команды

- Ctrl+D — свернуть все окна
- Ctrl+Esc — меню Start (Пуск)
- Ctrl+Shift+Esc — вызвать Task Manager (Менеджер Задач)
- Win+E — открыть новый File Explorer (Проводник)

Note: Guide, как включить или выключить те или иные комбинаций клавиш быстрого доступа Windows.

Note: Для очень продвинутых пользователей может быть интересна утилита AutoHotKey, которая позволяет запрограммировать любые действия на любые клавиши.

3 Установка программ

Менеджеры программ для Windows: **Scoop** (консольная), **WinGet** (консольная), **Chocolatey** (есть GUI). Потребуются все три менеджера, так как не все программы есть в каком-то одном менеджере. На сайте https://repology.org/projects/ есть сведения практически обо всех существующих программах распространяемых через интернет. Также там можно определить в каких репозиториях есть интересующая программа.

Существует UniGetUI — графический интерфейс к большинству менеджеров программ, но не всегда ставится (из-за санкций), не надёжен.

Все консольные программы надо устанавливать через *Scoop*, при этом программы будут установлены в домашнюю папку, и пути будут добавлены в переменную *PATH*, что автоматически позволит запускать эти программы из командной строки.

Графические программы следует устанавливать либо через *WinGet*, либо через *Chocolatey* — примерно одно и тоже. Через *winget* стоит устанавливать те программы, которым не требуется регулярное обновление, либо которые умеют сами обновляться, например Telegram — установил и забыл. Если программы требуют регулярного контроля над обновлениями, то это удобнее делать через *ChocolateyGUI.*

> В общем, все консольные программы устанавливать через *Scoop*, все не консольные программы ставить через *WinGet*, а через *Chocolatey* устанавливать то, чего нет в *WinGet*.

Note: Видимость установленных программ менеджерами программ. Scoop "видит" только те программы, что установил сам. *Winget* помимо программ, которые были им установлены, также "видит" программы, установленные через *Chocolatey*; для *Chocolatey* обратное неверно — если программа установлена не через *Chocolatey*, то она её не "видит".

***Note:* Большинство программ в этом руководстве являются *программами с открытым исходным кодом*. Обо всех *проприетарных* программах есть явная пометка об этом.**

***Note:* В данном руководстве везде предполагается использование *cmd.exe* (*Clink*) в качестве терминала для выполнения команд, если только явно не указано иное.**

***Note:* Списки установки программ в последующих трёх разделах устанавливают практически все программы, рассматриваемые в данном руководстве, кроме тех излишне больших программ, решение об установке которых должен принимать пользователь из-за размера занимаемого места. (<*TBD*>: синхронизировать списки и пометить по тексту что уже установлено)**

3.1 Scoop

https://scoop.sh/#/apps — сайт-каталог доступных программ.

Установка scoop с сайта https://scoop.sh/ — запустить две строчки в *PowerShell* (не Admin!):

```
Set-ExecutionPolicy -ExecutionPolicy RemoteSigned -Scope CurrentUser
Invoke-RestMethod -Uri https://get.scoop.sh | Invoke-Expression
```

После выполнить проверку:

```
scoop checkup
```

Здесь потребуется пара настроек Windows для *Developer Mode* и *LongPaths*, если они не были сделаны при установке Windows.

Установить все программы — в любом терминале (не Admin) выполнить:

```
scoop install git
scoop update

scoop config aria2-enabled false

scoop bucket add nerd-fonts
scoop install nerd-fonts/IosevkaTerm-NF-Mono
scoop install busybox
scoop install fd
scoop install ripgrep
scoop install fzf
scoop install ccat
scoop install btop
scoop install wttop
scoop install 7zip
scoop install curl
scoop install gcc
scoop install gdb
scoop install make
scoop install cmake
scoop install time
scoop install hyperfine
scoop install clipboard
scoop install imagemagick
scoop install ffmpeg
scoop install handbrake-cli
scoop install openssh
#scoop install cwrsync
scoop install concfg
scoop install neovim

scoop install python

scoop install winget

scoop bucket add extras
scoop install gnuplot
```

3.2 Winget

Список доступных к установке программ:
https://winstall.app/

winget уже должен быть установлен с помощью *scoop*:

scoop install winget.

Устанавливать программы из под пользователя, а не Admin!, при необходимости *winget* сам спросит права.

```
winget install --id=FarManager.FarManager -e
winget install --id=Hibbiki.Chromium -e
winget install --id=Mozilla.Firefox.ESR -e
winget install --id=Chocolatey.Chocolatey -e
winget install --id=OpenVPNTechnologies.OpenVPN -e
winget install --id=AutoHotkey.AutoHotkey -e
winget install --id=Microsoft.PowerShell -e
winget install --id=Microsoft.WindowsTerminal -e
winget install --id=nomacs.nomacs -e
winget install --id=xiaoyifang.GoldenDict-ng -e
winget install --id=WinDirStat.WinDirStat -e
winget install --id=PuTTY.PuTTY -e
winget install --id=Telegram.TelegramDesktop -e
winget install --id=Meld.Meld -e
winget install --id=WinMerge.WinMerge -e
winget install --id=Qalculate.Qalculate -e
winget install --id=Rufus.Rufus -e
winget install --id=hluk.CopyQ -e
winget install --id=FreeCAD.FreeCAD -e
winget install --id=KDE.LabPlot -e
```

3.3 Chocolatey

https://community.chocolatey.org/packages — каталог программ в *Chocolatey*.

Chocolatey уже был установлен с помощью *winget*:
winget install --id=Chocolatey.Chocolatey -e

Для установки *GUI* для *Chocolatey* запустить в Admin терминале:

```
choco install -y chocolateygui
```

Note: В настройках ChocolateyGUI (запускается уже из меню Start) лучше отключить вид в виде tiles/плиток — построчный показ удобнее.

В *Chocolatey* из терминала (так удобнее) — установить программы (Admin!):

```
choco install -y vcredist-all
choco install -y reflect-free
choco install -y vcxsrv
```

Note: В консольных терминалах для выполнения программ под правами *Admin* следует вставлять *sudo* перед командой, например: sudo choco install -y vcredist-all.

В *ChocolateyGUI*, в репозитории, эти программы также можно найти и установить вручную:

- Microsoft Visual C++ Runtime - all versions
- Macrium Reflect
- VcXsrv Windows X Server

3.4 Обновление (update) всех установленных в Windows программ

Все менеджеры программ предоставляют возможность обновить все программы одной командой — для этого выполнить в терминале (не Admin!):

```
scoop update --all
sudo choco update all -y
winget update --all
```

- *Scoop* может обновить только то, что он сам установил: scoop list. Обновить реестр программ и обновлений: scoop update, вывести список обновляемого: scoop status. Обновить всё: scoop update --all.
- *Chocolatey* может обновить только то, что он сам установил. В ChocolateyGUI при запуске он покажет, что можно обновить: нажать иконку-звёздочку. Обновить из CLI: sudo choco update all -y.
- *Winget* может обновить программы, которые были установлены и через winget, и через Chocolatey, список установленного: winget list, при наличии обновления эта команда укажет новую версию; либо, сразу показать список с доступными обновлениями: winget update. Обновить всё: winget update --all.

Note: Есть утилита, **topgrade**, которая апдейтит *абсолютно всё*: обновления Windows, драйвера, прошивки, scoop/choco/winget программы, всё вплоть до плагинов VSCode. Перед применением обязательно сделать бекап (например с помощью MacriumReflect), иначе, *topgrade* порой такое обновляет, что совсем не радует, например, до последнего обновления Win11. По хорошему, *topgrade* необходимо сначала поконфигурить, что обновлять, а что нет, и только потом запускать. Установка: scoop install topgrade, запуск: topgrade.

3.5 Некоторые программы и утилиты

Несколько параграфов о разных программах и утилитах, которые не требуют подробного описания установки или использования. Большинство из них были в списках установки выше, т.е. *уже* были установлены.

3.5.1 Программы для картинок

nomacs — программа для просмотра изображений; а также для их модификации, с широчайшими возможностями: конвертация, масштабирование, вращение, обрезка, работа с цветом, баланс белого, работа с raw форматами, интерактивная обработка нескольких изображений синхронно. Конечно, не фотошоп, но для обычной работы с картинками 99% необходимого функционала обеспечивает.

```
winget install --id=nomacs.nomacs -e
```

Gimp — opensource замена фотошопу; программа для тех, кому не достаточно возможностей *nomacs*.

```
winget install --id=GIMP.GIMP -e
```

3.5.2 pdf-viewer

Если бы не было просмотра *pdf* в каждом браузере, то можно было бы установить Sumatra PDF:

```
winget install --id=SumatraPDF.SumatraPDF -e
```

но сейчас это уже излишне.

3.5.3 Программки для работы с текстом

Офисные программы рассмотрены в отдельной главе *Офисные Пакеты*. Написание программных кодов предполагается в *VSCode*, настройка которого рассмотрена в отдельной *главе*.

- **Notepad++** — легковесная замена блокнота, ультраэдита, и проч., с подсветкой синтаксиса, широкими возможностями редактирования. Устанавливается: scoop install extras/notepadplusplus.
- **Meld** и **WinMerge** — это программы для сравнения и редактирования файлов бок о бок, наиболее популярные из открытых. Используются, в том числе, для разрешения конфликтов при слияниях *git*. Устанавливаются, соответственно: winget install --id=Meld.Meld -e и scoop install extras/winmerge.
- **GoldenDict-ng** — локальные словари для перевода. По умолчанию активируется (переводит) по двойному нажатию на <Ctrl+C>. Устанавливается так: winget install --id=xiaoyifang.GoldenDict-ng -e; но при установке никаких словарей не ставится, их необходимо дополнительно скачать и указать программе где они, скачанные, лежат. Свободные словари:
  - dict-freedict-eng-rus - English-Russian dictionary for the dict server/client

  - mueller7-dict - Mueller English/Russian dictionary in dict format
  - *<TBD>*

3.5.4 Утилиты для работы с дисками

- **WinDirStat** — утилита для контроля занимаемого места файлами на диске, показывает размер файлов в виде плиток. Установка: winget install --id=WinDirStat.WinDirStat -e
- **Rufus** — утилита для создания загрузочных флешек, для установки операционных систем Windows, Linux и пр. Установка: winget install --id=Rufus.Rufus -e.
- **GSmartControl** — утилита для получения информации *S.M.A.R.T. SSD/HDD*, в частности, для контроля истирания *SSD*. Установка: winget install --id=AlexanderShaduri.GSmartControl -e
- обновление прошивок **SSD**:
  - *Samsung*: sudo choco install -y samsung-magician
  - *Intel*: winget install --id=Solidigm.StorageTool -e
  - *Transcend*: winget install --id=TransendInfo.SSDScopeApp -e
  - *Kingston*: winget install --id=Kingston.SSDManager -e
  - *Micron*, 2: sudo choco install -y micron-storage-executive
  - *Crucial*: winget install --id=Crucial.StorageExecutive -e
  - *WD*: winget install --id=WesternDigital.Dashboard -e

3.5.5 Прочие системные утилиты

- для ноутбуков **Lenovo** есть утилита Lenovo System Update, которая может скачать/обновить BIOS и специфичные драйвера для Windows. Установка: winget install --id=Lenovo.SystemUpdate -e. Ещё вариант.
- для ноутбуков **Huawei** есть утилита PC Clone для копирования системы с/на другие компьютеры и ноутбуки. Копирование происходит: а) копированием всех файлов пользователя, б) установкой соответствующих программ на другом компьютере и копировании данных для каждой из этих программ (поддерживается ограниченный список программ). Должна работать не только с Huawei.
- **FanControl** — утилита для настройки скорости вращения вентиляторов, актуально для ноутбуков, подходит не для всех ноутбуков, особенно свежих моделей; установка: winget install --id=Rem0o.FanControl -e.

3.5.6 Драйвера

Если какое-то устройство не работает, или знаки вопроса в *Device Manager*, то можно попробовать поставить драйвера (без гарантии что заработает и ничего не сломается, и сначала сделать бэкап).

HW-Info — бесплатная *проприетарная* утилита, выводит список наличных аппаратных устройств:

```
winget install --id=REALiX.HWiNFO -e
```

SDIO https://www.snappy-driver-installer.org/[1] — Установщик драйверов:

```
winget install --id=samlab-ws.SnappyDriverInstaller -e
```

3.6 Backup операционной системы Windows

Резервное копирование систем Windows удобно делать используя программу **MacriumReflect** — это *проприетарная* программа, но бесплатная для персонального использования. Программа должна была быть установлена через *Chocolatey* (см. выше):

```
sudo choco install -y reflect-free
```

3.6.1 Особенности использования MacriumReflect

При первом бэкапе будет создан шаблон, который, в дальнейшем, можно будет запускать безо всех этих шагов с указанием пути, имени, и т.п. Шаблон во втором табе/вкладке.

Для возможности восстановления из бэкапов необходимо создать загрузочную запись, либо на флешке, либо на самом диске с установленной операционной системой. Это делается в меню *Other Tasks* > *Create Rescue Media*.

В меню *Other Tasks* > *Edit Defaults and Settings* > *Update Settings* отключить проверку обновлений — очень часто обновляется, будет раздражать. Рядом в *Backup Defaults* > *Retension Rules*, стоит отключить автоматическое удаление старых бэкапов.

[1] не ~~**SDI**~~!

3.6.2 duplicati

<TBD>

Open-source program, with *VSS* support!

https://duplicati.com/ https://docs.duplicati.com/

```
winget install --id=Duplicati.Duplicati  -e
```

3.7 Браузеры

Наиболее широко распространённые браузеры: Chromium (Chrome), Firefox, Edge, Brave. Chrome, Edge, Brave созданы на основе Chromium, Firefox — отдельная разработка. Все браузеры примерно равны, и у всех есть возможность установки наиболее востребованных плагинов. Edge и Chrome по умолчанию следят за пользователями, Chromium и Firefox не следят (так как их исходный код доступен и любой желающий параноик в этом может убедиться), но Edge и Chrome наиболее массовые браузеры, и, поэтому, наиболее протестированные и защищённые. На сегодняшний день для повседневной работы с не конфидициальными данными, в Windows вполне обосновано применять *Edge* (с установленным плагином *UBlock Origin*, и настройками, запрещающими отправку данных в *Edge* > *Settings* > *Privacy, Search and Services:*).

Особенности использования современных браузеров:

- *UBlock Origin. "Must have"* для всех браузеров, без вариантов! *Edge*, *Chrome/Chromium*, *Firefox*. После установки плагина в настройках дополнительно включить фильтр *Ru* в настройках плагина в табе *Filter lists* в самом низу в *Regions, languages*.
- *NoScript.* Во всех браузерах есть *NoScript* плагин, который разрешает запуск каких-либо *.js*-скриптов только по *белому списку*, т.е. никакие сторонние скрипты никогда *не запустятся*, если только их не *разрешить явно*. Одно из следствий работы этого плагина, состоит в том, что, если явно не разрешать работу js-скриптов, которые отвечают за локацию местоположения пользователя, то начинают работать те сайты, которые ограничивают доступ для пользователей из России.
- В *Firefox* есть встроенный менеджер паролей, защищённый *Мастер Паролем*, т.е. Firefox не позволяет посмотреть сохранённые пароли и реквизиты банковских карт пока не будет введён *Мастер Пароль*. У Chromium (Chrome, Edge) нет такой возможности — они позволяют любому посмотреть все сохранённые пароли. /Параноик Mode Off.

Примерный список плагинов приведён в конце параграфа на примере *Chromium*.

Далее рассмотрим некоторые варианты использования браузеров, учитывающие их особенности.

3.7.1 Конфиденциальные данные

Установить *FirefoxESR*: winget install --id=Mozilla.Firefox.ESR -e, установить *МастерПароль*, установить *NoScript*, *U-Block Origin*. Этот браузер будет для работы с конфиденциальными данными, для работы: с почтой, банковскими сайтами, и т.п. Поначалу придётся много раз добавлять сайты в разрешённые в *NoScript*, но их всё же ограниченное число.

3.7.2 Обычное использование интернета

Установить один из браузеров для работы по умолчанию (кроме уже задействованного FirefoxESR). Установить в нём все плагины, кроме *NoScript.*

3.7.3 Песочница

Brave позиционируется как наиболее защищённый браузер, и он может использоваться в качестве “песочницы”. winget install --id=Brave.Brave -e. В настройках в brave://settings/shields включить *Aggresive* в *Trackers & ads blocking*, и включить *Block Scripts*. В последующем использовании разрешать скрипты по очереди — иконка щита справа от адресной строки. В браузере ни в коем случае не сохранять логинов и паролей. Тогда в этом браузере можно абсолютно безопасно посещать любые недостоверенные сайты, и это не приведёт ни к компрометации данных, ни к вирусам. Также в Brave могут открываться некоторые сайты, которые в других браузерах не открываются из-за санкций.

3.7.4 Примерный набор плагинов

для браузера *Chromium*:

uBlock Origin
Open Tabs Next to Current
QR Code Generator
Enable right click
History Trends Unlimited
The Marvellous Suspender
Unpaywall

Для того, чтобы открывались подсанкционные сайты, например netlib.org — сайт математической библиотеки LAPACK, можно попробовать следующие плагины:
Runet Censorship Bypass
Snowflake

Есть плагин *uBlacklist*, который позволяет исключить из выдачи поисковиков (google.com и т.д.) некоторые сайты — заблокировать их от показа.

3.8 Far Manager и консольный терминал

3.8.1 Нулевой уровень сложности настройки

Если **Far** ещё не установлен, то установить:

```
scoop install 7zip
winget install --id=FarManager.FarManager -e
```

Note: Поменять *шрифты* можно кликнув правой кнопкой мыши на иконке Far, в свойствах.

Note: В Far'е можно открывать ssh-соединения с поддержкой X11 приложений, для этого в окне терминала Far (Ctrl+O) вызвать две команды:

```
set DISPLAY=127.0.0.1:0.0
ssh -X somename@somehost
```

3.8.2 Второй уровень сложности

Запускать **Far** в **WindowsTerminal**, в котором можно настроить цвета/палитры, *шрифты* и т.п.

Установить *Far* и *WindowsTerminal*:

```
scoop install 7zip
winget install --id=FarManager.FarManager -e
winget install --id=Microsoft.WindowsTerminal -e
```

3.8.2.1 Настройка Windows Terminal

- Добавить профиль для запуска Far: открыть WindowsTerminal *Settings*, в меню слева выбрать (внизу) + *Add a new profile*, нажать + *New empty profle*, задать имя профиля *Far*, в пункте *Command line* задать выполняемую команду, выбрав файл *"C:\Program Flies\Far Manager\Far.exe"*, в пункте *Starting directory* отжать галочку (станет *%USERPROFILE%*), для отображения иконки выбрать тот же самый *"C:\Program Flies\Far Manager\Far.exe"* — иконка будет браться непосредственно из *.exe*; сохранить. Выбрать профиль *Far* для старта: в меню *Settings* самый первый пункт. (Всё то же самое можно сделать через редактирование файла конфигурации settings.json, подробнее чуть ниже в Пункте про схемы)
- Настроить запуск WindowsTerminal по умолчанию для всех консольных команд и приложений: в настройках Windows в *Settings* > *Update & Security* > *For developers: Terminal* > *Windows Terminal*.

- Тема оформления, *Шрифт*, Форма курсора, меняются в настройках WindowsTerminal в *Settings* > *Defaults* > *Appearance* — это будут настройки по умолчанию для всех новых типов терминалов. Эти настройки можно будет изменить для каждого профиля терминала отдельно.
- Частоту мигания курсора можно изменить только глобально в настройках Windows: *Change cursor blinking rate*.
- В Windows Terminal для создания ssh-соединения (с поддержкой X11 приложений) вызвать следующую команду прямо из *cmd* (*clink*) сессии:

  ```
  set DISPLAY=127.0.0.1:0.0& ssh -X somename@somehost
  ```

  Можно настроить отдельный профиль для фиксированного ssh-соединения, чтобы сразу открывалась новая ssh-сессия при открытии профиля. Для этого сдублировать профиль *cmd.exe*, с такой командой:

  ```
  cmd.exe /c set DISPLAY=127.0.0.1:0.0& ssh -X somename@somehost
  ```

  Переменную DISPLAY можно установить перманентно в пользовательских переменных среды окружения командой:

  ```
  setx DISPLAY 127.0.0.1:0.0
  ```

- Использовать *PuTTY* в Windows Terminal: для того, чтобы *PuTTY* открывал соединение и показывал всё в Windows Terminal, необходимо настроить новый профиль, в котором будет вызываться загрузка соответствующего профиля PuTTY командой "plink.exe -load \"putty-session\"". (Команда *C:\Program Files\PuTTY\plink.exe*, это программа, которая осуществляет непосредственное соединение для *PuTTY*, с поддержкой ключей и *Pageant*.) Но Windows Terminal очень плохо поддерживает *plink.exe*, поэтому многие клавиши (стрелки) работать не будут.
- Дополнительные цветовые схемы см. в https://github.com/mbadolato/iTerm2-Color-Schemes или https://windowsterminalthemes.dev/. Установка дополнительных схем осуществляется вставкой соответствующего json-фрагмента настроек цветовой схемы в секцию "schemes": в общей конфигурации WindowsTerminal settings.json, которая открывается для редактирования по нажатию на “Шестерёнку” в левом нижнем углу в меню *Settings*.
  Просмотреть цвета текущей схемы можно командой: colortool -c, которая устанавливается: scoop install colortool.
- Для того, чтобы *Far* показывал другие цветовые схемы, помимо синей, необходимо отжать галочку в меню *Far*: *F9* > *Options* > *Colors: Palette* > *Classic Palette: Off*.
  Из не синих схем интересна схема Glacier, например.
- Для автоматического копирования в буфер обмена выделенного мышкой включить опцию *Settings* > *Interaction: Automatically copy selection to clipboard*. <new-01>

- Не показывать Warning на *multiline paste*: добавить строчку "multiLinePasteWarning": false, в самый верх (на первый уровень) конфига settings.json.
- Комбинации Ctrl+C, Ctrl+V, Ctrl+W, Ctrl+T для доступности в консольных приложениях следует перенастроить (в настройках в *Settings* > *Actions*) на варианты с Shift, т.е.: Ctrl+Shift+C, Ctrl+Shift+V, Ctrl+Shift+W, Ctrl+Shift+T и т.п.
- Для показа иконок в *Taskbar*'е в Windows должна быть включена опция *Show badges on taskbar buttons* в *Settings* > *Personalization* > *Taskbar*.
- Одним из интересных вариантов интеграции WindowsTerminal в Windows будет настройка *Панели задач* (*Taskbar*) без группировки окон — чтобы каждый терминал был в отдельном окне Windows; при этом надо установить опции для открытия каждого терминала в новом окне в *Settings* > *Startup* > *New instance behavior: Create a new window*.
- Для полнотекстового поиска по файлам, docx, pdf и т.п. смотри далее раздел *RipGrep*.
- Привычную переменную *HOME* можно задать в переменных среды окружения для пользователя выполнив:

```
setx HOME "%UserProfile%"
```

- Алиасы. Для каждой оболочки свои методы создания:
  - <*TBD*> https://stackoverflow.com/questions/17404165/how-to-run-a-command-on-command-prompt-startup-in-windows https://superuser.com/questions/1134368/create-permanent-doskey-in-windows-cmd
  - Для *PowerShell*: Guide как задать постоянные алиасы в профиле PS.
- У WindowsTerminal есть возможность настроить глобальную горячую клавишу на вызов WindowsTerminal откуда угодно. Для этого в settings.json прописать команду "globalSummon" в разделе "actions"::

```
"actions": [
 {
  "command": {
   "action": "globalSummon"
  },
  "keys": "alt+t"
 }
]
```

- Для WindowsTerminal доступны некоторые параметры запуска из командной строки, например, можно настроить открывать отдельный WindowsTerminal (новое окно) на запуск какого-то конкретного приложения, со специально настроенным профилем, например, создать алиас doskey vi=wt -d %CD% -p "wt-nvim" -- nvim $* и профиль с именем *"wt-nvim"* с командой %SystemRoot%\System32\cmd.exe /c nvim, тогда по команде vi test-01.cpp откроется новое окно с WindowsTerminal с открытым файлом *test-01.cpp* в редакторе *nvim.*

- Список всех действий WindowsTerminal, любое из них можно настроить на желаемую комбинацию клавиш.

3.8.2.2 Использование WindowsTerminal

См. также главу *Полезные консольные утилитки*.

<TBD>

Возможности WindowsTerminal.

- Ctrl-Shift-P — открыть *Command Palette* — панель действий
- Ctrl+Shift+T — открыть новую вкладку с терминалом по умолчанию
- Ctrl+Shift+N — открыть новое окно с терминалом по умолчанию
- Ctrl+Shift+Space — открыть селектор открытия новой вкладки
- Ctrl+Alt+1 — переключиться на первую вкладку
- Ctrl+Alt+2 — переключиться на вторую вкладку
- Ctrl+Shift+1 — открыть новую вкладку с профилем 0
- Ctrl+Shift+2 — открыть новую вкладку с профилем 1

WindowsTerminal умеет открывать терминалы не только в табах, но и панелях, т.е. в одном окне несколько терминалов, расположенных плиткой.

Список всех действий WindowsTerminal, любое из них можно настроить на желаемую комбинацию клавиш.

Links: WT by MS, WT actions, doskey, Guide.

3.8.3 Финальный уровень сложности

В принципе, на этом можно и остановиться, но для полноценной разработки/программирования рекомендуется установить и настроить нормальную командную оболочку (shell) **Clink**.

В *Приложении* показана настройка Clink.

После **установки и настройки Clink** останется только настроить для него профиль в WindowsTerminal, который проще всего получить сдублировав профиль для *cmd.exe*, после чего, в новом профиле, в качестве запускаемой команды прописать:

```
%SystemRoot%\System32\cmd.exe /k "clink inject"
```

Так же в Приложении есть краткая справка по использованию оболочки **Clink**.
См. также главу *Полезные консольные утилитки*.

3.8.4 Немного о шрифтах

При постоянном использовании консольных приложений одним из ключевых факторов эргономики являются шрифты. Не стоит пренебрегать шрифтами, времени много не займёт, а удобство повысит.

Для консольных приложений используются моноширинные шрифты. Посмотреть наиболее популярные бесплатные шрифты можно на сайте https://www.nerdfonts.com/font-downloads.

Для улучшения читабельности длинных строк можно использовать *сжатые* (*condensed*, *compressed*, *narrow*) шрифты, это шрифты, которые более высокие, чем широкие. Примеры сжатых шрифтов: Iosevka(view), D2CodingLigature(view).

В последние годы в консольных программах стали широко использоваться шрифты с иконками, смайликами, и прочими стрелочками. Проект https://www.nerdfonts.com/ добавляет набор иконок к любым шрифтам. При установке шрифтов, при прочих равных, следует выбирать версию *Nerd Font*, если есть в наличии.

Все, или почти все, шрифты можно установить через *Scoop*. Поиск шрифта, установка шрифта:

```
scoop bucket add nerd-fonts
scoop install nerd-fonts/Iosevka-NF-Mono
```

3.9 Многокарманный буфер обмена

CopyQ — многокарманный буфер обмена. *Note:* хоть в Windows и есть встроенный многокарманный буфер, но он очищается при перезагрузке, поэтому его невозможно использовать как некую временную “записную книжку” с сохранёнными фрагментами текста.

CopyQ поддерживает несколько списков хранения, позволяет исправлять (редактировать) хранимые фрагменты текста, поддерживает темы оформления, кроссплатформенный.

Установка:

```
winget install --id=hluk.CopyQ -e
```

~~Для автозапуска *CopyQ* при загрузке Windows, необходимо скопировать ссылку на CopyQ в папку автозапуска, для этого в меню *Start* правой кнопкой по нему выбрать *Open file location*, скопировать ярлык, и *Paste* его в папку автозапуска, открываемую по команде shell:startup в строке запуска <Win+R>.~~

Настройка:

- <Win+V> занят встроенным в Windows буфером обмена, поэтому, на вызов CopyQ можно назначить сочетание клавиш <Alt+V> в *Preferences* > *Shortcuts* > *Globals: Show/Hide main window*;
- в настройках *Preferences* > *History* увеличить количество хранимых заметок до 10000;

- в настройках *Preferences* > *Items* > *Images* увеличить максимальный хранимый размер изображений до 2560 на 1600;
- при копировании в буфер форматированного текста (rich text) из Word или html, форматирование сохраняется. Эту особенность можно отключить в *Preferences* > *Items* > *Text*. Или настроить *Paste as plain text* в *Shortcuts*.

Еще есть многокарманный буфер *Ditto*, если *CopyQ* чем-то не устроит, установка.

3.10 Системы виртуализации WSL и Docker

WSL требуется для программирования: для компиляции и отладки программ. **Docker** необходим для установки некоторых сложных приложений (*OpenFOAM*).

3.10.1 WSL

WSL — Windows Subsystem for Linux, это встроенная в Windows система виртуализации, позволяющая запускать Linux приложения непосредственно в Windows.

Сначала установить WSL2, для этого в терминале (Admin) запустить:

```
wsl --install
```

В процессе установки Ubuntu установить набор пакетов для программирования:

```
sudo apt update
sudo apt install aptitude mc bash-completion neovim wget curl \
      ca-certificates git build-essential gfortran libopenmpi-dev \
      mpi-default-bin libopenblas-dev libopenblas-openmp-dev \
      libhypre-dev liblapack-dev liblapack-doc libhdf5-mpi-dev \
      libscotch-dev petsc-dev libsuitesparse-dev libsuperlu-dev \
      libsuperlu-dist-dev fortran-language-server gdb \
      python3 python3-pip ripgrep fzf fd-find parallel libsixel-bin
```

Методы запуска консоли WSL:

- самый лучший вариант, это запуск в отдельном профиле в *WindowsTerminal*[2] (или в отдельном профиле в *Cmder*[3]);
- запуск команды wsl прямо из запущенного термина Cmd/Clink/PowerShell/Far, который запущен в WT или Cmder;
- запускать WSL или Ubuntu.exe из меню *Start/Пуск*;

[2] сам автоматически настроится при установке WSL
[3] настраивается автоматически в *Cmder* по *Setup Tasks* > *Add/refresh default tasks*

Note: Ubuntu.exe — это alias — Windows автоматически создаёт такие ссылки для каждой WSL системы для быстрого доступа/запуска.

Note: установленные в WSL2 графические Linux приложения работают “из коробки”.

3.10.1.1 WSL Manager

Существуют графические утилиты для управления WSL контейнерами, одна из них **WSLManager**; установка:

```
winget install --id=Bostrot.WSLManager -e
```

***Отступление* про использование *WSL*:**

Аргументы *some commands* после слова *wsl* в команде wsl some commands будут запущены внутри WSL виртуалки: например, команда wsl rg "some-phrase" запустит Linux утилиту rg, которая, в данном случае, будет выполнять полнотекстовый поиск *“some-phrase”* в текущей папке. Т.е. в любом терминале Windows можно выполнить команду Linux, в том числе для действий над своими файлами/данными лежащими на локальных дисках *C:*, *D:* и т.п.

Аналогично, скомпилировать программу из исходного кода, с использованием библиотеки *OpenMPI*:
wsl mpif90 -g -o test test.f
Запустить скомпилированную параллельную программу в параллельном режиме:
wsl mpirun -n 4 ./test
Запустить отладку скомпилированной программы:
wsl gdb test

В итоге, в Windows более не требуется сложная процедура установки компиляторов, библиотек, прописывания путей, и т.п. — это всё делается в Linux, который лучше для этого приспособлен.

3.10.2 Docker

Docker — это менеджер виртуальных машин; сам он не выполняет виртуальные машины, а только занимается организацией создания, хранения, и запуска машин с помощью других систем виртуализации, например, таких как WSL. Поэтому, устанавливать Docker следует только после установки WSL.

Установка:

```
winget install --id=Docker.DockerDesktop -e
```

Про установку и запуск виртуальных машин в Docker см. далее в *Дополнительных материалах*.

3.11 ssh/PuTTY

В Windows для удалённого доступа по протоколу *ssh* могут использоваться **Putty** и **OpenSSH**. Оба пакета выполняют один функционал. *OpenSSH* запускается в окне консольного теминала WindowsTerminal/Cmder/Conemu, поэтому в нём доступны все клавиши навигации, настроенные *шрифты* и цветовая схема. У *Putty* свой графический терминал с небольшим числом возможностей и настроек. В Windows выбор между *Putty* и *OpenSSH* определяется привычкой и удобством использования, по качеству ssh-соединения они одинаковы.

Документация по настройке ssh/Putty на Windows: SSH-on-Windows, Integrate-SSH-Agent-or-PuTTY-Agent.

Note: В Windows имеется встроенный *OpenSSH*, но из-за ANSI несовместимости его нельзя применять в терминалах Cmder/Conemu, поэтому потребуется ещё установить нормальный *OpenSSH*.

Установить оба, и *OpenSSH*, и *Putty*:

```
scoop install openssh
winget install --id=PuTTY.PuTTY -e
```

Для настройки входа без пароля потребуется добавить свой публичный (*.pub*) ключик к файлу ~/.ssh/authorized_keys на удалённом узле (хосте); далее рассмотрены оба варианта, и для *OpenSSH* и для *Putty*.

3.11.1 Вариант для OpenSSH соединения

Инструкция:

1. Посмотреть содержимое папки в домашней директории на Windows %UserProfile%\.ssh на наличие уже созданных ключей, если там есть пара файлов id_SOMEPROTOCOL и id_SOMEPROTOCOL.pub, то ключи уже созданы и перейти к следующему шагу, иначе создать их командой:

   ```
   ssh-keygen
   ```

   понажимать *Enter*, и ключи готовы.

2. Скопировать ключ на удалённый хост:

   ```
   ssh-copy-id user@hostname
   ```

3.11.2 Вариант для Putty соединения

Инструкция тут посложнее:

1. Открыть из меню Start *PuTTYgen*, сгенерировать ключ, сохранить обе части — public/private.
2. Выделить открытую часть и скопировать в буфер обмена, чтобы добавить в файл .ssh/authorized_keys на Linux, для чего зайти на Linux через Putty, и выполнить:

   ```
   cd ~/.ssh
   vi authorized_keys
   ```

   Далее процесс добавления ключа из буфера обмена в открытый в vi файл: набрать на клавиатуре команду:

   ```
   Go<Shift+Insert><Esc>:wq
   ```

   здесь <Shift+Insert> и <Esc> — это клавиатурные комбинации, а не строчки которые вводить.
   (*Note:* паблик ключ должен иметь форму одной строки, если это не так, то лишние переводы строки необходимо стереть: в vi это делается командой Shift+J, которая объединяет текущую и следующую строки; этой командой можно воспользоваться после <Esc>, перед :wq.)

   Поправить права доступа (на всякий случай):

   ```
   chmod 600 authorized_keys
   ```

   Можно выходить из сессии: exit == Ctrl+D.
3. Теперь добавить ключ в Putty для использования: запустить из Start *Pageant*. Правой кнопкой мышки в трее на иконке *Pageant* кликнуть и указать *Add key*, выбрать сохранённый приватный ключ *.ppk*. Чтобы вход по ключам всегда срабатывал, потребуется настроить автозапуск *Pageant*: скопировать ссылку на *Pageant* в папку автозапуска, для этого в меню Start правой кнопкой по нему выбрать *Open file location*, скопировать ярлык, и *Paste* его в папку автозапуска, открываемую по shell:startup в <Win+R>.

3.11.3 Конвертация ключей *Putty* в формат *ssh*

Формат хранения ключей у Putty отличается от формата *ssh*. Для того, чтобы сохранить в нормальном формате, у Putty, в утилите генерации ключей *C:\Program Files\PuTTY\puttygen.exe*, есть менюшка *Conversions* > *Export OpenSSH key (force new file format)*, которая сохраняет приватный ключ в формате *ssh* (сохранять в файл без расширения, в id_rsa). Формат *.pub* ключа идентичен нормальному, его конвертировать не потребуется. Эти два файла: id_rsa и id_rsa.pub, скопировать в папку %UserProfile%\.ssh, после чего ssh начнёт их видеть.

3.12 X-server

Установленный *WSL* несёт в себе “нативный” *X-server*, он работает, его можно использовать, но его производительность заметно ниже, чем у **VcXsrv**. Поэтому, установить VcXsrv:

```
sudo choco install -y vcxsrv
```

Note: VcXsrv запускается командой XLaunch из меню *Пуск*.

Note: При возникновении ошибок *X-Authority* при соединении, можно попробовать отключить контроль безопасности: для этого при запуске *XLaunch (VcXsrv)* поставить галочку “Disable Access Control”, это приведёт к разрешению на запуск графических приложений с *любого* удалённого узла.

3.12.1 OpenSSH

Для создания ssh-соединения в *OpenSSH* выполнить в *cmd*/*Clink*/*Far*:

```
set DISPLAY=127.0.0.1:0.0
ssh -X somename@somehost
```

тогда проброс (*Forwarding*) протокола *X11* заработает (Guide).

Ненужные пояснения:

- Здесь *127.0.0.1:0.0* — это IP-адрес *localhost*'а, на котором наш запущенный локальный X-сервер ожидает данных, и указание на номер дисплея (*0.0*) X-сервера. Здесь применяется явный IP-адрес, а не имя *localhost*, потому что последний может быть и IPv6.
- Опция -X (или -Y) указывает утилите ssh на создание X-Forwarding. -X — это вариант опции со включёнными ограничениями безопасности; -Y означает, что X11 приложение будет иметь полный доступ к отображаемому на локальном мониторе, этого варианта следует избегать, или применять только для доверенных удалённых хостов и доверенных X11 приложений.

3.12.2 PuTTY

При создании ssh-соединения через Putty, для возможности запуска графических приложений по протоколу *X11*, в настройках Putty необходимо устанавливать галочку *X-Forwarding* (эквивалентно опции -X/-Y у ssh).

3.12.3 Тест

После настройки *X* сервера проверить его работоспособность можно парочкой утилит: xclock — простые часы с циферблатом, и glxgears — OpenGL программка из пакета *mesa-utils*.

> *Отступление:* Когда графическое приложение отрисовывается по X11 протоколу, оно отправляет *"команды отрисовки"* на локальный X-сервер, т.е отправляет не картинку, а инструкции как эту картинку рисовать. В общем случае, эти X11 инструкции имеют полный доступ к экрану и могут считывать всё его содержимое. Поэтому, для безопасности, в X-серверах, по умолчанию, включён режим *"X11 SECURITY extension"*, когда может применяться только ограниченный набор X11 команд, которые не имеют доступа к экрану. Но, некоторые X11 приложения, для полноценной работы, не могут ограничиться таким урезанным набором X11 команд, и, для них была введена опция -Y, отключающая ограничения *"X11 SECURITY extension"*. Эту опцию следует применять только в случае, если *необходимое* приложение, которому вы доверяете, без неё не работает.

3.13 Удалённый рабочий стол

Помимо привычного *RDP* — Remote Desktop Protocol, существуют несколько инструментов для удалённого подсоединения, с более удобным и широким функционалом, которые могут проходить через интернет.

<TBD>

Один из них, это *кроссплатформенный* **RustDesk**, позволяющий подсоединяться к рабочему столу компьютера в локальной сети. Также, через выделенный сервер, возможен коннект к *любому* компьютеру через интернет, без необходимости VPN. Установка:

```
winget install --id=RustDesk.RustDesk -e
```

3.14 Протокол Sixel

Sixel — это текстовый протокол, позволяющий передавать и отрисовывать изображения прямо в терминале без использования X-Server'а. Данные передаются в виде ASCII символов либо непосредственно в терминал при локальном запуске программы, например в WSL, либо по сети через ssh соединение. Графическая информация преобразуется в символы, а терминал производит обратное преобразование в пиксели и отображение.

Windows Terminal начиная с версии 1.22 поддерживает протокол Sixel, об этом рассказывается на странице документации; там же приведены примеры применения протокола Sixel в терминале с WSL:

- отображение файлов картинок непосредственно в терминале:

```
img2sixel -w 800 some-image.png; echo ""
```

- вывод графической информации от работы программы в терминале, в данном случае работы python-скрипта:

```
python3 custom_shaded_3d_surface.py | img2sixel; echo ""
```

3.15 OpenVPN

OpenVPN.

```
winget install --id=OpenVPNTechnologies.OpenVPN -e
```

Настройка: Проще всего скопировать ключи прямо вместе с их папкой, в "*C:\Program Files\OpenVPN\config*". OpenVPN увидит их (после перезапуска?) и будет показывать в виде отдельной папки конфигурации (появится возможность выбирать из нескольких ключей).

> *Warning:* Ни в коем случае не пытаться загрузить ключи через меню интерфейса *Import* > *Import File...*, потому что в этом случае будет скопирован только один файл из трёх!

Legacy OpenVPN

OpenVPN версии 2.6.xy не работает с более старыми ключами, т.к. в них изменились некоторые параметры (при использовании ключей в устаревшем формате появляется запрос на ввод пароля).

Поэтому, либо скачать предыдущую версию 2.5.xy (2.5.10) с сайта https://openvpn.net/community-downloads/ и, после установки, зафиксировать эту версию командой winget pin add OpenVPNTechnologies.OpenVPN, либо ~~отредактировать *.ovpn* файл~~ скачать ключики заново — они будут уже в новом формате.

Этап 2: Установка и настройка иногда нужных программ

4 Настройка среды для программирования

В руководстве рассматривается вариант настройки универсальной среды разработки **VSCode** в Windows, который будет использовать инструменты Linux для компиляции, запуска и отладки программ. Для этого в VSCode используются плагины WSL и Remote Development, которые позволяют запускать компиляторы и отладчики, как локально (*WSL*), так и удалённо по *ssh*. *Note:* Также существует вариант запуска процесса разработки внутри виртуальных машин в Docker.

Сначала установить *WSL2* и *Docker* — см. выше.

4.1 VSCode

Docs, Getting started, Keybindings, Keybindings.pdf.

Если VSCode ещё не установлен, то установить:

```
winget install --id=Microsoft.VisualStudioCode -e
```

Для запуска компилятов и т.п. в Windows используется префикс wsl:

```
wsl gfortran -g -o test test.f
wsl gdb test
```

VSCode умеет так запускать компиляторы и отладчики при помощи расширения WSL.

Для начала работы, после установки расширения *WSL*, в Cmd/PS перейти в папку с проектом или исходным кодом программы, и запустить команду:

```
wsl code .
```

code — это название исполняемого файла VSCode, она так называется. В результате выполнения команды запустится VSCode с открытой текущей папкой (.).

Note: при таком опосредованном выполнении в WSL, VSCode расширения (extensions) будут устанавливаться в раздел плагинов *WSL* — в нижний список установленных расширений. Т.е. в VSCode будет два набора расширений, и, возможно, какие-то расширения будут установлены и там и там, например Python, который можно использовать как в WSL, так и локально с помощью нативного *python*.

Расширение WSL запускает компилятор и пр. локально, непосредственно на компьютере. Существует вариант удалённой работы через *ssh* по сети — на кластер или внутрь виртуальных машин — делается это с помощью расширения Remote Development. Это расширение позволяет проводить удалённую работу/отладку с любого компьютера, в том числе через интернет,

причём все настройки сохраняются на стороне удалённого сервера/виртуалки (в папке проекта *.vscode*), и проекты будут настроены одинаково с любого удалённого места доступа.

Note: Профессиональные разработчики в своей работе применяют Docker, который позволяет для каждого запуска отладки создавать новый, эталонно настроенный контейнер (виртуалку), и уже с ним работать. Эта технология позволяет:

- иметь разные версии компиляторов, библиотек и настроек для отладки разных приложений;
- проводить разработку и отладку на гарантированно одинаковом наборе исходных данных, без необходимости после каждого запуска вручную очищать и копировать данные;
- также, эта технология позволяет настроить *Continuous Integration (CI)* — сервис Непрерывной Интеграции, запускающий набор тестов для каждого изменения (commit) внесённого в исходный код.

4.2 C/C++

Расширение C/C++ для VSCode должно работать “из коробки”.

Для удобства необходимо создать git репозиторий в папке проекта, тогда VSCode автоматически найдёт файлы, которые следует скомпилировать и отлаживать.

Отладка в Windows

```
cd project
wsl code .
```

<TBD>

Отладка в Windows не должна отличаться от отладки в Linux, см. следующий пункт.

Отладка в Linux

https://code.visualstudio.com/docs/editor/debugging

```
cd project
code .
```

Простой вариант

Для запуска отладки надо ткнуть *Breakpoint* в коде (кликнуть левее номеров строк), и запустить из меню *Run* > *Start debugging* (F5). VSCode скомпилирует файлы и запустит отладку.

Вариант посложнее

Для отладки в каких-то более сложных случаях, например, когда требуется передать аргументы командной строки, потребуется создать:

а) Build configuration, *tasks.json*.

Создать конфигурацию: *Ctrl+Shift+P*: *Tasks: Configure Default Build Task* > *gcc debug*, откроется *tasks.json*, его просто сохранить. Скомпилировать: *Ctrl+Shift+P*: *Tasks: Run Build Task* (*Ctrl+Shift+B*)

Конфигурация по умолчанию:

```
{
  "tasks": [
    {
      "type": "cppbuild",
      "label": "C/C++: gcc build active file",
      "command": "/usr/bin/gcc",
      "args": [
        "-fdiagnostics-color=always",
        "-g",
        "${file}",
        "-o",
        "${fileDirname}/${fileBasenameNoExtension}"
      ],
      "options": {
        "cwd": "${fileDirname}"
      },
      "problemMatcher": [
        "$gcc"
      ],
      "group": {
        "kind": "build",
        "isDefault": true
      },
      "detail": "Task generated by Debugger."
    }
  ],
  "version": "2.0.0"
}
```

б) сгенерировать и отредактировать файл настроек запуска отладки проекта ***launch.json***: в панели слева открыть панель отладки (жучок с треугольником), нажать ссылку *create a launch.json file*, из выпадающего меню выбрать *Launch C++*, и отредактировать только что созданный *launch.json*. Изменить поля для имени и аргументов, например:

```
"program": "${workspaceFolder}/test",
"args": ["mon-coordinates.csv", "1", "13"],
```

```
{
  // https://code.visualstudio.com/docs/editor/debugging#_launch-configurations
  "version": "0.2.0",
  "configurations": [
    {
      "name": "(gdb) Launch",
      "type": "cppdbg",
      "request": "launch",
      "program": "${workspaceFolder}/test",
      "args": ["mon-coordinates.csv", "1", "13"],
      "stopAtEntry": false,
      "cwd": "${workspaceFolder}",
      "environment": [],
      "externalConsole": false,
      "MIMode": "gdb",
      "setupCommands": [
        {
          "description": "Enable pretty-printing for gdb",
          "text": "-enable-pretty-printing",
          "ignoreFailures": true
        },
        {
          "description": "Set Disassembly Flavor to Intel",
          "text": "-gdb-set disassembly-flavor intel",
          "ignoreFailures": true
        }
      ]
    }
  ]
}
```

Notes

Для просмотра векторов в окошке *Watch* нажать `+` и набрать `argv, 4`, где *4* это длина просматриваемого участка вектора `argv`.

4.3 Python

На Windows можно установить нативный компилятор Python и его библиотеки, без использования WSL.

У Python есть несколько менеджеров окружений, которые используются для разделения установленных пакетов на независимые “песочницы”, с тем, чтобы было возможно установить пакеты, которые имеют зависимости, конфликтующие с зависимостями других пакетов. Помимо встроенного в Python менеджера ***pip*** существует простой и удобный ***pipx***. На практике, при установке широкоизвестных библиотек, лучше использовать стандартный метод pip install --upgrade some-python-library. Но если устанавливается Python-программа, особенно с большим числом зависимостей, то её лучше установить в отдельное окружение: pipx install some-python-app.

Установка *python* (*pip* идёт в комплекте):

```
scoop install python
```

После установки в консоли было выведено сообщение о возможности регистрации *python* для других приложений, и для этого предлагается выполнить reg-файл %UserProfile%\scoop\apps\python\current\install-pep-514.reg, что стоит сделать.

Note: после установки, для того, чтобы *.py* файлы стали запускаемыми (как *.exe*), необходимо ассоциировать их с %UserProfile%\scoop\apps\python\current\python.exe, либо просто запустив и выбрав, либо через меню *Choose a default app for each type of file* в *Пуск*.

Note: после установки Python, чтобы python-программы с расширением *.py* можно было запускать без добавки расширения (как это не обязательно *.exe*), необходимо, чтобы расширение *.py* присутствовало в переменной окружения *PATHEXT*.

Установка *pipx*:

```
python -m pip install --upgrade pipx
pipx ensurepath
```

Note: после установки какой-либо python-программы с помощью *pip* или *pipx*, она будет в путях. Но, вполне возможно, что это будет python-скрипт без какого-либо расширения, ни *.exe*, ни *.py* (как на Linux). В таком случае необходимо сделать *.exe* ссылку на этот скрипт. Например, после установки python -m pip install --upgrade eg, создать ссылку: scoop shim add eg python.exe --m eg, здесь на ссылку eg.exe создаётся ассоциация python.exe -m eg, которая будет работать в Windows.

Списки установленных пакетов библиотек и python-программ: pip list, pipx list.

Установить математические и HPC библиотеки:

```
python -m pip install --upgrade numpy
python -m pip install --upgrade plotly
python -m pip install --upgrade numba
python -m pip install --upgrade mpi4py
python -m pip install --upgrade jupyterlab
winget install --id=Microsoft.msmpisdk -e
winget install --id=Microsoft.msmpi -e
```

Note: Здесь в списке присутствует библиотека **numba**, это *JIT*[4] компилятор Python, применяемый для высокопроизводительных вычислений (HPC). С его помощью можно добиться скорости выполнения Python программ наравне с C/C++ или Fortran. Применение этой библиотеки совершенно тривиально: импортировать `from numba import jit` и добавить `@jit` перед определением своих функций `def ...`.

Для разработки на Python есть несколько вариантов IDE: VSCode, Spyder, Positron, проприетарный PyCharm.

Python в VSCode

VSCode рекомендуется как универсальная среда разработки, где освоив работу на каком-либо языке программирования, разработка на других языках уже не будет представлять сложности. Python в VSCode использовать не сложно.

```
scoop install extras/vscode
```

Расширение Python для VSCode позволяет выполнять участки кода по Shift+Enter. Также можно выполнять построчную отладку по *Debug*, *Run*, *NextLine*, *EnterInto*, и т.д.

Python в Spyder

В **Spyder** всё настроено для Python "из коробки". Основной упор сделан на *Data Science*: обработка данных на Python, построение графиков, статистика. По стилю работы похож на MatLab. При первом запуске проводится краткий обучающий тур основам использования.

```
winget install --id=Spyder.Spyder -e
```

Note: При наличии HiDPI экрана в настройках для этого есть опция *Tools > Preferences > Application: Interface > Enable auto high DPI scaling*.

[4] Just-In-Time

Python в Positron

Так же для *Data Science* стоит попробовать **Positron**. Это VSCode, где встроили набор плагинов для обработки данных на Python/R и проч. По стилю работы очень похож на VSCode (что подкупает ненужностью изучения чего-то непохожего), но это пока проект в beta стадии, могут встречаться ошибки.

```
python -m pip install --upgrade ipykernel
scoop install extras/positron
```

4.3.1 Блокноты на Python

Для создания ***Интерактивных блокнотов*** (*журналов*) на Python следует предпочесть расширение **Jupytext** (вместо *Jupyter*), сохранённые файлы которого имеют нормальный ***.py*** формат со специально оформленными комментариями, а не какой-то непонятный *JSON*.

<*TBD*> Подробнее об интерактивных блокнотах, реактивном программировании.

4.4 Fortran

CHECKME: <TBD>

```
cd my-cool-fortran-project
wsl code .
```

Установить расширение Modern Fortran, переключиться (Switch) на *Pre-Release* версию расширения.

Если подсветка не заработала, то разрешить в настройках *fortls* (должен быть уже установлен *fortran-language-server* в *WSL*).

Для компиляции потребуется файл настроек задач task.json:

Создать файл *tasks.json* для настройки компиляции: открыть командную палитру <Ctrl+Shift+P>, ввести *Tasks: Configure Task*, выбрать *Create tasks.json file from template*, выбрать *Others*, отредактировать на:

```
{
 // https://code.visualstudio.com/docs/editor/tasks#_custom-tasks
 "version": "2.0.0",
 "tasks": [
  {
   "label": "build",
   "type": "shell",
```

```
    "command": "gfortran",
    "args": [
     "-o",
     "${workspaceFolder}/program",
     "${file}"
    ],
    "group": {
     "kind": "build",
     "isDefault": true
    },
    "problemMatcher": ["$gcc"],
    "detail": "Generated task by VSCode"
   }
  ]
}
```

Для отладки потребуется создать/отредактировать файл launch.json: открыть командную палитру <Ctrl-Shift-P>, *Debug*, *Edit launch.json*:

```
{
 // https://code.visualstudio.com/docs/editor/debugging#_launch-configurations
 "version": "0.2.0",
 "configurations": [
  {
   "name": "(gdb) Fortran",
   "type": "cppdbg",
   "request": "launch",
   "program": "${workspaceFolder}/a.out",
   "args": [], // Possible input args for a.out
   "stopAtEntry": false,
   "cwd": "${workspaceFolder}",
   "environment": [],
   "externalConsole": false,
   "MIMode": "gdb",
   "setupCommands": [
    {
     "description": "Enable pretty-printing for gdb",
     "text": "-enable-pretty-printing",
     "ignoreFailures": true,
    },
   ],
  },
 ],
}
```

4.5 Julia

Установить **Julia**:

```
winget install --id=Julialang.Juliaup -e
juliaup add lts
julia
```

```
] add Revise, OhMyREPL, About, Unitful, Roots,
    DataStructures, Plots, Chairmarks, Debugger
```

Настроить %UserProfile%\.julia\config\startup.jl:

```
atreplinit() do repl
  @eval using OhMyREPL
  @eval using Unitful
  @eval using LinearAlgebra
  @eval using Chairmarks
  @eval using Debugger
  @eval using DataStructures
  @eval using About
  @eval using Revise
end
```

Установить расширение Julia для VSCode (пример установки).

После установки создать новый test1.jl файл, набрать в нём 1+1, и нажать Ctrl+Enter, должно появиться всплывающее сообщение с ответом 2, по Esc пропадёт; в открывшемся терминале также должен появиться ответ 2 (при первом запуске, или после обновления, период ожидания до ответа может быть продолжительным).

4.6 Система контроля версий Git

git — это распределённая система контроля версий для любых текстовых файлов; не подходит для бинарных файлов. Изменения отслеживаются построчно, на основе хешей строк.

Инициализация репозитория в текущей папке (.):

```
git init .
```

После создания или изменения файлов, системе *git* необходимо указать на них — какие именно файлы должны быть сохранены:

```
git add some-file some-another-file
```

Сохранение изменений — коммит (*commit*):

```
git commit -m "Some commit message"
```

Note: Рекомендуется сообщение в коммите начинать с заглавной буквы, и опускать финальную точку.

По сути, этих трёх команд достаточно для ведения истории изменений текстов или кодов в своём локальном репозитории при одиночной разработке. Другие команды *git* нужны для более точного отслеживания изменений, ведения нескольких параллельных историй (веток), для слияния с изменениями созданными другими разработчиками.

Также бывают полезными следующие команды:

- `git status` — посмотреть какие файлы были изменены;
- `git diff` — посмотреть непосредственные изменения;
- `git diff --cached` — посмотреть непосредственные изменения, которые уже были добавлены командой `git add ....`;
- `git log` — посмотреть историю изменений (лог);
- `git log -p` — посмотреть полную историю изменений (лог);
- `git log -p some-file-name` — посмотреть полную историю изменений (лог) одного файла;

Также очень полезной практикой является копирование в свой репозиторий подходящего файла `.gitignore` из списка https://github.com/github/gitignore, после чего команда `git status` не будет показывать наличие неотслеживаемых (*Untracked*) ненужных файлов в репозитории. А команда `git add` (без явного указания конкретных файлов) не будет пытаться добавлять всякие сопутствующие бинарные или временные файлы. *Note:* Добавлять все подряд файлы (`*`) в репозитории это плохо из-за того, что будут добавлены бинарные файлы, временные файлы и т.п., после чего их из истории репозитория уже будет не удалить, размер репозитория вырастет без какой-либо пользы.

4.6.1 Ветки (*branches*)

Логика воркфлоу с ветками состоит в том, что разработка и отладка чего-либо ведётся не в главной *master* ветке, а в других, (временных) ветках. При таком подходе в *master* всегда находится нормальный, рабочий код. Плюс, ветки позволяют разрабатывать код в несколько параллельных потоков, независимо в разных ветках.

Ветки создаются командой

```
git branch branch-name
```

После чего изменения будут уже вноситься в новую ветку *branch-name*, а файлы в исходной ветке останутся без изменений. Переключаться между ветками можно командой

```
git checkout another-branch-name
```

Пока изменения не будут внесены (*commit*) переключиться на другую ветку не удастся.

После того, как нужный функционал был внесён в ветку, её можно слить (*merge*) с исходной веткой, для этого надо переключиться на исходную ветку, подтянуть туда изменения (если были), и выполнить слияние:

```
git checkout master
git pull origin master
git merge branch-name
```

Note: При слиянии есть возможность собрать все коммиты в один коммит для того, чтобы история не захламлялась промежуточными, возможно даже неработоспособными, состояниями: git merge --squash branch-name.

При слиянии веток возможно возникнут конфликты, если файлы в исходном репозитории изменились после создания ветки *branch-name*. В этом случае git предложит провести трёхстороннее слияние в meld или winmerge. После слияния изменения в файле или файлы необходимо сохранить в новом коммите:

```
git status
git add ....
git commit -m "Merge upstream commits"
```

4.6.2 Особенности команды *git add*

Изменённые после последнего коммита строки видны командой git diff.

После применения git add .... изменённые строки попадают в кеш, после чего они будут видны командой git diff --cached, а команда git diff покажет пустой список.

При дальнейшем редактировании файлов изменённые строки, те что ещё не добавлены add, будут видны git diff, а закешированные git diff --cached.

В коммит попадают только данные, закешированные командой git add ....

4.6.3 github.com

Сайт https://github.com/ даёт возможность бесплатного хранения своих репозиториев онлайн. Плюс развитая система ведения багов (*issues*), пул-реквестов (*pull request — PR*), возможность ведения релизов, вики-страниц, возможность отслеживания изменений.

Большая доля программ с открытым исходным кодом хостится на *github.com*. Сейчас персональная страница *github.com* выступает даже как портфолио для современных разработчиков.

Перед работой с github потребуется настроить *ssh* ключи для скачивания и загрузки репозиториев (через *https* изменения в репозитории не внести), также потребуется настройка

двух-факторной аутентификации. Всё это подробно пошагово описано в руководстве по созданию аккаунта на github, и созданию своего первого репозитория.

Некоторые особенности работы с github:

- при работе с github помимо создания нового репозитория, есть возможность загрузить туда существующий репозиторий;
- абсолютно желательно явно указывать лицензию на свой код, тексты или данные. Выбрать лицензию можно на странице public-license-selector;
- всегда следует считать репозиторий на github "основным", а себе подтягивать (клонировать) репозиторий на компьютер для внесения изменений;
- для внесения изменений в "чужие" репозитории следует создавать его форк (*fork*) на github, делать правки, и создавать пул-реквест в исходный репозиторий.

4.6.4 git в VSCode

В VSCode есть поддержка git "из коробки": есть возможность добавлять файлы, делать коммиты, переключаться по веткам; в общем большинство операций доступны из интерфейса VSCode, необходимо лишь чтобы git был установлен.

4.7 Полезные консольные утилитки

4.7.1 *TL;DR — Too Long; Didn't Read*

tldr — https://tldr.sh/.

Набор небольших справок по командам Windows, Linux, и т.д., по страничке буквально. В ней более 5000 справок, и, скорее всего, есть справки по всем командам, которые упоминаются в этом руководстве.

Установка (*Note:* программа называется *tldr*, а пакет называется *tlrc*):

```
scoop install tlrc
tldr --gen-config > %UserProfile%\AppData\Roaming\tlrc\config.toml
```

После отредактировать в конфиге %UserProfile%\AppData\Roaming\tlrc\config.toml в секции [cache] опцию languages = ['ru','en'], чтобы, по возможности, были справки на русском языке.

Note: при первом запуске скачивает с инета архив со справками и распаковывает, занимает меньше минуты.

Использование:

```
tldr robocopy
```

4.7.2 robocopy

Встроенная утилита Windows для копирования файлов и папок с сохранением прав и времени создания. Применяется командой:

```
robocopy d:some-dir e:some-another-dir /E /DCOPY:T
```

"some-dir" будет скопирована в *"some-another-dir"*, получив в результате *"some-another-dir\some-dir"*.

4.7.3 fd — поиск файлов

Установка:

```
scoop install fd
```

Использование

Утилита fd в качестве шаблона поиска файлов использует регулярные выражения (по умолчанию), поэтому вместо классического шаблона find -iname '*PATTERN*' следует использовать ~~банальный~~ удобный:

```
fd PATTERN
```

то есть для поиска курсовой набрать: fd курсов, или, если не нашлось, может так поможет: fd kursov.

fd считает свой шаблон в режиме *ignorecase*, если только в шаблоне нет заглавных букв, иначе шаблон *casesensitive*. Опция -i задаёт режим *ignorecase*.

Для поиска по привычным шаблонам со звёздочкой (*) применяется опция -g/--glob:

```
fd --glob "task*.py"
fd -g "task*.{h,c,c++,cpp}" Projects
```

Эта же опция применяется для поиска по *точному имени*.

Поиск файлов по расширению:

```
fd --extension docx
fd -e docx
```

Включить в поиск скрытые (системные) папки и файлы:

```
fd --hidden -g "*.conf" %UserProfile%
```

Поиск с одновременным выполнением какой-либо команды над найденным файлом:

```
fd -e doc %HOME% --exec soffice -convert-to docx {}
```

```
fd --hidden "\.vhdx$" C:\ --exec ls -l {}
```

Опция -t, --type позволяет искать отдельно папки, файлы, и т.д., по типу: f/file, d/directory, l/symlink, x/executable, e/empty, например, найти все *git* репозитории на диске *C:* :

```
fd --hidden -g "\.git" C:\ -t d
```

Note: fd в поиске учитывает спецификации настроек проектов, *git*, и прочего; то есть, если в .gitignore указаны файлы или папки, то fd пропустит их, даже если они удовлетворяют критериям поиска. Для обхода есть опция -I, --no-ignore.

4.7.4 ripgrep и rga

ripgrep (rg)
Быстрый поиск по текстовым файлам.

rga
rg all — поиск по документам: pdf, docx (но не doc), html, ipynb, внутри архивов. rga внутри использует rg для поиска по распакованному содержимому документов, и, все опции rg, это и опции rga.

```
scoop install ripgrep
scoop install pandoc
scoop install poppler
scoop install rga
```

Настройка: создать переменную окружения RIPGREP_CONFIG_PATH указывающую на файл %HOME%\.config\ripgreprc; создать этот файл с конфигурацией:

```
--smart-case
--auto-hybrid-regex
--glob=!*~
--glob=!*~?
# https://github.com/BurntSushi/ripgrep/blob/master/FAQ.md#how-do-i-configure-ripgreps-colors
--colors=path:fg:blue
```

С такой конфигурацией, если в шаблоне нет заглавных букв, то rg будет считать шаблон *ignorecase*, иначе шаблон *casesensitive*. Строчка конфигурации *--auto-hybrid-regex* указывает

применять разные библиотеки *regex* в зависимости от сложности шаблона — для простых шаблонов это значительно ускоряет поиск. Применённая опция --glob=!*~ указывает пропускать (!) файлы оканчивающиеся на ~.

Использование

Как и fd, rg производит поиск по шаблонам в форме регулярных выражений, поэтому не нужны какие-либо звёздочки, а надо прямо набирать:

```
rg sometext
```

Поиск по файлам определённого типа опцией g/--glob:

```
rg --glob "*.py" задача
rg -g "*.py" -g "*.ipynb" задача
rg задача -g "*.{c,c++,h}"
```

```
rga -g "*.{docx,pdf}" отчет
```

Note: Для rga тип документов для поиска можно задать выбором адаптера (просмотрщика) с помощью опций таких как rga --rga-adapters=pandoc,zip, но, это достаточно широкие классы документов, поэтому лучше задавать по расширению, как было сделано выше. Адаптеры rga:

- *pandoc* отвечает за: .epub, .odt, .docx, .fb2, .ipynb, .html, .htm;
- *poppler* за .pdf;
- *decompress*: .als, .bz2, .gz, .tbz, .tbz2, .tgz, .xz, .zst;
- *tar*: tar;
- *zip*: .zip, .jar

Поиск по целым словам опцией -w/--word-regexp:

```
rg -w argparse -g "*.py"
```

Вывести только имена файлов содержащих искомое — опция -l/--files-with-matches.

Вывести контекст найденного, это опции A/B/C — *After*, *Before*, *Context*. Следующее выражение выводит полную историю команд, а затем rg ищет команды над *init.lua* и выводит не только саму команду, но и ближайшие 5. Это выражение, так же, показывает особенность регулярных выражений — в них точка обозначает любой символ, а вот для поиска самой точки её необходимо экранировать:

```
history | rg -C5 "init\.lua"
```

Опять же, аналогично fd, поиск по скрытым файлам с опцией --hidden.

4.7.5 parallel

GNU **parallel**, это утилита, которая: запускает в параллель несколько задач; отслеживает их завершение, чтобы запустить следующую в очереди задачу; отслеживает вывод на экран, чтобы информация не выводилась вперемешку; может разбивать большие файлы для обработки в параллель по кусочкам; и многое другое. Википедиа, Tutorial.

В Windows может использоваться parallel из *WSL*:

```
wsl parallel -j4 "cmd.exe /c magick convert {} -resize 1920x1080  resized_{}" ::: *.jpg
```

```
ls -1 *.jpg | wsl parallel -j4 "cmd.exe /c magick convert {} -resize 1920x1080 {.}.png"
```

В этих командах утилита обработки изображений *magick* (из пакета *ImageMagick*) запускается в параллель. Здесь:

- -j4 определяет число параллельно запускаемых задач;
- ::: отделяет запускаемую команду от аргументов для *parallel*;
- {} — строка замены, на место которой будет подставлен аргумент для распараллеливаемой команды;
- {.} — строка замены, в которой будет удалено расширение файла;
- используется cmd.exe /c чтобы не пришлось прописывать полные пути до запускаемой команды, так как wsl не знает путей *PATH* Windows;

Note: Установка *parallel* в *WSL*: wsl sudo apt install parallel.

4.7.6 time

time — утилита из мира Linux, позволяющая измерить время выполнения программы.
Установка: scoop install time.

Использование вполне обычное, только необходимо использовать команду timecmd, т.к. имя *time* занято — это встроенная команда Windows с другим функционалом:

```
timecmd make all
```

Note: При использовании современной темы оформления оболочки (shell) подобная утилита не требуется, так как сам терминал пишет затраченное время.

Ещё существует утилита hyperfine, это расширенный аналог time.
Установка: scoop install hyperfine.

Использование аналогично утилите time:

```
hyperfine make all
```

4.7.7 top

Утилиты схожие с top: btop, wttop

```
scoop install btop
scoop install wttop
```

4.7.8 ccat

Утилита cat, которая выводит содержимое файлов с подсветкой, как в приличных редакторах:

```
scoop install ccat
```

4.7.9 mc — Midnight Commander

Классика на Windows тоже работает, и умеет *Shell link...*.

```
scoop install extras/mc
```

Note: Сменить диск: Alt+D

Note: Терминал в mc по Ctrl+O нормально не работает.

4.7.10 Остальные Linux утилиты

Большое количество стандартных Linux консольных утилит входит в пакет *BusyBox*.

Note: **BusyBox** — это набор Linux *команд*, скомпилированных для Windows, которые поддерживают wildcard globbing как в Linux.

Установка:

```
scoop install busybox
```

Общие опции:

- -h — [h]uman, выводить размеры в *Kb (KiB)*, *Mb (MiB)*, и т.д., а не в байтах;

***Note**:* ls из состава BusyBox не умеет показывать файлы с русскими буквами в имени файла. Поэтому, её следует заменить на утилиту ls из состава приложения git:

```
scoop shim add ls %UserProfile%\scoop\apps\git\current\usr\bin\ls.exe
```

Пока не пофиксят этот бar, данную процедуру придётся повторять после каждого обновления BusyBox, благо такие обновления могут быть проведены только вручную командой scoop update --all.

4.7.10.1 BusyBox команды

[[[ar arch ascii ash awk base32 base64 basename bash bc bunzip2 busybox bzcat bzip2 cal cat cdrop chattr chmod cksum clear cmp comm cp cpio crc32 cut date dc dd df diff dirname dos2unix dpkg dpkg-deb drop du echo ed egrep env expand expr factor false fgrep find fold free fsync ftpget ftpput getopt grep groups gunzip gzip hd head hexdump httpd iconv id inotifyd install ipcalc jn kill killall lash less link ln logname ls lsattr lzcat lzma lzop lzopcat make man md5sum mkdir mktemp mv nc nl nproc od paste patch pdpmake pdrop pgrep pidof pipe_progress pkill printenv printf ps pwd readlink realpath reset rev rm rmdir rpm rpm2cpio sed seq sh sha1sum sha256sum sha3sum sha512sum shred shuf sleep sort split ssl_client stat strings su sum sync tac tail tar tee test time timeout touch tr true truncate ts tsort ttysize uname uncompress unexpand uniq unix2dos unlink unlzma unlzop unxz unzip uptime usleep uudecode uuencode vi(tutorial) watch wc wget which whoami whois xargs xxd xz xzcat yes zcat

Note: Подробное описание по командам на русском языке можно найти на сайте https://www.opennet.ru/man_1.shtml.

Некоторые утилиты:

4.7.10.2 du и df

du — Disk Usage — показывает объём, занимаемый файлами и/или папками; опция -s — суммарный размер, а не каждого элемента.
df — Disk Full — показывает объём дисков, полный, свободный, занятый.

Посмотреть занимаемое место файлами и/или папками:

```
du -sh .
du -sh some-folder
du -sh * | sort -h
```

Посмотреть общее и занятое место на дисках:

```
df -h
```

4.7.10.3 head и tail

head -n3 *.txt — вывести первые три строки;
head -n-3 *.txt — вывести все строки кроме последних трёх;
tail -n3 *.txt — вывести три последних строки;
tail -n+3 *.txt — вывести все строки кроме первых трёх;

tail -n+1 *.txt — вывести содержимое всех файлов, предварив названием файла.

4.7.10.4 paste

Объединяет несколько текстовых файлов таким образом, что в объединённом файле каждая строка состоит из строки первого файла, затем идёт разделитель (*–delimiter*), затем строка второго, и т.д. Данная утилита удобна для объединения нескольких файлов мониторинга (лог-файлов) в один файл.

```
paste -d' ' data1.out data2.out > data.out
```

4.7.10.5 join

Объединяет два текстовых файла по одинаковым значениям в первом поле каждой строки — объединяет данные по первому столбцу. Данные должны быть отсортированы по объединяемому полю. Данная утилита удобна для объединения нескольких файлов мониторинга (лог-файлов) в один файл.

```
join data1.out data2.out > data.out
```

4.7.10.6 wget

Скачивает с инета страницу или файл: wget URL

Скачать сайт: -r — recursive, -np — no-parent, -p — также скачать всё что требуется для отображения страницы, -c — продолжить (continue) закачку с места разрыва, если он случится, --restrict-file-names=nocontrol — после загрузки не экранировать русские буквы в именах файлов. URL не обязательно должен быть корневой страницей.

```
wget -r -np -p -c --restrict-file-names=nocontrol URL
```

4.7.10.7 xargs

Применить команду к каждой строке в списке:

```
fd --hidden "\.vhdx$" %HOME% | xargs -I@ cp -v @ E:\backup\wsl\
```

4.7.10.8 watch

Опция -d подсвечивает изменения.

watch -d df -h — отслеживать заполненность дисков; по умолчанию, периодичность 2 секунды.

<TBD>

watch -d 'cat /proc/vmstat | grep numa' — отслеживать результат периодического выполнения команды.

4.7.11 Совпадающие системные имена

С помощью алиасов можно настроить нормальные имена для Linux утилит, имена которых совпадают с именами системных Windows утилит.

Например, настроить имя для утилит sort, time. Для *Clink* добавить строчки в *"%LOCALAPPDATA%\clink\clink_start.cmd"*:

```
doskey sort=%UserProfile%\scoop\shims\sort.exe $*
doskey time=%UserProfile%\scoop\shims\time.exe $*
```

Note: Для *PowerShell* смотри соответствующий пункт главы Настройки Windows Terminal.

5 Работа с документами

5.1 Офисные пакеты

Существует, как минимум, два офисных пакета с открытым исходным кодом: LibreOffice и OnlyOffice, которые позиционируются как вполне разумная замена MSOffice для вариантов персонального использования. У MSOffice есть заметные преимущества в совместной, распределённой работе над документами в организациях, с поддержкой сервиса SharePoint/OneDrive, и т.п., но и здесь появились альтернативы. У всех пакетов есть ленточный/ribbon интерфейс схожий с MSOffice. Все поддерживают стили MSOffice, отображение и набор формул.

Note: у всех офисных пакетов есть некоторое несовпадение в отображении открытых документов с MSOffice, которое связано с тем, что пользователи оформляют документы и презентации не на основе стилей, а вручную подгоняя вид пробелами и отступами. Если же использовать правильный, рекомендуемый Microsoft, поход к оформлению документов, то различия в отображении документов будут минимальны, а скорее всего, вообще неразличимы.

Общие рекомендации при использовании офисных пакетов:

- Использовать исключительно стили для оформления. Пора забыть про пробелы и табы для выравнивания!
- Шаблоны — это основа любого документа!

- При редактировании чужого документа обязательно включать режим правок/рецензирования!
- Якори — это плохо!
- *.doc* давно пора закопать!
- Пора обновить свой офисный пакет, ему уже почти 20 лет, там всё стало намного удобнее.

5.1.1 LibreOffice

LibreOffice
Консервативный офисный пакет для персонального использования. Превосходно справляется с конвертацией документов из/в форматов MSOffice; единственное бесплатное приложение, которое умеет правильно читать документы старого формата *.doc*.

```
scoop bucket add extras
scoop install extras/libreoffice
scoop shim add soffice %UserProfile%\scoop\apps\libreoffice\current\LibreOffice\program\soffice.exe
scoop shim add loffice %UserProfile%\scoop\apps\libreoffice\current\LibreOffice\program\soffice.exe
```

Note: В последних версиях есть *ленточный (ribbon)* стиль оформления интерфейса, похожий на MSOffice; включение через *View* > *User Interface: UI Variants* > *Tabbed*; выключение через это же меню, которое будет расположено в правом верхнем углу в виде трёх полосочек.

Note: для мониторов с высоким разрешением (High DPI) потребуется переключиться на *SVG* набор иконок в меню (<Alt-F12>) *Tools* > *Options* > *LibreOffice* > *View*. *Ленточный (ribbon)* интерфейс не совместим с мониторами с высоким разрешением.

Применение для конвертации *.doc* в *.docx* (с обновлением в поддерживаемый формат MSEquation), в текущей папке:

```
soffice -headless -convert-to docx *.doc
```

То же, рекурсивно по папкам:

```
fd -e doc --exec soffice -convert-to docx {}
```

Вывести содержимое *.doc*/*.docx* на экран:

```
soffice -cat somedoc.doc
```

5.1.2 OnlyOffice

OnlyOffice — офисный пакет от Российских разработчиков, участвует в программе импортозамещения. Opensource, кроссплатформенный, имеет версию для Android и iOS. В качестве

формата хранения документов использует форматы файлов MSOffice (какого-то своего особенного формата не имеет). Умеет редактировать *pdf*. Установка:

```
winget install --id=ONLYOFFICE.DesktopEditors -e
```

Помимо десктоп версии, у OnlyOffice есть открытая версия офиса для облачного, совместного использования на платформе частного облака NextCloud. Но, данные возможности не имеют смысла при персональном, не коллективном использовании, буквально избыточны; поэтому, здесь только ссылка на guide по установке для желающих попробовать.

Также у OnlyOffice есть OnlyOffice-Server, предназначенный для организации облачного сервиса, аналога Office365 и SharePoint/OneDrive, с поддержкой распределённой, совместной работы.

> *Отступление про редактирование pdf.*
> Внутренний формат pdf не содержит информации о последовательности символов и слов, а содержит информацию, в каком месте страницы нарисовать тот или символ или слово (и, как следствие, пробелы тоже не хранит). Даже в отдельном параграфе слова не обязательно будут в естественном порядке. С математическими формулами или таблицами ситуация совсем катастрофическая. Поэтому, все программы для редактирования pdf “додумывают” информацию о том, в какой последовательности идут символы и слова. Поэтому, самый лучший способ редактирования pdf, это сконвертировать pdf в Markdown методом OCR или ИИ, и полученный Markdown сконвертировать в docx.

5.1.3 Collabora

Collabora Online
Ещё один бесплатный открытый офисный пакет с возможностью совместного, распределённого редактирования документов; эти возможности хорошо интегрированы в платформу частного облака NextCloud. Этот офисный пакет работает исключительно “онлайн”, т.е. его нельзя “установить” на компьютер. Опять же, для персонального использования это избыточно; и здесь упоминается только для полноты обзора.

5.1.4 MSOffice tips

5.1.4.1 Нумерация рисунков в Word

Сначала необходимо вставить рисунок в документ, выбрав соответствующий пункт меню «Вставка» и выбрав рисунок из файловой системы или из другого источника. Затем, после вставки рисунка, необходимо выбрать его и перейти во вкладку «Ссылки» в верхнем меню Word.

Во вкладке «Ссылки» находится группа команд «Подписи», которая содержит кнопку «Подписи для рисунков». При нажатии на эту кнопку открывается диалоговое окно, в котором можно

выбрать опции для автоматической нумерации рисунков. В окне можно указать формат нумерации, префикс и суффикс для нумерации, а также выбрать стиль подписей для рисунков.

link.

5.1.4.2 Сделать прозрачным фон картинок в PowerPoint

Раньше (2003) этот пункт меню был в быстром доступе на панели работы с рисунками. Сейчас он спрятан в *<TBD>*

5.2 Текстовые документы

На данный момент Markdown это основной формат разметки для текстовых документов. Помимо этого существует ещё пара десятков форматов с аналогичным функционалом (в том числе LaTeX), но после того, как github взял Markdown в качестве своего основного языка разметки, все остальные постепенно вышли из использования.

Ещё существует LaTeX, но это излишне переусложнённо и не читабельно для целей написания простых статей, веб-страниц, или документации к программам. Здесь LaTeX упоминается только из-за его языка разметки математических выражений, который повсеместно распространился, в отрыве от самого LaTeX'а.

Основополагающей идеей Markdown является сохранение, и, даже, повышение читабельности текста, чтобы можно было читать и писать документы в любом простом редакторе. В Markdown есть аннотации для выделения и оформления таких элементов, как: заголовки, параграфы, списки, таблицы, акценты текста, встраивание рисунков, аннотация кода, встраивание математических формул, сноски, ссылки на главы и ссылки на внешние ресурсы. Существуют расширения Markdown для: библиографий и цитирования, рисования диаграмм, выполнения кода скриптов. С одной стороны, это простой текст который можно просматривать хоть выводом cat на экран, а с другой стороны, для Markdown есть множество приложений для полноценного графического отображения содержимого в виде, привычном по офисным редакторам.

В силу того, что Markdown это всё же просто текст, то в нём не содержится каких-либо стилей или графического оформления. Таким занимаются программы для отображения, рендеринга и конвертации. Именно в этих программах будет выбираться размер шрифта, межстрочный интервал и т.п. при конвертации в pdf или docx. Обычно, это применение либо встроенных в программы шаблонов стиля, либо использование шаблонов стиля предоставляемых пользователем.

5.2.1 Markdown

Редактирование **Markdown** и синхронное отображение поддерживается в *VSCode* "из коробки", в том числе, есть отображение картинок, таблиц, формул, сносок, и проч. https://code.

visualstudio.com/docs/languages/markdown. Встроенный в VSCode Markdown Preview можно донастраивать расширениями. Открыть превью в *VSCode*: Ctrl+Shift+V.

Для просмотра без редактирования можно использовать любой браузер с одним из плагинов для просмотра Markdown, например Markdown Viewer. Это расширение предоставляет возможность использовать все возможности Markdown: формулы, рисунки, таблицы, сноски, и т.д.

В Google docs Markdown — это один из форматов для экспорта, причём, при экспорте, рисунки становятся встроенными (embedded) прямо в файл *.md*.

Для конвертации *в* и *из* формата Markdown во всевозможные другие применяется конвертер *pandoc*, о нём будет дальше.

5.2.2 Quarto

Quarto — система вёрстки на основе Markdown, с одновременным отображением отрендеренного содержимого:

- умеет показывать формулы, картинки, код, таблицы;
- может выполнять код на *Julia*, *Python*, *R*, *Observable JS*, и показывать результат выполнения в тексте;
- умеет отображать диаграммы в форматах Mermaid и Graphviz (*dot*);
- формат файлов Quarto (*.qmd*) — это Markdown с некоторыми расширениями, т.е. простой текстовый файл, который можно редактировать в любом текстовом редакторе;
- поддерживается в VSCode расширении, с возможностью синхронного отображения отрендеренного содержимого;
- *.qmd* файлы конвертируются (рендерятся) в форматы *pdf*, *docx*, *pptx*, *html* с учётом заданных стилей:
  - Quarto имеет предустановленные стили, для оформления статей, презентаций, сайтов, и др. В случае отсутствия явного указания стиля будет использован стиль по умолчанию *article*.
  - вместо предустановленных стилей, можно задавать пользовательские стили; их можно указать либо в виде файла шаблона стилей *dotx* или *potx*, либо указать на файл готового документа *docx* или *pptx*, с которого будет скопирован стиль оформления. Для оформления *pdf* можно непосредственно внедрять код *LaTeX* для явного указания стилей и/или пакетов;
- умеет загружать *.ipynb* и рендерить в различные форматы;
- JupyterLab умеет показывать Quarto с полным форматированием;

- может отрендерить обычные файлы с кодом *.py*, *.jl*, *.R* в *pdf*, *html* и т.п., где в файлах в комментариях специального формата (*percent format*) помещены *Markdown* блоки. Данный подход позволяет вести разработку программ привычным способом, а по завершении разработки сгенерировать документ отчёта прямо с файла программы;
- "под капотом" использует pandoc.

Установка:

```
winget install --id=Posit.Quarto -e
python3 -m pip install nbclient
quarto install tinytex
quarto install chromium
```

Можно использовать в VSCode с соответствующим плагином. А можно просто написать *.qmd* файл в любом редакторе, и сконвертировать в выбранный формат: quarto render win11-for-engineers-manual.qmd --to html (как данное руководство).

Выполняемые блоки кода

Помимо отображения блоков кода и математических формул в *LaTeX* нотации в Markdown/Quarto документы можно добавлять выполняемые блоки кода, результат выполнения которых будет подставлен на страницу, для примера, в главе про Matplotlib показан блок кода на питон и результат его выполнения тут же добавлен в документ. Выполняемые блоки кода аннотируются фигурными скобками вокруг названия типа блока кода, например, {python}. Как и в других *Интерактивных блокнотах*, в *Quarto* выполняемые блоки кода имеют единое пространство имён, т.е. объявив переменную в одном блоке, она будет доступна в последующих блоках, причём даже в блоках разного типа (через $ синтаксис):

````
```{julia}
n = 3
```
````

```
3
```

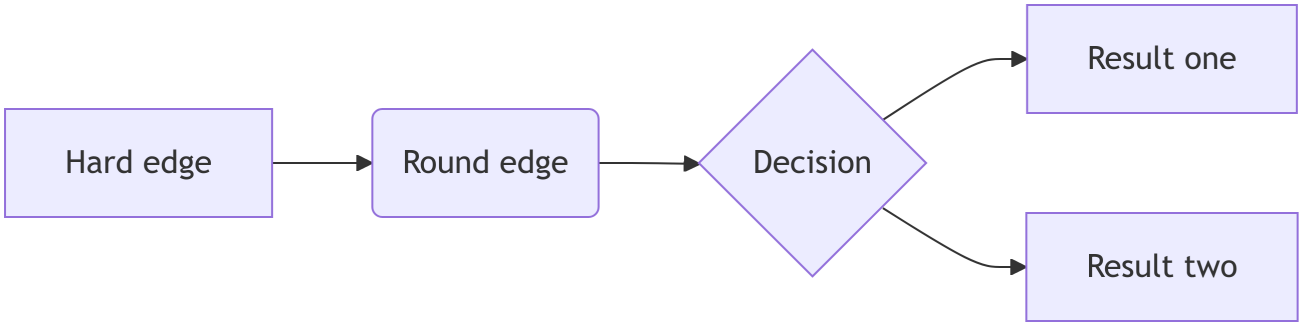
````
```{python}
for i in range(1, $n + 1):
  print("i =", i)
```
````

```
i = 1
i = 2
i = 3
```

Note: В силу бага в ```{python} блоках нельзя объявить #| echo: fenced.

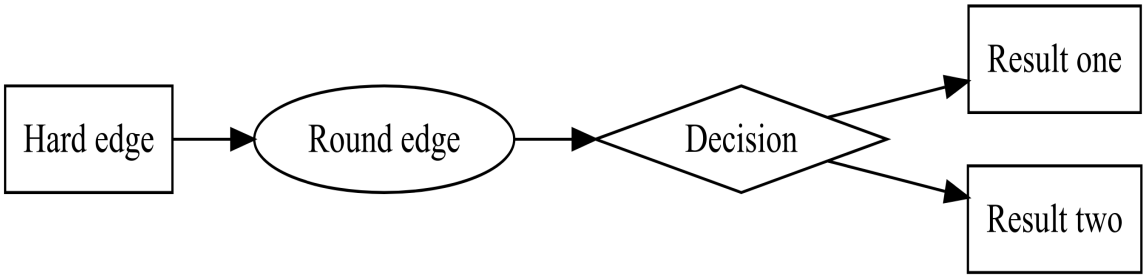
Также, Quarto выполняет и выводит диаграммы, с использованием аннотированных блоков кода в форматах Mermaid (примеры) и Graphviz(dot) (примеры). *Note:* помимо этих двух типов диаграмм в Quarto есть поддержка других типов диаграмм (*TikZ* и др.) через плагины pandoc.

Пример диаграммы Mermaid со страницы документации Quarto. Этот блок кода Mermaid выполняется по месту и отрисовывается вместо кода:



На месте диаграммы в этом документе, на самом деле, стоит код Mermaid код с аннотацией выполнения ({}):

```
```{mermaid}
flowchart LR
  A[Hard edge] --> B(Round edge)
  B --> C{Decision}
  C --> D[Result one]
  C --> E[Result two]
```
```

Аналогично работает отрисовка диаграмм Graphviz (*dot*), здесь показан пример сразу вместе с блоком кода:

```
```{dot}
//| fig-width: 6
//| fig-height: 1.5
digraph G {
  rankdir = LR;

  A [label="Hard edge" shape=box];
  B [label="Round edge" shape=ellipse];
  C [label="Decision" shape=diamond];
  D [label="Result one" shape=box];
  E [label="Result two" shape=box];

  A -> B;
```
```

```
  B -> C;
  C -> D;
  C -> E;
}
```
```




Выполняемые блоки кода для языков *Python*, *Julia* и *R*, по умолчанию, отображаются вместе с результатами их выполнения; выполняемые блоки кода диаграмм по умолчанию не отображаются в результирующем документе. Это можно изменить опцией выполнения #| echo:, что было проделано выше на примере диаграммы Graphviz добавлением опции //| echo: fenced; подробности опций отображения блоков кода и результатов выполнения описаны в документации.

В документе Quarto могут выполняться достаточно большие блоки кода, их результат выполнения будет отрисован в документе, и, вполне возможно, создавать статьи, или, скажем, курсовые работы вместе отчётом, в одном *qmd* файле. При наличии большого числа блоков кода, или длительных вычислений в них, для контроля времени затрачиваемого на выполнения блоков кода во время рендеринга (quarto render) существуют опции запрета на выполнение или выполнения только при изменении кода, кеширования результатов выполнения и другие.

### Некоторые особенности блоков Julia

В современных версиях *Quarto*, для выполняемых блоков кода языка *Julia* есть возможность их выполнения непосредственно в julia (иначе, по умолчанию, используется *Jupyter*). Для этого в заголовке необходимо прописать опцию engine: julia; этого достаточно для Quarto документов с блоками кода *Julia*. Если же в qmd-документе, при этом, присутствуют одновременно ещё *Python*, или *R* блоки, то дополнительно потребуется выполнить следующую команду в текущей директории:

```
julia --project=. -e 'import Pkg; Pkg.add("PythonCall"); Pkg.add("CondaPkg"); Pkg.add("RCall");'
julia --project=. -e 'using PythonCall'
```

что эквивалентно выполнению следующих команд в терминале julia:
```

```
julia> import Pkg
julia> Pkg.activate(".")
julia> Pkg.add("PythonCall")
julia> Pkg.add("CondaPkg")
julia> Pkg.add("RCall")
julia> using PythonCall
```

Эти операции требуются для загрузки пакета *PythonCall*, а также его предзагрузки и инициализации, в процессе которой будет также установлен пакет NumPy.

Для добавления пакетов *Python* следует выполнить команды:

```
julia --project=. -e 'import CondaPkg; CondaPkg.add("numpy");  CondaPkg.add("matplotlib");'
```

или

```
julia> using CondaPkg
julia> # press ] to enter the Pkg REPL
pkg> conda add numpy
pkg> conda add matplotlib
```

Также потребуется установить используемые в выполняемых блоках кода Julia пакеты, например, Pkg.add("Plots").

Но проще всего вставить следующий блок кода где-нибудь в начале документа:

````
```{julia}
#| echo: false
#| include: false
import Pkg

if !(Base.active_project() !== nothing && !startswith(Base.active_project(), Base.Sys.BINDIR))
  Pkg.activate(".")
end

julia_packages = ["CondaPkg", "PythonCall", "Plots", "LaTeXStrings"]

for package_str in julia_packages
  package = Symbol(package_str)
  try
    eval(:(using $package))
  catch
    Pkg.add(package_str)
    eval(:(using $package))
  end
```
````

```
end

using CondaPkg
python_packages = ["numpy", "matplotlib"]

for package in python_packages
  try
    pyimport(package)
  catch
    CondaPkg.add(package)
  end
end

using PythonCall
```

Note: При использовании движка engine: julia (и, соответственно, пакета *QuartoNotebookRunner*) имеет место бar, в связи с чем следует в заголовке в секции *format* объявлять опцию *html* перед *pdf*:

```
format:
 html:
 pdf:
```

Note: В случае возникновения ошибки про *Qt6Base_jll* при выполнении команды quarto render следует выполнить эту команду повторно.

Note: Хотелось бы отметить, что смешение языков выполняемых блоков в одном qmd-документе — это плохая идея.

5.2.3 *LaTeX*

Существует сайт overleaf.com — это онлайн-редактор *LaTeX* с поддержкой автоматического рендеринга (отображения) результата, контроля версий, совместного редактирования. Если требуется написать статью на *LaTeX*, а не знаешь как, то этот сайт это правильный выбор. А если знаешь как, то какие-либо советы излишни.

5.2.4 Универсальный конвертер Pandoc

pandoc — инструмент командной строки (CLI) для конвертации документов в различных форматах в документы других различных форматов: pdf, docx, pptx, html, Markdown могут быть сконвертированы друг в друга почти без ограничений, в том числе и с сохранением математических формул. Поддерживает перенос библиографий, цитирования, стилей оформления,

таблиц в другие форматы файлов. Есть ограничение: pdf нельзя сконвертировать в другие форматы. Установка:

```
scoop install pandoc
```

Примеры использования:

```
pandoc --from=html --to=markdown --wrap=none some-saved-site-page.html
```

```
pandoc --to=markdown --wrap=none mine-article-with-math-formulas.docx
```

```
pandoc -s math.tex -o example30.docx
```

Опцию --from использовать необязательно, pandoc в большинстве случаев автоматически поймёт исходный формат по расширению или содержимому. Выходной формат надо обозначать в случаях неоднозначного соответствия расширения файла формату, как в случае с *.md*, которому соответствует несколько диалектов Markdown. См. руководство https://pandoc.org/MANUAL.html.

5.3 Программы для ведения заметок

Приложения для ведения заметок разделяются по способу хранения базы заметок: один локальный файл с базой, множество локальных *.md* файлов и картинок, облачное хранение. Выбор программы во многом определяется выбором способа хранения.

Среди множества существующих программ для ведения заметок следует рассматривать только opensource программы, которые используют формат Markdown и поддерживают формулы в LaTeX нотации: https://alternativeto.net/category/productivity/note-taking/. Программ много, можно выбрать на любой вкус.

5.3.1 Logseq

Пример приложения с хранением каждой записи в отдельном файле — **Logseq**. Считается свободной альтернативой *Obsidian*. Гайд по LogSeq.

```
winget install --id=Logseq.Logseq  -e
```

5.3.2 dendron

Ещё одно приложение с хранением каждой записи в отдельном файле — плагин **dendron** для VSCode. Это приложение в качестве основной особенности указывает удобный поиск и

навигацию по базе в десятки тысяч заметок. Инструкция.

5.3.3 TiddlyWiki

В качестве программы для ведения заметок с одним локальным файлом можно рассмотреть **TiddlyWiki** — https://tiddlywiki.com. Это браузерное JavaScript приложение в виде одного *.html* файла, т.е., чтобы посмотреть или отредактировать заметки, не потребуется установка каких-либо программ, а всего лишь открыть этот файл в браузере. Поддерживает язык разметки Markdown и формулы LaTeX. Минус этого приложения автоматически вытекает из его плюса — так как база заметок это один файл, то при добавлении картинок этот файл быстро разрастается до десятков, если не сотен мегабайт, и становится тормозным. А если вставлять картинки в виде ссылок на файлы, то теряется концепция хранилища в виде одного файла. Можно использовать векторные картинки, например в формате SVG, но их сложно делать, а tikz не поддерживается. Так же из минусов — это отсутствие Autosave "из коробки".

Для удобства использования потребуется настроить Autosave для TiddlyWiki: в *Firefox* установить плагин *Timimi*, также установить приложение *timimi-2-1-1-Windows-Firefox.exe*. Autosave работает только в *Firefox*, в *Chromium* почему-то не работает.

Скачать https://tiddlywiki.com/#GettingStarted — пустой файл заметок. Открыть в *Firefox*. Установить плагины:
CodeMirror
CodeMirror Close Brackets
CodeMirror Mode Markdown
CodeMirror Search and Replace
Highlight
KaTeX
Markdown

Note: Есть поддержка режима редактирования Vim в плагине *CodeMirror Keymap Vim*. Но для *очень* продвинутых пользователей этот плагин не актуален, так как они уже настроили себе расширение https://github.com/glacambre/firenvim.

5.3.4 AI-based

Данный пункт написан при поддержке LLM.

Появляются приложения со встроенными ИИ-помощниками (ассистентами). Разумно предположить, что в недалёком будущем это станет мейнстримом — облегчение человеческой деятельности в областях монотонной или однообразной работы.

Одним из таких приложений с интеграцией ИИ-ассистента является Reor, это опенсорс приложение для ведения заметок. В Reor встроена утилита Ollama, которая позволяет использовать локальную языковую модель (LLM). Также поддерживается подключение внешних LLM через OpenAI-совместимый протокол. *Note:* Для работы с локальными LLM для приложения требуется мощный компьютер, точнее, требуется наличие видеокарты.

Заметки в Reor хранятся в виде отдельных Markdown-файлов (*.md*), организованных в папках. Пользователь может создавать подпапки для структурирования информации. При установке можно указать папку с уже существующими Markdown-заметками, и Reor продолжит работу с ними, не нарушая текущую организацию файлов.

В процессе работы Reor выступает как RAG AI-ассистент (Retrieval-Augmented Generation). Это означает, что встроенная языковая модель (LLM) анализирует содержимое заметок и на основе этого предоставляет дополнительные функции: отвечает на вопросы, учитывает контекст заметок, автоматически устанавливает семантические связи между ними и предлагает подсказки при написании текста. Подробнее на официальной сайте.

Установка Reor:

```
winget install --id=ReorProject.Reor -e
```

Note: В целом, на данном этапе развития, не стоит ожидать от таких приложений чего-то выдающегося. Да и шумит компьютер сильно при работе с локальными LLM.

Подробнее про используемые термины и технологии в Приложение: AI/LLM.

5.4 Обработка картинок и создание видео

5.4.1 ImageMagick

ImageMagick — пакетная обработка файлов изображений из командной строки.

```
scoop install imagemagick
```

Note: GraphicsMagick — тоже самое, что и ImageMagick, но работает быстрее — лучше распараллеливается, но, может ~~*глючить*~~ *выполнять обработку с ошибками*. Запускается в точности как ImageMagick, только с префиксом *gm*, а не *magick*, например: gm mogrify .... ~~Использовать стоит только в случае обработки очень большого количества файлов (сотни и больше), и убедившись, что GraphicsMagick выполняет всё идентично с ImageMagick~~ Использовать не стоит, так как для параллелизации есть другие инструменты, например, *GNU parallel*.

ImageMagick применяется как magick operation ..., где *operation*:

- convert преобразует изображение и записывает результат в другой указанный файл.
- mogrify модифицирует исходный файл.
- montage собирает (монтирует) несколько изображений в один файл. По умолчанию уменьшает размер до 120x120. montage -mode concatenate -tile 1x -geometry 1920x1080 *.jpg out.jpg — соединить все изображения в одну вертикальную ленту, отмасштабировав в один размер 1920x1080 с сохранением пропорций.

Note: Порядок опций имеет значение — несколько действий будут выполняться друг за другом, а опции, вроде *-gravity*, или указания цвета, имеют влияние только на следующие за ними команды.

Note: префикс magick можно опускать и применять непосредственно соответствующие утилиты; кроме convert, имя которой совпадает с одноимённой утилитой Windows.

Таблица 1: Указание размеров изображения для утилит *ImageMagick* (далее показаны на примерах).

Размер	Общее описание[5]
*scale***%**	Высота и ширина масштабируются на указанный процент.
*scale-x***x***scale-y***%**	Высота и ширина масштабируются отдельно на указанные проценты.
*width***x**	Ширина задана, высота автоматически подбирается для сохранения соотношения сторон.
x*height*	Высота задана, ширина автоматически подбирается для сохранения соотношения сторон.
*width***x***height*	Заданы максимальные значения высоты и ширины, соотношение сторон сохраняется.
*width***x***height***^**	Заданы минимальные значения ширины и высоты, соотношение сторон сохраняется.
*width***x***height***!**	Ширина и высота заданы явно, исходное соотношение сторон игнорируется.
*width***x***height***>**	Уменьшает изображение, если его размер(ы) **больше** указанных ширины и/или высоты.
*width***x***height***<**	Увеличивает изображение, если его размер(ы) **меньше** указанных ширины и/или высоты.
*area***@**	Изменяет размер изображения до указанной площади в пикселях. Соотношение сторон сохраняется.
*x***:***y*	Здесь x и y обозначают соотношение сторон (например, 3:2 = 1.5).
*x***:***y***^**	Удаляет строки или столбцы для достижения заданного соотношения сторон.
*x***:***y***#**	Добавляет строки или столбцы для достижения заданного соотношения сторон.
{*size***}{***offset***}**	Указание смещения (по умолчанию *+0+0*). Ниже *{size}* относится к любой из вышеуказанных форм.

Размер	Общее описание
{*size***}{**+-**}***x***{**+-**}***y*	Горизонтальное и вертикальное смещения x и y, указанные в пикселях. Знаки обязательны для обоих. Смещения зависят от опции *-gravity*[6]. Смещения не зависят от *%* или других операторов размера. Обратите внимание, что положительные смещения *X* и *Y* направлены внутрь к центру изображения для всех опций *-gravity*, кроме 'Center'. Для 'East' *+X* — влево. Для 'South' *+Y* — вверх. Для 'SouthEast' *+X* — влево и *+Y* — вверх. Для 'Center' используется обычное направление *X* и *Y* (*+X* — вправо, *+Y* — вниз).

Отступление **об экранировани символов**

Особенности экранирования аргументов команд в командной строке в Windows и Linux. Помимо обычных символов перенаправления потоков (>, <, |) в Windows особое значение имеют символы: ^, % (^ используется для экранирования других символов, а % используется для разыменования переменных). Они могут быть использованы в качестве аргументов следующим способом: там где нужен % надо писать %%, а ^ должен быть экранирован кавычками, иначе будет проигнорирован. В Linux особые символы !, (,), \; для использования в качестве аргументов они должны быть экранированы либо кавычками, либо с помощью \; и сам символ \ тоже должен быть экранирован как \\. Также, в Windows и Linux одинарные и двойные кавычки имеют противоположную роль: в Linux одинарные кавычки более "сильные", чем двойные; в Windows наоборот. Поэтому при использовании в Windows команд из мира Linux надо заменять ' на ", и наоборот.

См. также пункт Ограничения командной строки windows.

5.4.1.1 Примеры

Конвертация всех *jpg* в меньшее качество — *75* (по умолчанию 92), задать имена выходных файлов как **-converted.jpg*: *CHEKME:*

```
magick convert -quality 75 -set filename: "%t" "%[filename:fname]-converted.jpg" *.jpg
magick convert -quality 75 -set filename:fname "%t" "%[filename:fname]-converted.jpg" *.jpg
```

Отмасштабировать картинки с сохранением *aspect ratio*:

```
mogrify -resize 50%% *.png
```

[5] Фактическое поведение может варьироваться в зависимости от различных опций и настроек.
[6] Подробности для опции *-gravity* см. в руководстве по программе.

```
mogrify -resize 1920x1080 *.png
```

Без сохранения соотношения сторон (здесь кавычки для экранирования символа !, который в некоторых терминалах имеет специальное значение):

```
mogrify -resize "1920x1080!" *.png
```

Конвертация всех изображений из папки *images* в *jpeg* формат, с масштабированием всех картинок в размер 1920 пикселя по ширине, с сохранением в папку *images-transformed*:

```
magick convert images -resize 1920x JPEG:images-transformed
```

Обрезать рамки одного цвета по краям картинки, уменьшить, и добавить рамку белого цвета шириной 8 пикселей (опция *+repage* для обновления внутренней системы координат *png* при обрезке):

```
mogrify -trim -resize 50%% -bordercolor white -border 8x8 +repage *.png
```

Обрезать под заданный размер равномерно к центру, который смещён от геометрического центра на заданную величину (это кадрирование под нужный размер):

```
mogrify -gravity Center -crop 1920x1080+40+100 +repage *.png
```

Отмасштабировать в размер 1920x1080 с сохранением пропорций, недостающие поля нарастить белым цветом от центра (для прозрачности использовать *none*), указать тип *png* палитры цветов Truecolor без прозрачности (для палитры с прозрачностью задать *6*):

```
mogrify -resize 1920x1080 -background white -gravity Center -extent 1920x1080  +repage -define 'png:color-type=2' *.png
```

Скомпоновать картинки в *pdf*:

```
magick *.jpg output.pdf
```

5.4.1.2 Анимированный gif

Собрать все картинки в текущей директории в анимированный *gif*. Опция *-delay* задаёт частоту кадров в форматах: *N* по 10 миллисекунд, или, *KxL*: *L* кадров на *K* секунд. По умолчанию *-delay 1* — 100 кадров в секунду, но эта величина зависит от версии ImageMagick, поэтому лучше указывать её явно.

```
magick -delay 5 *.jpg movie.gif
```

```
magick -delay 1x20 *.png movie.gif
```

5.4.2 ffmpeg

ffmpeg — программа командной строки для создания, записи, конвертации, и проигрывания видео и аудио файлов.

Установка:

```
scoop install ffmpeg
scoop install handbrake-cli
```

5.4.2.1 Создание видео из кадров

Создать видео из набора кадров , *“velmagfield-%04d.png”* — шаблон входных файлов:

```
ffmpeg -framerate 20 -i "velmagfield-%04d.png" -r 20 -hide_banner -c:a copy -c:v libx264 -crf 18 ^
    -profile:v main -tune animation -preset veryslow -vf format=yuv420p,scale=-1:1080 ^
    velmagfield.mp4
```

Применяемые опции:

- -framerate 20 — задать частоту “кадров” входного потока *png* изображений (соответствует шагу по времени 0.05 секунды);
- -r 20 — задать частоту кадров выходного видео;
- -vf scale=-1:1080 — отмасштабировать видео до *1080p*; результирующее разрешение должно делиться на два;
- -c:a copy -c:v libx264 — задают аудио и видеокодеки, в данном случае аудиокодек будет отсутствовать (он скопирован из входных изображений, где его нет), видеокодек — *H.264*;
- -crf 18 — задаёт качество картинки видео (в ущерб размеру); по умолчанию *23*, *17* — видео неотличимо от исходных изображений;
- -preset veryslow — кодировать видео с особой тщательностью, в ущерб времени; другие варианты: medium, fast, slow;
- -vf format=yuv420p или -pix_fmt yuv420p — кодировать видео в цветовое пространство “4:2:0”, а не в современный вариант “4:4:4”, тогда полученное видео должно проигрываться в PowerPoint и других устаревших плеерах;
- формат выходного файла определяется его расширением; при этом, размер файла практически не зависит от расширения.

Чтобы видео загружалось в *Telegram* как видео, а не GIF, необходимо, чтобы в нём был аудиопоток[7]; закодировать видео с добавлением беззвучного аудиопотока[8] и всеми настройками совместимости с *TG*[9]:

[7] https://awesomeprogrammer.com/blog/2023/01/15/how-to-force-telegram-to-upload-mp4-as-mp4-and-not-gifs/
[8] https://stackoverflow.com/questions/12368151/adding-silent-audio-in-ffmpeg#answer-18700245
[9] https://stackoverflow.com/questions/38702981/which-video-format-is-right-for-sendvideo-method-in-telegram-bot-api/77255128#77255128

```
ffmpeg -framerate 20 -i "velmagfield-%04d.png" -f lavfi -i aevalsrc=0 -hide_banner -r 20 ^
    -max_muxing_queue_size 9999 -c:v libx264 -crf 18 -maxrate 4.5M -preset medium ^
    -flags +global_header -pix_fmt yuv420p -profile:v baseline -movflags +faststart ^
    -c:a aac -ac 2 -map 0:v -map 1:a -shortest velmagfield.mp4
```

Note: У *ffmpeg* очень много опций: ffmpeg --help full, поэтому задавать специфические характеристики видео достаточно сложно. Для упрощения кодирования видео есть несколько графических оболочек для *ffmpeg*: handbrake, ffmpeg_batch, clever-FFmpeg-GUI. Они могут ровно то, что может *ffmpeg*, а именно: переконвертация видео в другой кодек, другое разрешение, кадрирование, обрезка по времени, но в графическом интерфейсе. Эти приложения могут быть установлены с помощью *Scoop* или *Winget*.

По ссылке приведён итоговый скрипт для создания видео из набора *png* картинок.

Отступление: **Создание видео в древнем avi формате**

Это устаревший способ кодирования видео, и он вам не потребуется; нужен только на старых серверах.

При таком варианте кодирования размер изображений должен быть кратным 16, этого можно добиться, либо округлив размер до 16 и перекодировать в него, либо сразу перекодировать в разрешение 1920x1072.

Вычислить размер изображения кратный 16 пикселям в большую сторону:

```
magick convert 0000.png  -format '%[fx:16*int((w+15)/16)]x%[fx:16*int((h+15)/16)]' info:
```

Отмасштабировать в размер 1920x1072 с сохранением пропорций, недостающие поля нарастить белым цветом от центра (для прозрачности использовать *none*), указать тип *png* палитры цветов Truecolor без прозрачности (для палитры с прозрачностью задать *6*):

```
mogrify -resize 1920x1072 -background white -gravity Center -extent 1920x1072  +repage -define 'png:color-type=2' *.png
```

И создать древний avi:

```
png2yuv -f 20 -Ip -j '%04d.png' | yuv2lav -b 2048 -o export.avi
```

Который можно перекодировать в видео с кодеком поновее, в *mpeg4* или *H.264*:

```
HandBrakeCLI --preset "Production Standard" --encoder mpeg4 -i export.avi -o export.mp4
```

```
HandBrakeCLI --preset "Production Standard" --encoder x264 -i export.avi -o export.mp4
```

Этап 3: Сложные нужные программы

6 Инженерные программы

6.1 OpenFOAM

OpenFOAM

Онлайн книжка по OpenFOAM — Notes on Computational Fluid Dynamics: General Principles (2022) C. Greenshields, H. Weller.

OpenFOAM будет запускаться из docker-контейнера из под WSL. В такой комбинации: OpenFOAM сможет работать в параллельном режиме, будет запускаться ParaView без X-сервера в нативном WSL2 графическом режиме.

Дальше предполагается, что *DockerDesktop* уже запущен.

Устанавливается с помощью docker из каталога openfoam, например[10]:

```
docker pull openfoam/openfoam11-paraview510:latest
```

Установка: в WSL скачивается запускающий *OpenFOAM* скрипт из официального каталога http://dl.openfoam.org/docker/ и настраивается; для этого выполнить следующие команды в WSL терминале (открывается командой wsl, или в профиле WSL в WT/Conemu/Cmder):

```
mkdir -p $HOME/software
cd $HOME/software
wget http://dl.openfoam.org/docker/openfoam11-linux
sudo mkdir -p /usr/local/bin
sudo cp openfoam11-linux /usr/local/bin
sudo chmod a+x /usr/local/bin/openfoam11-linux
exit
```

Запуск OpenFOAM: выполнить из любого терминала:

```
cd some-project-dir
wsl openfoam11-linux -x
```

Note: если изначально OpenFOAM не был закачен с помощью *docker pull ...*, то OpenFOAM будет загружен при первом запуске, но при этом не будет прогресс-бара.

Note: скрипт запуска *openfoam11-linux* имеет небольшой баг: после его запуска и выхода, запуск в этом же терминале wsl (без параметров) будет произведён в последней mount-point OpenFOAM контейнера; это абсолютно ни на что не влияет, просто к сведению.

[10] Скачивает 1.5 ГБ, займёт 3 ГБ

6.2 ParaView

Нативный **ParaView**, который умеет применять аппаратное ускорение видеоадаптера (OpenGL), должен быть побыстрее, чем встроенный в *OpenFOAM*. Установить (долго скачивает!):

```
winget install --id=Kitware.ParaView -e
```

6.3 FreeCAD

FreeCAD — opensource CAD программа. Использование разделяется на режимы, в зависимости от выбранного Workbench: есть режимы создания 2D чертежа (Draft&Sketcher), два режима создания 3D деталей (Part&PartDesign), сборки геометрии (Assembly), создания сетки с помощью GMSH (Mesh), создание и решения задач МКЭ (FEM). Все эти возможности показаны в соответствующих туториалах.

Установка:

```
winget install --id=FreeCAD.FreeCAD -e
```

Настройка:

- режиму навигации мышкой продуктов Ansys соответствует режим *Blender* во FreeCAD
- цветовая схема *FreeCAD Light* самая разборчивая.

6.4 Blender

<TBD>

```
winget install --id=BlenderFoundation.Blender -e
```

6.5 3D reverse engineering

<TBD>

https://alternativeto.net/software/geomagic-design-x/

CloudCompare, about.

MeshLab — https://www.meshlab.net/, about. The open source system for processing and editing 3D triangular meshes.
It provides a set of tools for editing, cleaning, healing, inspecting, rendering, texturing and converting

meshes. It offers features for processing raw data produced by 3D digitization tools/devices and for preparing models for 3D printing.

See also https://github.com/topics/3d-reconstruction

7 Математика

7.1 Калькулятор!

Qalculate! — калькулятор с широчайшими возможностями: единицы измерения, интервальная арифметика, полиномы, решение уравнений, простое дифференцирование и интегрирование, матрицы и вектора! Примеры вычислений.

Установка:

```
winget install --id=Qalculate.Qalculate -e`.
```

7.2 Matlab аналоги

7.2.1 Scilab

Scilab — открытый программный комплекс, полноценная замена Matlab. Есть встроенный аналог Simulink — Xcos. Установка:

```
winget install --id=Scilab.Scilab -e
```

7.2.2 Octave

Octave — простая среда для разработки и отладки Matlab программ. Simulink не поддерживается.

```
scoop install octave
```

7.3 SageMath

SageMath — opensource программа для символьной математики, альтернатива Mathcad, Mathematica.

Устанавливается с помощью docker из репозитория https://hub.docker.com/r/sagemath/sagemath[11]:

```
docker run --name sage -p8888:8888 sagemath/sagemath:latest sage-jupyter
```

В консоли появится сообщение вида

```
To access the server, open this file in a browser:
    ...
  Or copy and paste one of these URLs:
    http://127.0.0.1:8888/tree?token=
```

По этой ссылке будет доступно запущенное SageMath приложение. В нём создать новую сессию: кнопка *New* справа сверху, из выпадающего списка выбрать SageMath. Созданный Jupyter NoteBook после работы сохранить в виде .ipynb файла, и скачать (*Download*) в файловом менеджере на первой странице, чтобы сохранить копию в своих папках, помимо сохранённой версии в docker контейнере.

В последующем, для запуска *SageMath* набирать столь сложную команду не потребуется: контейнер можно будет запускать либо из интерфейса *DockerDesktop*, либо из командной строки командой `docker start sage` (см. *Управление Docker*).

7.4 Maxima

Ещё есть *CAS (Computer Algebra Systems)* программа для символьной математики **Maxima**/**WxMaxima**, использование которой достаточно не тривиально, но это, возможно, наилучшая открытая *CAS*.

```
scoop install extras/maxima
```

Дополнительно, можно установить чуть более новый GUI со страницы wxMaxima-developers.

7.5 Построение графиков

7.5.1 LabPlot

labplot.kde.org
Открытая программа для построения графиков по табличным данным, альтернатива

[11] размер скачиваемого образа 1.5 Гбайт

Origin/Grapher. Понимает Markdown и LaTeX формулы; умеет загружать данные напрямую из Jupyter Notebook, включая Python, Julia, Maxima. Умеет *Digitization* картинок — вытаскивать данные из графиков с картинок.

```
winget install --id=KDE.LabPlot -e
```

7.5.2 Veusz

veusz.github.io
Простая программа для построения графиков по табличным данным.

```
scoop install extras/veusz
```

7.5.3 Matplotlib

matplotlib.org
Библиотека для Python и C++ для построения графиков, удобна для постоянных пользователей Python; установка:

```
python -m pip install -U matplotlib
```

Пример использования библиотеки *matplotlib* из C++ (отсюда):

```
// (c) 2025 https://discourse.julialang.org/u/Freya_the_Goddess

// g++ main.cpp -o main -std=c++11 -I/usr/include/python3.9 -lpython3.9 -DWITHOUT_NUMPY
// g++ main.cpp -o main -std=c++11 -lpython3.9 -I/usr/include/python3.9 -I/usr/lib/python3.9/site-packages/numpy/core/inc

#include "matplotlibcpp.h"
#include <cmath>

namespace plt = matplotlibcpp;

int main()
{
   // Prepare data
   int n = 1000;
   double xl = 0.0, xh = 10.0;
   std::vector<double> x(n), y(n), z(n);

   for(int i=0; i<=n; ++i)
   {
```

```
        x.at(i) = xl + i*(xh-xl)/n;
        y.at(i) = pow(1.0 / pow(x.at(i) - 1.0, 2.0), 1.0/3.0);
    }

    // Set the size of output image to 1200x780 pixels
    plt::figure_size(1200, 780);
    // Plot line from given x and y data. Color is selected automatically.
    plt::plot(x, y);
    // Plot a line whose name will show up as "x exp(-x)" in the legend
    plt::named_plot("1/(x-1)^(2/3)", x, y);
    // Set x-axis to interval [0,10]
    plt::xlim(xl, xh);
    // Add graph title
    plt::title("Plot of 1/(x-1)^(2/3)");
    // Enable legend.
    plt::legend();
    plt::show();
    // Save the image (file format is determined by the extension)
    // plt::save("basic.png");
}
```

Тоже самое на Python:

```
import numpy as np
import matplotlib.pyplot as plt

n = 1000
xl = 0.0; xh = 10.0
x = np.linspace(xl, xh, n)
#y = np.power(x - 1, -2 / 3)
y = np.power(1 / (x - 1)**2, 1/3)

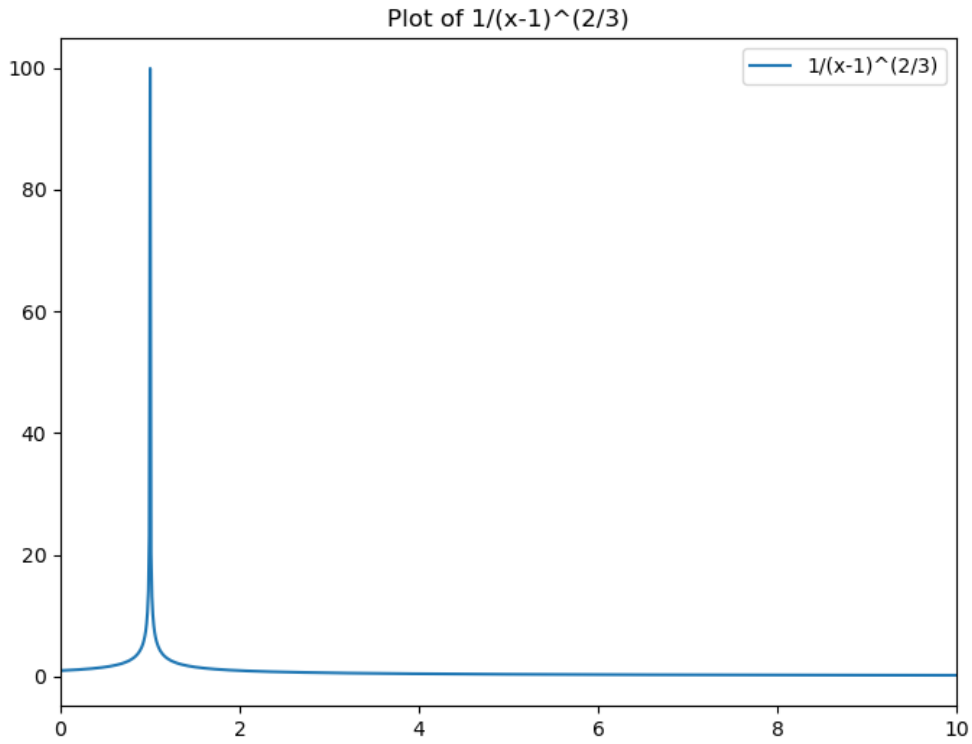
# plot 12 by 7.8 inches size at default 100 DPI gives 1200 by 780 picture
# but for Quarto inline-pic there is should be real size:
#plt.figure(figsize=(12, 7.8))
plt.figure(figsize=(8, 6))

plt.plot(x, y, label="1/(x-1)^(2/3)")

# Set x-axis to interval [0,10]
plt.xlim(xl, xh)

plt.title("Plot of 1/(x-1)^(2/3)")
```

```
plt.legend()
#plt.show()
plt.savefig("basic-python.png")
```



7.5.4 Julia

Для построения графиков на Julia существует множество пакетов, наиболее распространённый из них это Plots.jl. Документацию по параметрам отображения графиков можно найти на страницах документации: общие атрибуты, оси, параметры изображения и другие. Далее представлен код для построения графика аналогичного вышеприведённым.

```
```{julia}
using LaTeXStrings
using Plots
import Plots:mm

n = 1000
xl = 0.0
```
```

```
xh = 10.0

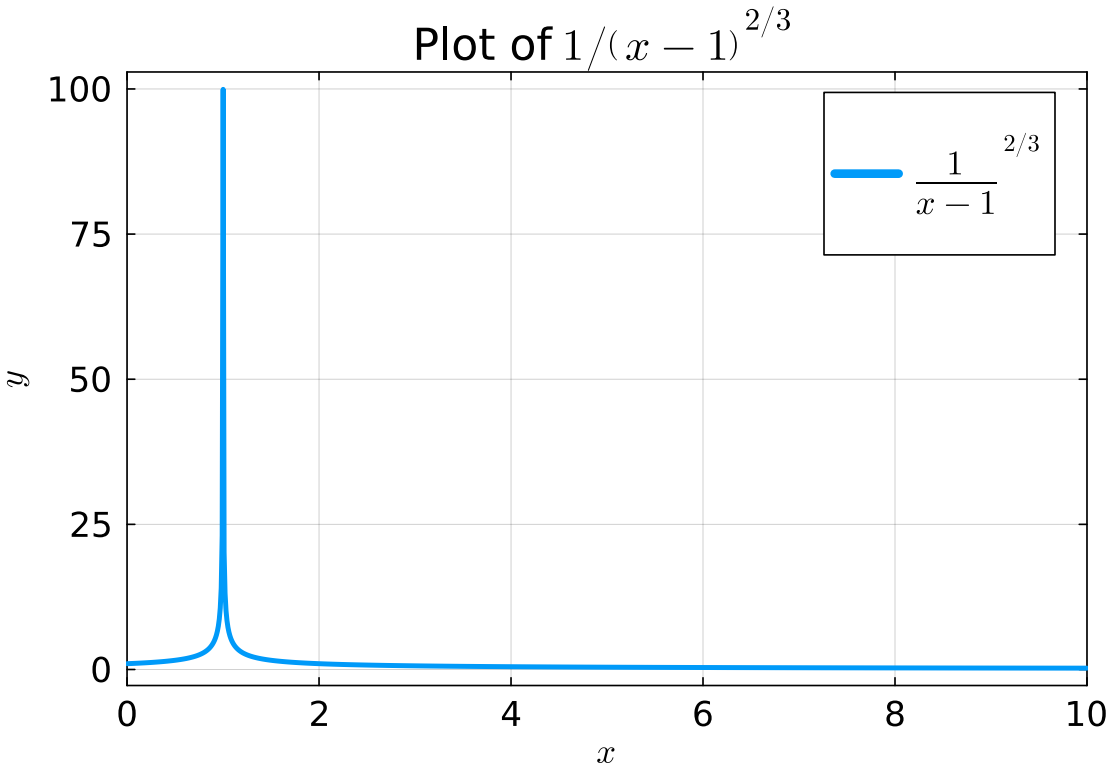
x = LinRange(xl, xh, n)

y = (1 ./ (x .- 1).^2).^(1/3)

# explicit init GR backend of Plots.jl
gr(size=(700,500))

plot(x, y,
    label=L"$\frac{1}{x-1}^{2/3}$",
    title=L"Plot of $1/\left(x-1\right)^{2/3}$", # also can be `:none`
    xlims=(xl, xh),
    xlabel=L"$x$",
    ylabel=L"$y$",
    legend=:topright,  # by default `:best`
    #size=(700,500),   # picture size in pixels (width x height)
    #size=(1200,780),   # picture size in pixels (width x height)
    #dpi=96,          # picture DPI
    #margin=10mm,
    left_margin=8mm,
    bottom_margin=8mm,
    linewidth=3,
    #markersize=8,
    #markerstrokewidth=2,
    legendfontsize=14,
    guidefontsize=14,
    titlefontsize=18,
    tickfontsize=14,
    #grid=true,           # enable grid
    #gridalpha=0.5,        # grid transparency
    #gridlinewidth=1.5,     # grid thickness
    #minorgrid=true,       # enable secondary grid
    #minorgridalpha=0.3,    # secondary grid transparency
    #minorgridlinewidth=1.0, # secondary grid thickness
    framestyle=:box
)

#savefig("basic-julia.png")
```



7.5.5 Gnuplot

www.gnuplot.info
Консольная программа для построения графиков с широкими возможностями. Примеры, обзор на русском.

```
scoop install gnuplot
```

Пример построения изображения графика на gnuplot. Этот фрагмент кода необходимо сохранить в файл, например *sample-plot.gp*, и выполнить команду gnuplot sample-plot.gp.

```
# set the image size (in pixels)
set terminal pngcairo size 1200,780 enhanced font 'Verdana,18'

# save the plot to a file
set output 'basic.png'

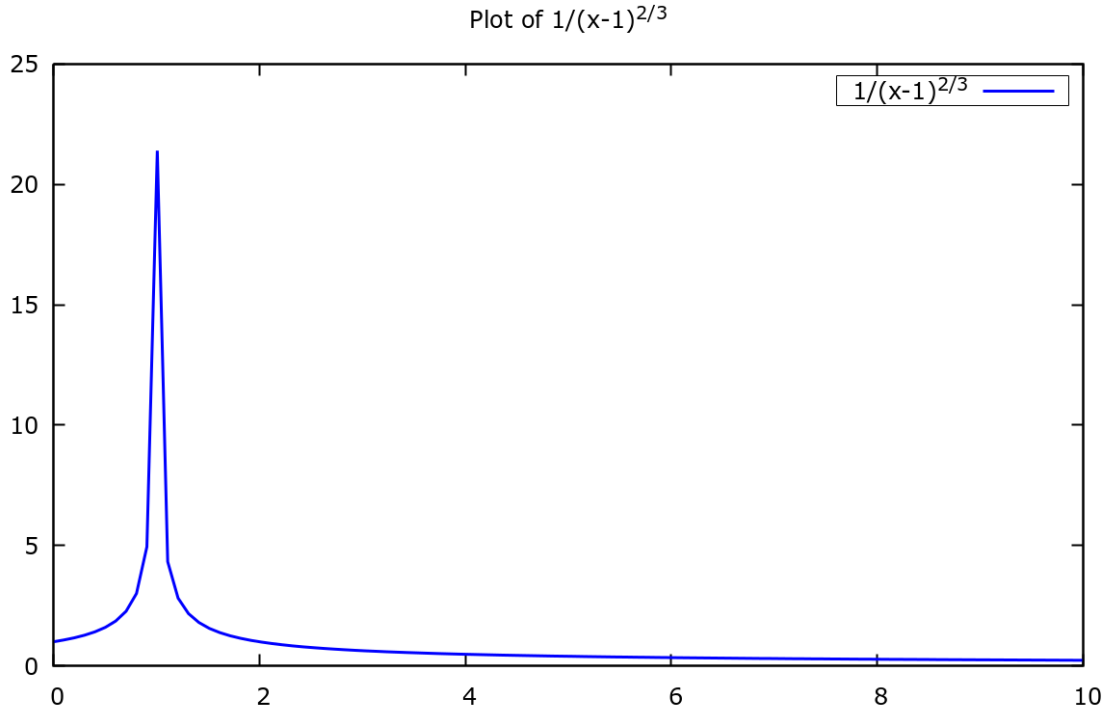
# set the plot title
set title "1/(x-1)^{2/3}"
```

```
# set axis ranges
set xrange [0:10]  # Set x-axis range from 0 to 10
set yrange [*:*]   # Automatically determine y-axis range. The `[*:*]` is the default behavior.

# enable the legend
set key box

set border linewidth 2

# plot the function
plot [0:10] (1.0/(x - 1.0)**2)**(1.0/3.0) title "1/(x-1)^{2/3}" with lines linewidth 3 linecolor rgb "blue"
```



Более сложный пример построения графика в gnuplot с обработкой данных из файла и построением двойной оси *Y*:

```
scale = 2.0

set terminal pngcairo size 640*scale,480*scale fontscale scale linewidth scale pointscale scale
set output 'report-water-height.out.png'
set nokey

# setting up graph axes properties
set border 1+2+4+8 lw 4
```

```
set xtics border mirror in scale 1, 0
set ytics border mirror in scale 1, 0
set mxtics  5
set mytics  5

# setting up graph grid properties
set grid xtics ytics  mxtics mytics lt 1 lw 1 lc rgb "black", lt 1 lw 0 lc rgb "black"

# some specific function for data processing
fun1(x) = 0.00142635 + 0.698578*x - 0.913872*x**2 + 0.75045*x**3 - 0.302874*x**4 + 0.0593173*x**5 - 0.00438143*x**6

# set up second Y-axis with values `40 - i`, where `i` is corresponds the first Y-axis
do for [i=-100:100:2] { set y2tics add (sprintf("%g",i) 40.0-i) }

# `$3` and `$2` tells it to use the data from the third and second columns of data-file
# `every ::3` specifies to plot the graph skipping the first three rows of data
plot [:][:] 'report-water-mass.out' using ($3):( (1.01 - fun1($2/1000.0)) / 0.0254 ) every ::3 with lines lw 3 lc rgb "blue"
```

Файл с данными *report-water-mass.out* имеет следующий вид

```
"report-def-0-rfile"
"Time Step" "report-def-0 etc.."
("Time Step" "report-def-0" "flow-time")
0 1907.704977059886 0
10 1703.23047212398 0.5
20 1752.389260134199 1
30 2163.713605058183 1.5
40 2384.578894811974 2
50 2556.318405498912 2.5
60 2655.06514863053 3
70 2722.175536462764 3.5
80 2743.568186672038 4
90 2716.455743643943 4.5
100 2613.660462368277 5
110 2527.183466538219 5.5
120 2466.08010246078 6
130 2360.274092810494 6.5
140 2315.101811862138 7
150 2430.425040697297 7.5
160 2581.324755366251 8
170 2677.462160156302 8.5
180 2788.573315838712 9
190 2842.559141891374 9.5
200 2825.030621610734 10.00000000000001
```

```
210 2672.220368620244 10.5
220 2468.360142802453 11
```

Этап 4: Сложные ненужные программы

8 *Приложение:* AI/LLM

Note: В этой главе в основном говорится про работу с локальными моделями *LLM* — которые хранятся и запускаются непосредственно на компьютере. Но, все приведённые принципы работы верны и для онлайн моделей.

8.1 Введение

Большие языковые модели (Large Language Models, LLM), это искусственные нейронные сети (НС) типа трансформер, обученные на больших объёмах данных. LLM понимают естественные языки, и генерируют ответы на них; содержат в себе большой объём связной информации, на основе которой строят ответы; имеют способности к размышлению в виде построения логических цепочек, в том числе дополнения, интерполяции, ассоциации последовательностей слов; умеют работать с текстами: переводить, анализировать на связность, непротиворечивость, рецензировать, и, создавать новые тексты по запросу. Помимо работы с естественными языками работают с языками программирования и другими искусственными языками: программируют, ищут баги, создают коды для станков с ЧПУ. Специально обученные сети работают с графической информацией: распознавание изображений и видео, OCR, генерация и/или модификация изображения и видео. Аналогично, работают и с аудио-информацией, в частности распознавание и генерация речи, генерация музыки. Могут выступать в качестве управляющего контура для роботов и станков, и в качестве автопилотов.

Несколько упрощённо работу нейросети-трансформера можно описать как манипуляции над цепочкой точек (эмбеддингов[12]) в многомерном[13] семантическом пространстве[14]. В таком семантическом пространстве близкие понятия находятся на небольшом расстоянии друг от друга; в этом пространстве работают *разностные отношения* (*difference relations*): “Король - Мужчина = Королева”. Поэтому можно утверждать что, языковые модели в самом деле “понимают” информацию с которой работают; что “умеют мыслить”, выделяя главное, находя ассоциации, *разностные отношения*, и проч.; умеют находить решения, заполняя лакуны в цепочках эмбеддингов интерполяцией в семантическом пространстве, продляя цепочки ембеддингов экстраполяцией.

Из характерных особенностей нейронных сетей типа трансформер можно выделить:

- входной текст разбивается на токены — это минимальные единицы информации для нейросети. Каждому знаку препинания соответствует один токен. Каждому короткому, или общеупотребительному слову соответствует один токен. Сложным или редким словам соответствуют 2, 3 или более, токенов, как правило, по одному токену на корень

[12] см. описание алгоритма в конце главы

[13] типичная размерность 512

[14] Каждая точка-эмбеддинг представляет собой короткий вектор из floating-point величин, и над последовательностью таких векторов проводятся алгебраические операции.

слова, на приставку, суффикс, окончание, что очень характерно для русского языка, в английском большинство слов это один токен. На выходе из трансформера токены обратно превращаются в текст;

- в процессе работы, на каждой итерации, на выходе из сети в качестве ответа генерируется один токен, который присоединяется в конец цепочки, состоящей, из исходного запроса, и последовательности уже присоединённых в конец токенов ответа, полученных на предыдущих итерациях; после чего пополненная цепочка токенов опять посылается на вход и начинается новая итерация;
- типичная максимальная длина входного контекста — тысячи и десятки тысяч токенов; длина выходного контекста сравнима с размером входа; но бывает и на порядки больше длины входа у специализированных трансформеров;
- сеть-трансформер обычно состоит из 10-40 двойных слоёв;
- несколько приближённо, но можно сказать, что каждый слой НС представляет собой матрицу[15], и прохождение информации через слой является операцией умножения этой матрицы на каждый вектор (ембеддинг) из набора, представляющего запрос плюс текущую часть ответа;
- отлично распараллеливается и обучение, и выполнение;

Упрощённое описание алгоритма работы нейросети-трансформера приведено в конце главы.

Где начать

По степени доступности языковые модели LLM можно разделить на две категории:

- закрытые модели, которые недоступны для загрузки и локального использования;
- модели с открытыми весами, которые предоставляют возможность загрузки и запуска на локальных вычислительных ресурсах. Большинство из открытых моделей доступны к загрузке с сайтов https://huggingface.co/ и https://ollama.com/.

И закрытые модели, и модели с открытыми весами, одинаково доступны в качестве онлайн-сервисов. Среди них есть сервисы с бесплатным доступом, есть с платным, и, большинство онлайн-сервисов имеют гибридный тип доступа.

Наиболее популярные бесплатные онлайн-сервисы LLM, доступные из России:

- GigaChat https://giga.chat/ от Сбера;
- YandexGPT https://ya.ru/ai/gpt от Яндекса;
- DeepSeek https://chat.deepseek.com/ от DeepSeek (из Китая, как и последующие);
- Qwen https://chat.qwen.ai/ от Alibaba;
- Kimi K2 https://www.kimi.com/ от Moonshot AI;
- GLM https://chat.z.ai/ от Zhipu AI.

[15] подробнее в описании алгоритма

Все эти онлайн-сервисы бесплатны только в конечном количестве запросов (в сутки), при превышении которого потребуется оплата. Доступ по API для ИИ ассистентов, агентов, RAG, и пр. доступен только за плату.

Note: Конечно, возникает естественный вопрос, а зачем тогда платить, если есть возможность скачать и запустить модели самостоятельно? Причина в том, что объём современных сильных моделей составляет сотни гигабайт[16], и никакой персональный компьютер их не сможет нормально запустить; причём, покупка и эксплуатация необходимой для LLM вычислительной инфраструктуры выйдет много дороже, чем использование платных онлайн LLM сервисов.

8.2 PROMPTS

На данный момент основной формой работы с LLM является диалог (чат). Другие варианты взаимодействия, такие как ИИ-помощники для программирования и написания текстов, агенты, являются слишком сложными в настройке программного обеспечения, имеют санкционные ограничения с доступом, и, наконец, все эти возможности требуют, либо оплаты онлайн сервисов LLM, либо больших вычислительных ресурсов для своего использования.

При работе в диалоге с LLM на первый план выходят вопросы (запросы, prompts) к LLM: в зависимости от того, насколько точно человек сможет сформулировать описание задачи и задать вопрос, настолько адекватно и точно модель будет отвечать. Модель может несколько "догадываться" о контексте, о неявной части вопроса, но не стоит на это рассчитывать, и, следует *явно* и *точно* формулировать запросы.

8.2.1 Примеры запросов к ЯМ/LLM

Для выполнения задач надо прямо в лоб на русском спрашивать:

> Напиши программу для параллельного умножения матрицы размером MxN на вектор с использованием MPI на языке Fortran. Последовательно объясни действия процедур.

> Напиши программу для интегрирования задачи трёх тел на примере Солнца, Юпитера и Земли методом Адамса-Башфорта на Python. Последовательно объясни действия функций.

> Как на github создать PR в чужой проект, если у меня уже есть форк этого проекта со внесёнными мной коммитами и мержами апстрима?

> Как настроить VSCode для компиляции и отладки программ на языке Fortran?

[16] DeepSeek — 1.4ТБ, Qwen3 — 480ГБ

> Аккуратно сконвертируй информацию с приложенного pdf
> в размеченный Markdown код, **сохраняй форматирование**.
>
> @article-with-formulas.pdf

> Переведи на английский язык текст начиная со следующего абзаца,
> выведи только перевод, сохраняй форматирование.
>
> Ехали медведи\
> На велосипеде.\
> А за ними кот\
> Задом наперёд.\
> А за ним комарики\
> На воздушном шарике.

> Тщательно переведи на русский язык **каждую строчку** прикреплённого справочника.
>
> @Reference.md

> Оформи в виде Markdown таблиц прикреплённый справочник.
>
> @Справочник.md

> Скомбинируй два прикреплённых справочника в виде Markdown таблиц из трёх колонок,
> где в первой колонке будет общая первая колонка, во второй колонке будет описание
> на русском, и в третьей колонке будет описание на английском.
>
> @Справочник.md, @Reference.md

> Напиши скрипт на Julia для интегрирования массы, методом трапеций, по данным из файла:
>
> - исходные данные для интегрирования во второй колонке файла, поток массы, в [kg/s];
> - соответствующие временные метки в третьей колонке, в [s];
>
> Как результат работы, скрипт должен отправлять на консоль данные построчно.
> В каждой строке должны выводиться следующие величины, разделённые пробелами:
>
> 1. порядковый номер;
> 2. суммарная, накопленная к этому моменту масса, в [gramm];
> 3. момент времени, [s].
>
> Комментарии в скрипте должны быть на английском языке.

> Отдельно поясни работу скрипта на русском языке.

> Напиши скрипт, на Python с использованием PyTorch, для обучения нейросети MLP (Feedforward) для аппроксимации (регрессии) функции одного double параметра от нескольких входных double аргументов.
>
> - Задаваемые параметры, это: Re, alpha, ta_u, ta_l, tb_u, tb_l, alpha_c, alpha_b.
>
> - Искомые (аппроксимируемые) параметры для регрессии, это: Cl, Cd, Cm, Cp.
>
> - Исходные данные были заранее вычислены и сохранены файле CSV (dataset_random-20250604-121627.csv.xz), с заголовком следующего вида: "Re,alpha,ta_u,ta_l,tb_u,tb_l,alpha_c,alpha_b,Cl,Cd,Cm,Cp"
>
> - В скрипте требуется явно разделить входную выборку из CSV на данные для обучения, и данные для проверки.
>
> Искомые параметры должны обрабатываться по отдельности --- в скрипте должна производиться работа только с одной величиной (скажем Cl), и в результате должна получаться одна MLP сеть. Для получения других MLP сетей от других величин я, в дальнейшем, самостоятельно заменю все вхождения Cl в скрипте на другие величины.
>
> В скрипте необходима процедура сохранения полученной MLP в общеупотребительный формат хранения нейросетей PyTorch для дальнейшей загрузки и использования.

> Добавь в приведённый код ускорение обучения и применения на GPU Nvidia.

Для улучшения качества ответов связанных с математикой, к запросу следует добавлять фразу вида:

> Please reason step by step, and put your final answer within \boxed{}.

> Пожалуйста, рассуждай шаг за шагом, и помести окончательный ответ в \boxed{}.

8.2.2 Структуризация запросов и Markdown

В вышеприведённых примерах используется Markdown разметка ** для выделения текста, используется обратный слэш (\) в конце строк, который в Markdown обозначает перенос строки без начала нового параграфа. Также можно использовать бэктики ` и ``` для выделения участков кода.

При необходимости, запросы можно записывать в одну строку, в нужных местах вставляя символы \n для переноса строки для форматирования; это может потребоваться при работе через командную строку (CLI) или программные интерфейсы (API), когда модели передаётся запрос в виде одной строки текста.

Языковые модели отлично понимают разметку Markdown, поэтому, такую разметку рекомендуется всегда использовать, для усиления акцентов, для форматирования текста с целью лучшей структуризации информации.

Также для усиления акцентов могут использоваться формулы (слова) вежливости, такие как "Пожалуйста", на которые языковые модели превосходно реагируют.

Даже для несложных запросов ответы моделей значительно улучшаются, если запрос должным образом структурирован (см. выше примеры интегрирования методом трапеций и PyTorch). Ещё немаловажный плюс от структурирования запросов в том, что после написания такого запроса у пользователя остаётся хорошо структурированная формулировка решаемой задачи.

За один запрос модель вряд ли сможет выполнить сложную задачу, поэтому такие задачи необходимо разделять на этапы, как в примере со справочником; либо составлять сложные, состоящие из многих пунктов, запросы, с примерами, с критериями, и т.п.; см. далее Инжиниринг запросов.

8.2.3 Задание и отображение математических выражений

Все языковые модели при работе с запросами и при выдаче ответов используют Markdown в качестве языка разметки. В Markdown все математические выражения (формулы) записываются с использованием нотаций языка разметки *LaTeX*. Для выделения в тексте формул *LaTeX* используются два типа нотаций. Современная нотация выделения формул использует символы \(| \) для встроенных (inline math) формул, и \[| \] для блочных (display math) формул; и устаревшая нотация, введённая ещё Д.Кнутом, использует символы $ | $ для встроенных формул, и $$ | $$ для блочных, т.е. у устаревшей нотации нет разделения на открывающие и закрывающие выделители формул, что нехорошо в контексте длинных или неточных текстов, где в случае потери только лишь одного символа весь последующий текст становится нечитабельным.

Все онлайн-сервисы, а также большинство программ, используют надёжную современную нотацию выделения формул. Но, в запросах, также, понимают и устаревшую нотацию, которую несколько проще набирать для простых математических выражений.

8.2.4 Инжиниринг запросов

Для изучения методов составления *экспертных запросов* можно порекомендовать руководство от Google: *"Prompt Engineering"*.

Или в русском переводе, *"Руководство Google по промпт-инжинирингу"*, по разделам:

1. Основы промпт-инжиниринга и базовые техники
2. Продвинутые техники промптинга и работа с кодом

3. Лучшие практики и рекомендации

В приведённом руководстве достаточно подробно рассматриваются большинство аспектов языковых моделей с точки зрения экспертного использования: управляющие опции LLM, составление запросов, цепочки рассуждений, и многое другое.

Изучить данное руководство, или подобное ему, весьма полезно для понимания механизмов работы LLM. Правда, через пару-тройку лет подобные руководства станут неактуальными, так как языковые модели станут достаточно “умными” чтобы понимать людей и без специальных “приёмов” формирования запросов, а для сложных задач будет достаточно подробного *технического задания* (*ТЗ*).

В качестве примера инжиниринга запросов можно рассмотреть ситуацию, когда взаимодействие с LLM заходит в тупик и нужного результата не получается добиться даже с большим количеством уточняющих запросов. В таком случае можно прибегнуть к методу, когда сама модель будет помогать составлять новый запрос анализируя историю чата; для этого ей потребуется запрос вида (цитируется с https://habr.com/ru/articles/914640/):

```
# Роль
Ты - экспертный аналитик промптов, специализирующийся на работе с <...>.

Твоя задача --- провести глубокий анализ всей истории диалога и создать
оптимизированный промпт для получения корректного результата с первой попытки.

# Задача
Проанализируй весь контекст нашего чата от начала до конца, включая:

- Все мои первоначальные запросы
- Все твои ответы и попытки выполнения задач
- Все мои исправления, комментарии и указания на ошибки
- Паттерны проблем, которые повторялись

## Особое внимание удели

- Проблемам с <...>
- Ошибкам в анализе <...>
- Недопониманию контекста или требований
- Техническим ограничениям, которые не были учтены

# Инструкции по анализу
1. Определи корневые причины каждой ошибки или неточности
2. Выяви, какая информация была упущена в исходных промптах
3. Найди паттерны в моих исправлениях - что я систематически добавляю или корректирую
4. Оцени, какие дополнительные контекстные данные нужны для точного выполнения

# Формат результата
```

```
Создай новый, улучшенный промпт в следующем формате:


**ОПТИМИЗИРОВАННЫЙ ПРОМПТ:**

[Здесь полный текст нового промпта, готовый к копированию]


**КЛЮЧЕВЫЕ УЛУЧШЕНИЯ:**

- [Перечисли 3-5 основных изменений по сравнению с предыдущими попытками]


**ПРЕДОТВРАЩЁННЫЕ ОШИБКИ:**

- [Укажи конкретные проблемы из истории чата, которые теперь должны быть решены]

# Стиль результата
- Промпт должен быть максимально конкретным и однозначным
- Включи все необходимые технические детали и ограничения
- Предусмотри возможные краевые случаи на основе истории ошибок
- Используй чёткие, недвусмысленные формулировки
```

В запросе необходимо подставить свои формулировки в места <...>.

8.2.5 *Отступление* об особенностях запросов для локальных LLM

В LLM предыдущих поколений качество запроса было критически важно для успешности решения задач. Современные LLM, обученные методом *Reinforcement Learning*, уже не столь сильно зависят от ясности, явности, и однозначности запросов. И тем не менее, все онлайн-сервисы языковых моделей при работе автоматически добавляют к пользовательскому запросу так называемый *системный запрос* (*System Prompt*), который улучшает и структурирует "мыслительную цепочку" языковой модели.

Некоторые коммерческие сервисы открывают свои *системные запросы*; они были собраны, и представлены в библиотеке https://github.com/elder-plinius/CL4R1T4S. По текстам системных запросов видно, что, во-первых, они очень объёмные — десятки килобайт, это многие страницы текста; а во-вторых, что достаточно значительная часть из этих десятков килобайт запроса состоит именно в попытке улучшить структуру мышления модели; остальная часть системного запроса задаёт ограничения на выполнение для безопасности ответов.

При локальном применении LLM некий *системный запрос* тоже добавляется, но он совершенно не столь широк и универсален как системные запросы коммерческих систем, и, часто

представляет собой буквально одно предложение вида *“You are AI assistant.”*. Посмотреть применяемый в локальной LLM *системный запрос* можно, либо на сайте *hf.co*, либо в применяемой программе.

Поэтому, если хочется, чтобы ответы от локальной LLM были столь же качественными, как и ответы от коммерческих сервисов, надо не лениться, и писать хорошие запросы (см. п. Инжиниринг запросов), а ещё лучше, добавлять в запрос “шапку”, которая будет объяснять модели *как ей следует думать*. Примеры хорошо структурированных запросов (prompts) можно посмотреть в библиотеках запросов, например https://github.com/0xeb/TheBigPromptLibrary или https://github.com/abilzerian/LLM-Prompt-Library.

8.2.6 *Отступление* о потенциальных направлениях улучшения запросов к большим языковым моделям

В 2025г начали развиваться методологии формирования запросов с целью создания ~~Операционной Системы~~ Контекстно Ориентированных Систем на основе языковых моделей LLM. Данный подход предполагает создание запроса специального вида, в процессе выполнения которого образуется *цепочка размышлений*, формирующая цикл размышления в LLM модели, который, в свою очередь, образует *граф состояний* (*граф размышлений*), с возможностью возвращаться обратно по цепочкам графа, или двигаться по циклу графа размышлений вперёд с возможностью модификации состояний (текста/эмбеддингов). Данный подход позволяет с лёгкостью проводить самоинтроспекцию, самоанализ, итеративное самоулучшение ответов модели в виде модификации запроса и повторного выполнения обновлённого запроса в рамках цикла обработки запроса.

Такой подход позволяет формировать в *системе*, состоящей из языковой модели и запроса, некоторую *“среду размышлений”*, в рамках которой пользователь может углубляться в детали, возвращаться обратно на верхний уровень графа размышлений, что-то изучать подробнее, прояснять и так далее — иными словами, проводить работу в рамках заданной темы в контексте уже заданных вопросов, уже заданных ограничений, и уже полученных ответов.

Один из примеров создания подобных систем демонстрируется в публикации “Промт для изучения чего угодно: протокол Олега-Деминга”, а сами запросы представлены в двух вариантах: стандартный и въедливый. Как видно из текстов запросов это весьма сложные *коды*, которые содержат переменные, циклы, парсеры, возможность отладки, и другие техники программирования, которые никак не ожидаешь увидеть в запросах к языковым моделям. Чуть подробнее о некоторых особенностях представленной системы. Все вопросы и ответы языковая модель автоматически нумерует, чтобы к ним в дальнейшем можно обращаться по *индексу*. Система управляется командами *Do:*, *zoom*, *up*, *root*, *expand*, *iterate*, *advance*, *evolve*, а также *query_state* для отладки. Пользователь пишет в последующих запросах одну из вышеперечисленных команд, как правило ссылаясь на *индекс* предыдущих вопросов/ответов, возможно с некоторыми другими параметрами, и языковая модель отвечает на команду, учитывая, и текущее положение “точки фокуса” в *графе размышлений*, и состояние *графа размышлений* — его контекст.

Работу с такой системой, при наличии переходов по ссылкам-индексам, можно представить как взаимодействие с сайтом, например, с Википедией, но, в отличии от статического сайта,

при работе с системой *“новые страницы”* генерируются на лету с учётом предыдущих запросов и уже полученных ответов.

Легко представить, как в такой системе набором специальных вопросов можно сформировать *“среду”* для выполнения какой-то конкретной, специфической задачи, где всё будет заточено под необходимый контекст, и любые новые вопросы будут интерпретироваться исключительно в заданных рамках. В дальнейшем, выгружая и сохраняя сформированные цепочки запросов целиком, появляется возможность делать “сейвы” сформированной системы, получая на выходе интерактивные отчёты, руководства или книги, которые можно копировать, передавать, продавать.

Скорее всего в ближайшее время все коммерческие большие языковые модели перейдут на *Системные Запросы* (*System Prompt*) в какой-то похожей форме; и, даже вполне возможно, что кто-то уже сейчас применяет схожие технологии.

Note: Исходя из сложности упомянутых выше запросов, следует ожидать, что их смогут выполнить только достаточно большие Reasoning LLM, а доступные локальные языковые модели с запросами такой сложности не справятся.

8.3 Некоторая классификация открытых языковых моделей

Note: Здесь рассматриваются модели работающие с текстами. Модели работающие с мультимедиа информацией имеют схожие категории, но всё же у них есть своя специфика.

После первичного обучения, когда внутрь LLM поместили весь интернет, возникает, так называемая, *Base* модель. Эта модель не умеет делать чего-либо определённого, а может только отвечать наиболее вероятными ассоциациями на запросы.

Далее base модель дообучают для той или иной цели.

Для ведения диалогов base модель дообучают на массивных наборах вопрос-ответ, чтобы модель в дальнейшем, по этой аналогии, отвечала на новые вопросы. Такая модель для чата называется *Instruct* модель, так как она выполняет инструкции — отвечает на вопросы.

Другой вариант дообучения base моделей состоит в том, что модель учат составлять логические цепочки связанные с запросом. Обучают, опять же, на массивах логических цепочек, таких как математические доказательства, тексты по логике, и т.п. На заключительном этапе такого обучения у модели формируется специальная форма ответа, в процессе которого, модель сначала строит логические цепочки и ассоциации связанные с запросом, которые, возможно, потребуются в ответе; после чего эта информация включается в исходный запрос, и на основе этого пополненного запроса строится полный ответ. Это модели с мышлением, *Reasoning* модели. Простые instruct-модели без мышления называют *Non-Reasoning* моделями. При равных размерах, *Reasoning* модели всегда сильнее простых моделей, но и гораздо медленнее.

Отдельно можно выделить модели обученные и/или дообученные на больших массивах программного кода с упором на логичность, точность и педантичность до самого последнего символа, в ущерб “человечности” диалога и прочих социальных штучек. Такие модели называются *Coder*, *Coding* и т.п. Основная область применения этих моделей, это работа в составе

средств разработки и *Copilot*, для задач автокомплита, написания кода, тестирования, поиска ошибок в коде, написания документации по готовому коду; также могут применяться для поиска логических ошибок в текстах и формулах. *Coder* модели, это *Non-Reasoning* модели, заточенные, в первую очередь на точность и скорость; они, конечно, могут написать простую программу по запросу, но если потребуется создание чего-то со сложной математикой или алгоритмами, то coder модели однозначно уступят любой сравнимой по размеру *Reasoning* модели. (*Note: И*з-за того, что для продуктивной работы программистов требуется хорошая отзывчивость среды разработки, потребуется большая скорость выполнения LLM, для чего нужны очень мощные вычислительные ресурсы, которые недоступны большинству пользователей; поэтому, скорее всего *в качестве Copilot'а*, эти модели придётся использовать по платной подписке не смотря на их открытость.)

Ещё бывают математические модели, которых дообучали на логике и доказательствах теорем. Они предназначены для проверки уравнений и доказательств, и, для составления новых уравнений и доказательств — с проверками справляются, с составлением новых доказательств пока не очень. Называются *Math*, *Proof*, *Prove* и т.п. Можно сказать, что это *Reasoning* модели на максималках — они будут долго и скурпулёзно перебирать разные варианты и гипотезы, либо пока не найдётся верное решение, либо пока не будет достигнута предельная длина цепочки токенов.

Ассистенты, и, в частности, *Copilot'ы* (code assistant) — это основанные на LLM инструменты, предназначенные для предоставления подсказок, автодополнения текста или кода "на лету". Они могут использовать и *Reasoning*, и *Non-Reasoning* модели. Функционал *Ассистентов* основан на использовании специального *системного запроса*, и оптимизации под скорость работы, чтобы исключить большие задержки при работе с пользователем. Архитектурно это не отдельный класс моделей, а форма практического применения моделей.

Для полноты обзора: модели для визуального распознавания текста, OCR — это *VL*, *Vision*, *Multimodal*; *Embedding* модели превращают текст в вектор эмбеддингов (см. алгоритм работы LLM).

Резюмируя:

- модели *base* нужны *только* для исследователей и создателей LLM для дальнейшего дообучения для каких-то частных конкретных задач;
- *Instruct* модели умеют вести диалог, все онлайн-сервисы — это instruct модели;
- *Instruct* модели делятся на *Reasoning*, и *Non-Reasoning*. Первые лучше отвечают, но намного медленнее вторых;
- *Coder* модели для программирования, могут писать код или документацию к коду по запросу; могут применяться в составе сред разработки, в том числе как *Copilot*;
- *Math* модели вряд ли понадобятся;
- *Vision* (*VL*) вместе с *Embedding* используются как вспомогательные модели для OCR загружаемых документов;
- *Multimodal* модели — это универсальные модели со встроенным OCR, бывают и *Reasoning*, и *Non-Reasoning*;

- *Ассистенты*, и, в частности, *Copilot'ы* — это форма практического применения моделей, а не отдельный класс моделей.

8.4 Технологии *RL* и *MoE*

Прогресс в области LLM технологий происходит невероятными темпами, ещё год назад было невозможно себе представить, что на персональном компьютере можно запустить хоть что-то дающее хоть какой-то результат. Полгода назад запускаемые локально модели требовали слишком больших ресурсов для нормальных результатов. А уже сегодня на ноутбуке можно запускать модели, которые дают вполне нормальные результаты. Этого добились комбинацией двух подходов в обучении LLM: *RL* и *MoE*.

Новейшие *Instruct* модели (2025) дообучаются методом *Reinforcement Learning* (*RL*) — Обучение с подкреплением, в отличии от предыдущего поколения LLM, которые обучались методом *Reinforcement Learning from Human Feedback* (RLHF) — Обучение с подкреплением на основе отзывов. В методе *RL*, при обучении, помимо заранее заданных массивов вопрос-ответ, или массивов логических цепочек, модель сама в процессе обучения создаёт новые пары вопрос-ответ, новые логические цепочки, и проверяет их на корректность, либо самостоятельно, либо с помощью другой языковой модели; результаты этой проверки становятся ключом для метода подкрепления и дальнейшего закрепления требуемого поведения. Этот метод обучения приводит к некой *Самоинтроспекции* (*Self-Reasoning*) моделей, в результате чего внутренние данные моделей становятся лучше взаимосвязаны, кластеризованы, лучше ассоциированы; такие модели меньше ошибаются, лучше улавливают контекст запросов. Обученные методом *RL* модели могут быть как *Reasoning*, так и *Non-Reasoning*.

Другой подход обучения LLM позволяет создавать модели *"Mixture of Experts"* (*MoE*), или *"модели экспертов"* (*"комбинация экспертов"*), в которых, при выполнении, в каждый момент времени задействуется только небольшая часть (типично 5–10 %) весов, непосредственно ответственная за выполнение запросов текущего типа. Такое поведение обеспечено кластеризацией весов со схожими аспектами задач, областями ответственности. *MoE* модели можно представить как набор небольших специализированных моделей в общей упаковке, с маршрутизатором отправляющим данные тому или иному "эксперту" в зависимости от содержимого данных. Модели *MoE* имеют преимущество перед обычными "плотными" моделями в том, что требуют на порядок меньшего количества вычислительных затрат на выполнение при равных размерах. Из недостатков следует отметить, что общая точность несколько падает по сравнению с "плотными" моделями того же размера.

Комбинация методов *обучения с подкреплением* (*RL*) и *моделей экспертов* (*MoE*) позволяет создавать модели, обладающие высокой скоростью работы и точностью, а также обеспечивающие заданные стратегии поведения системы. В качестве примера подобных современных моделей можно привести открытые *MoE* модели семейства *Qwen3*.

8.5 Размеры и квантизация моделей

В качестве одной из базовых характеристик языковых моделей можно отметить их размер — пишется как 3B, 70B и т.п., обозначает количество коэффициентов-весов в модели в B-billions — в миллиардах. Например, у не квантизированной (см. далее) 3B модели с размером весов *FP16* (2 байта), размер всей модели будет 3E9 * 2 байт = 6 Гбайт.

Модели бывают исходные, где все веса имеют размер использовавшийся при обучении модели, *FP16* или даже *FP32.* И бывают квантизированные модели, веса у которых были округлены до меньшего количества занимаемых бит для экономии занимаемой памяти и ускорения работы. Уровней квантизации-округления много разных, от 8 бит на коэффициент (*Q8_0*), до 1-2 битов на коэффициент (*Q2_K* и т.п.). Квантизированные модели всегда менее точные, чем исходные, поэтому, при возможности, следует использовать неквантизированные исходные *FP16* модели. Характерная зависимость точности (perplexity) моделей от степени квантизации представлена на графике Perplexity vs. Quantization. Из графика следует, что *Q8_0* практически неотличима от *FP16*; но чем больше модель ~~огрублена~~ округлена, тем хуже её качество. Также из графика следует, что даже самая сжатая модель большего размера будет лучше, чем не квантизированная модель меньшего размера.

Разные коэффициенты/веса моделей имеют разное влияние на конечный результат работы моделей, и есть некоторое количество весов (супер-веса, super weights), которые в наибольшей степени определяют точность ответа, и, даже небольшое округление этих весов приводит к сильной деградации работы модели. Алгоритм квантизации должен уметь находить такие ключевые веса и ограждать их от излишнего округления. Поэтому, модель, квантизированная до одного и того же уровня плотности, но разными подходами и алгоритмами квантизации, может иметь принципиальное разное качество.

Процесс квантизации моделей является достаточно сложной процедурой, которая сама по себе требует в какой-то мере “обучения” алгоритма под каждую квантизируемую модель. Такие подходы “умной” квантизации начинают развиваться, и, в качестве примера можно привести проект *Unsloth Dynamic 2.0 Quants*, в рамках которого проведена такая “умная” динамическая квантизация множества самых популярных открытых языковых моделей. (На сайте по приведённой ссылке в каталоге представленных моделей, файлы LLM с динамической квантизацией обозначаются суффиксом *-UD-*). Подобная динамическая квантизация позволяет практически без потери качества модели значительно уменьшить её размер и время выполнения.

При выборе степени квантизации/сжатия модели для локального использования важное значение имеет область применения LLM: при написании программ или формул, т.е. там, где принципиальное значение имеет каждый символ и любые ошибки недопустимы, даже незначительная квантизация *Q8_0* может вносить существенные ошибки. Для практического применения всегда стоит опробовать и сравнивать модели в применении к конкретной задаче: и *FP16* модель меньшего размера, и модели *Q8_0*, *Q4_K_M* большего размера.

На практике, применение локальных LLM сильно ограничено их фактическим размером, в первую очередь это ограничение связано со скоростью шины памяти — такой большой массив данных как модель, просто физически долго прокачивать из памяти в процессор.

Для “плотных” моделей, как показывает опыт применения LLM, при отсутствии GPU, для современного процессора следует выбирать модель размером до 6 Гбайт (оптимально до 3 Гбайт), иначе будет работать совсем уж медленно. При наличии GPU, если размер модели помещается в VRAM, то работа модели будет в разы быстрее, что связано с, в разы более широкой шиной памяти VRAM. Если же модель полностью не помещается в VRAM, то та часть модели, что не поместилась в VRAM, будет обрабатываться процессором и будет иметь вышеизложенные соответствующие ограничения на размер.

Для *MoE* моделей ограничения по размеру на порядок слабее, и тут на первый план выходит скорее ограничение общего количества оперативной памяти — какая модель поместилась в оперативную память, с той и следует работать.

Подробнее о степенях квантизации в главе *О квантизации моделей*.

8.6 Неполный список открытых доступных LLM моделей

- семейства моделей Qwen3 и Qwen3-Coder, и модель QwQ-32B, от Alibaba;
- семейство моделей GLM4 от Zhipu AI;
- семейство моделей gemma 3n от Google;
- модель gpt-oss-20b от OpenAI;
- ещё есть семейство моделей ERNIE-4.5 от Baidu, но здесь есть сложности с запуском.

Практически для всех этих моделей компания Unsloth AI создала квантованные версии моделей в формате GGUF, которые доступны на странице https://huggingface.co/unsloth.

8.7 Специализация моделей

Современные, сильные универсальные модели имеют размер более сотни миллиардов весов (100B), что делает невозможным их локальный запуск. Небольшие модели, доступные по размеру для локального запуска, имеют те или иные сильные и слабые стороны; и выбор подходящей специализированной модели порой имеет большее влияние на успешность решения задачи, чем выбор модели большего размера.

Note: Этот список мало того что неполный, так ещё постоянно устаревает; и, скорее всего, на данный момент уже не актуален.

- Современные (2025) *MoE* модели, например Qwen3-30B-A3B, очень хорошо справляются с большинством типов задач; но не очень быстрые.
- Для математики, формул, программирования, работы с Markdown, хорошо работают модели для программирования, например Qwen2.5 Coder.
- Для математики и формул лучше всего работают *Reasoning* модели.

- Со сложными задачами могут справиться *только Reasoning* модели, например Qwen3-30B-A3B, QwQ-32B или DeepSeek R1.
- Для перевода текстов может применяться любая модель, которая понимает русский язык; но, есть специализированная модель Aya Expance, заточенная именно для *переводов на всевозможные языки*.
- Для ответов на простые вопросы не стоит привлекать *Reasoning* модели — времени потратится много, объём ответа вырастет, а точность будет не особо лучше.

Особенности работы *MoE* моделей на примере *Qwen3*

Весной 2025 вышел набор универсальных (гибридных) моделей *Qwen3*, все из которых одновременно являются и *Reasoning*, и *Non-Reasoning*. Также, среди набора моделей *Qwen3*, пара моделей являются *"Mixture of Experts"* (*MoE*) моделями, и меньшая из них (*Qwen3-30B-A3B*) отлично работает на персоналке/ноутбуке без выделенного видеоадаптера. В этом параграфе речь пойдёт именно об этой *MoE* модели *Qwen3-30B-A3B*, причём в варианте динамической квантизации от Unsloth.

Note: Гибридные модели проигрывают по качеству своим специализированным альтернативам, поэтому стоит ожидать, что в дальнейшем модели всё же будут разделяться на *Reasoning* и *Non-Reasoning*.

Модели *"Mixture of Experts"* (*MoE*) в каждый конкретный момент работы задействуют только небольшую часть весов, например модель *Qwen3-30B-A3B* задействует только 3B из 30B весов; что ведёт к ускорению работы *на порядок*, по сравнению с обычными, "плотными" моделями. Но, общая точность несколько падает, и соответствует, примерно, *в два раза* меньшим, по размеру, "плотными" моделями. Конкретный пример: квантизированная модель *Qwen3-30B-A3B-GGUF:Q4_K_XL*[17] на типичном процессоре выполняется в пять с лишним раз быстрее, чем плотная модель той же точности *Qwen3-14B-GGUF:Q4_K_XL*[18], и обеспечивает 10-15 токенов в секунду, или страницу текста в минуту (ср. со страницей в 5 минут).

Ещё одной особенностью *MoE* моделей является то, что квантизация не сильно ускоряет выполнение по сравнению с исходными моделями, т.е. время выполнения моделей не сильно зависит от степени квантизации. Одна из самых сильных квантизаций *Q2_K* выполняется всего-лишь в два раза быстрее исходной *FP16* модели (не смотря на то, что по объёму в 8 раз меньше), а квантизация *Q8_0* по скорости выполнения отличается от *FP16* максимум на 20–30 %. Поэтому, для *MoE* моделей, следует выбирать самую лёгкую степень квантизации, которую удастся загрузить в память компьютера. В частности, для модели *Qwen3-30B-A3B*, в зависимости от объёма оперативной памяти, выбор следующий:

- если памяти 16 ГБ, то можно попробовать запустить модель в квантизации *Q2_K_XL*[19]. (*Note:* 16 ГБ для *MoE* LLM модели занимающей 12 ГБ, это совсем впритык, но возможно.

[17] занимает 18 ГБ
[18] занимает 9 ГБ
[19] занимает 12 ГБ

Выполнение, скорее всего, будет несколько подтормаживать, не обеспечивая доступную для процессора скорость в 10 токенов в секунду);

- если памяти 32 ГБ и более, то хорошо подойдёт квантизация *Q4_K_XL*[20]. По вычислительным затратам на выполнение, эта квантизация, отличается менее чем на 30 % от меньшей *Q2_K_XL*;
- если памяти 64 ГБ, то можно попробовать *Q8_K_XL*[21], которая будет выполняться в полтора раза медленнее, чем *Q2_K_XL*, при, практически, идеальной точности, очень близкой по качеству к официальной онлайн модели *Qwen3-30B-A3B* на сайте https://chat.qwen.ai/.

При использовании программы *ollama* (см. далее) для загрузки LLM выполнить[22]:

```
ollama pull hf.co/unsloth/Qwen3-30B-A3B-GGUF:Q2_K_XL
```

Для гибридной языковой модели *Qwen3-30B-A3B* режим работы *Reasoning / Non-Reasoning* выбирается добавлением в конец запроса ключа /no_think для режима без размышлений. В качестве примера показано выполнение запроса к LLM из командной строки с ключом для отключения режима размышлений:

```
ollama run hf.co/unsloth/Qwen3-30B-A3B-GGUF:Q2_K_XL "Посчитай сумму цифр в числе 2 в 70 степени? /no_think" --ve
```

Модель *Qwen3-Coder* отлично работает не только с программированием, но и с обычными текстами:

```
ollama run hf.co/unsloth/Qwen3-Coder-30B-A3B-Instruct-1M-GGUF:Q2_K_XL "Напиши список существительных слов из
```

8.8 Программы для работы с локальными LLM

В этой главе рассмотрены несколько программ для применения LLM, из тех что попроще.

LLM провайдеры

Существует небольшое число инструментов, непосредственно выполняющих LLM: llama.cpp, vLLM, transformers, плюс какие-то графические — это LLM провайдеры. Есть менеджеры библиотек LLM моделей: Ollama, huggingface_hub. И, есть большое количество программ для запуска LLM — это интерфейсы, графические “оболочки”, которые запускают тот или иной LLM провайдер, и используют менеджеры библиотек LLM.

Safetensors, это исходный формат, в котором обучаются модели; он не поддерживает квантование. Работать непосредственно с форматом *Safetensors* умеют только *PyTorch*, *HuggingFace*

[20] занимает 18 ГБ
[21] занимает 36 ГБ
[22] занимает 12 ГБ

Transformers и *vLLM*. Остальные библиотеки и провайдеры конвертируют файлы моделей в другие форматы, более совместимые для применения на GPU, а также позволяющие провести квантование моделей.

Таблица 2: LLM провайдеры

Название	CPU	GPU	Формат LLM	Замечания
PyTorch	Да	Да	Safetensors	Базовая Python библиотека, написана на C++
HuggingFace Transformers	Да	Да	Safetensors	Использует PyTorch
vLLM	Нет	Да	Safetensors, GPTQ, AWQ	Написана на Python/C++, не использует сторонние библиотеки
llama.cpp	Да	Да	GGUF, может импортировать GPTQ/AWQ	Написана на C/C++, не использует сторонние библиотеки
Ollama	Да	Да	Свой формат, загружает через GGUF	Написана на основе llama.cpp

Параметры и шаблоны

При передаче модели запроса на выполнение, используются *шаблоны*, которые преобразуют запрос в специально промаркированный текст, на формате которых обучались модели — у каждой модели обычно свой шаблон, и отступление от конкретного формата запроса сильно влияет на качество ответа. Помимо применения обычных шаблонов при запросе, существуют шаблоны шаблонов (jinja) с широкими возможностями форматирования запроса в целевой формат шаблона (применяется в LM Studio).

Также стоит учитывать, что моделям в зависимости от режима работы (с размышлениями или без) требуется задавать разные наборы параметров LLM, таких как Temperature, TopP, TopK, MinP и т.д.

Поэтому, один и тот же запрос к одной и той же модели в разных программах, или даже просто в разных режимах, может выдавать очень разные по качеству ответы, в зависимости от применяемых параметров, шаблонов и шаблонов шаблонов.

Спекулятивное декодирование

Спекулятивное декодирование (*Speculative Decoding*), это подход, при котором вначале лёгкой моделью создаётся “черновик” цепочки токенов, после чего основная модель на основе чер-

новой цепочки генерирует ответ. Этот подход позволяет в два-три раза ускорить генерацию ответа.

8.8.1 LM Studio

https://lmstudio.ai/

Удобная программа “По одной кнопке”. Удобный встроенный поиск моделей, возможность настройки параметров модели при запуске, умеет использовать *спекулятивное декодирование*. Умеет загружать pdf/txt прямо при запросе. Но есть два больших минуса: не opensorce — *проприетарная*, и, *лицензия запрещает любое коммерческое применение результатов!* А так да, для “поиграться”, самое то.

Установить:

```
winget install --id=ElementLabs.LMStudio -e
```

Использование:

- для увеличения или уменьшения шрифта в интерфейсе использовать <*Ctrl*+> и <*Ctrl*-> (основан на *Electron*);
- для контроля над LLM необходимо включить режим “Power User” в самом низу, если ещё не включен;
- выбрать и скачать LLM модель: слева иконка лупы;
- управление параметрами запуска скачанных моделей (здесь же удаление): слева иконка открытой папочки;
- выбрать модель из уже скачанных для загрузки, или сменить загруженную модель на другую: сверху плашка с названием модели, при нажатии выпадет список доступных для загрузки;
- начать чат с моделью: слева в самом верху иконка сообщения;
- работа с документами: “скрепкой” подцепить файл и запрос будет работать с данными из файла.

Note: LMStudio (на 2024) использует устаревшую нотацию для отображения формул, поэтому, в LMStudio при запросах связанных с математикой, необходимо явно указывать использовать устаревшие символы, добавляя к запросу фразу: *(Use dollar and double dollar signs for math in answer)*.

Note: Существует *opensource* аналог LMStudio — **Jan**, но, конечно, пока с не столь широким функционалом. Установить можно через *Winget*.

8.8.2 Open WebUI (with Ollama)

https://openwebui.com/
Достаточно большие возможности при несложной установке и простом интерфейсе. Умеет отображать Markdown и LaTeX формулы.

Установить Ollama:

```
winget install --id=Ollama.Ollama -e
```

Установка[23] и запуск Open WebUI с использованием уже установленного Ollama:

```
docker run -d -p 3000:8080 --gpus all ^
    --add-host=host.docker.internal:host-gateway ^
    -v open-webui:/app/backend/data --name open-webui ^
    --restart always ghcr.io/open-webui/open-webui:cuda
```

Open WebUI при этом запустится как фоновый процесс (все его свойства/характеристики можно посмотреть в интерфейсе *Docker Desktop*). По адресу http://localhost:3000 появится приложение в виде веб-страницы.

В случае ошибки соединения *Open WebUI: Server Connection Error* есть другие варианты установки.

После установки вышеприведённую команду больше не потребуется запускать. Open WebUI можно запустить либо через графический интерфейс DockerDesktop, либо из командной строки:

```
docker start open-webui
```

Завершить Open WebUI: docker stop open-webui

Для скачивания моделей следует применять команды, скопированные со страницы https://ollama.com/library для выбранной LLM модели, где команду run заменить на pull, например:

```
ollama pull qwen2.5-coder:1.5b-instruct-fp16
```

(Вроде бы есть экспериментальная возможность загрузки моделей прямо изнутри Open WebUI, но это надо где-то включать)

При необходимости обновления Ollama до новой версии выполнить команду:

```
winget upgrade ollama
```

При необходимости обновления Open WebUI до новой версии выполнить команду:

[23] размер скачиваемого образа 4.0ГБ, займёт 8.5ГБ

```
docker run --rm --volume /var/run/docker.sock:/var/run/docker.sock ^
    containrrr/watchtower --run-once open-webui
```

Использование:

При первом запуске программа попросит задать имя, имейл и пароль, это чтобы никто другой не воспользовался данными пользователя — они хранятся в зашифрованном виде, а имейл и пароль никуда не передаются. Если забыть пароль, то данные будут потеряны.

- Выбрать модель из загруженных: сверху слева кнопка *Arena Model*.
- Новый чат/список чатов: слева сверху три горизонтальные полоски.

Note: В Open WebUI по умолчанию стоит включенной настройка *Rich Text Input for Chat*, которая пытается на лету превращать набираемый текст с разметкой Markdown в отформатированный текст, что приводит к излишней "оптимизации" и потере некоторых символов вроде перевода строк и т.п. Эту опцию следует отключить в *Settings* > *Interface* > *Chat* > *Rich Text Input for Chat*.

8.8.3 Page Assist (with Ollama)

Page Assist — это плагин для Chromium. Этот плагин умеет в разметку Markdown, рисовать LaTeX формулы, и подсвечивать программный код. Удобный инструмент, но на текущей версии (v1.3.4) начинает сильно тормозить, если поднакопятся чат-сессии или данные (bug).

Установить Ollama (если ещё не установлен):

```
winget install --id=Ollama.Ollama -e
```

Установить плагин для Chromium Page Assist - A Web UI for Local AI Models.

На странице https://ollama.com/models выбрать модель (например qwen2.5-coder:1.5b или 3b), скачать (с момента установки плагина появится кнопочка-стрелочка *Скачать* рядом с моделью), и запустить в Page Assist.

8.8.4 vLLM

<TBD>

https://github.com/vllm-project/vllm
LLM провайдер, поддерживающий большое количество форматов моделей, мультимодальные модели. Его главное преимущество — он может запускать исходные модели в формате *SafeTensors*, без преобразования в упрощённые форматы типа GGUF, без каких-либо изменений.

Может использоваться приложениями в той же роли что и Ollama.

Установка и запуск, вариант попроще

Установка:

```
pipx install vllm huggingface_hub[cli]
```

Скачивание модели с https://huggingface.co/ и запуск сервиса с *OpenAI API* доступного по адресу http://localhost:8088

```
vllm serve Qwen/Qwen2.5-Coder-3B-Instruct --port 8088
```

Здесь 8088 — номер порта, на котором будет работать сервис.

Установка и запуск

```
docker run --name vllm--Qwen--Qwen2.5-Coder-3B-Instruct --runtime nvidia --gpus all ^
  -v ~/.cache/huggingface:/root/.cache/huggingface -p 8088:8000 ^
  --ipc=host vllm/vllm-openai:latest --model Qwen/Qwen2.5-Coder-3B-Instruct
```

И запуск в дальнейшем

```
docker start vllm--Qwen--Qwen2.5-Coder-3B-Instruct
```

```
docker stop vllm--Qwen--Qwen2.5-Coder-3B-Instruct
```

Старт контейнера не мгновенный, зависит от размера модели — пока модель загрузится с диска в память и/или видеопамять.

API сервиса будет по адресу http://localhost:8088.

Запуск через docker предполагает свой контейнер на каждую LLM модель. Конечно, быстро модельки так не по-перебираешь, но *vLLM* на такое и не рассчитан, *vLLM*, это, скорее, про постоянно работающий сервис.

Note: Удалить docker контейнер: docker container rm vllm--Qwen--Qwen2.5-Coder-3B-Instruct. Переименовать docker контейнер: docker container rename vllm--Qwen--Qwen2.5-Coder-3B-Instruct new-name.

8.9 ИИ помощники

8.9.1 *Roo Code*

https://roocode.com/ https://github.com/RooCodeInc/Roo-Code, форк *cline*.

8.9.2 *Continue*

https://www.continue.dev/ https://github.com/continuedev/continue

<TBD>

Внедряем AI Code Assistant в разработку бесплатно и без вендорлока — Инструкция:

https://ollama.com/blog/continue-code-assistant

https://vsegpt.ru/ExtTools/Continue

8.10 Работа с документами

Для работы со своими документами/файлами применяется технология RAG — Retrieval-Augmented Generation. Продвинутые пользователи могут использовать более мощные инструменты, например SillyTavern, а здесь далее будут описаны способы попроще.

8.10.1 LMStudio

В этом приложении нет какого-то особого *RAG* или *Баз Знаний (Knowledge Base)*, всё намного проще — прикрепляются файлы “скрепкой”, и запускается запрос. Вполне неплохо справляется.

8.10.2 Open WebUI

В Open WebUI можно создать *Базу Знаний (Knowledge Base)* на основе локальных текстов, инструкция:

- Загрузить данные: *Workspace* > *Knowledge* > + *Create a Knowledge Base* и загрузить файлы.
- Создать Модель с загруженными данными: *Workspace* > *Models* > + *Add New Model*, выбрать исходную LLM модель, выбрать данные *Knowledge Source*, сохранить новую Модель-с-Данными (это может быть долго).
- Выбрать созданную Модель-с-Данными для нового чата.

8.10.3 Page Assist

В Page Assist для создания *Базы Знаний* на основе локальных текстов потребуется скачать *embedding* модель nomic, которая будет читать загруженные текстовые документы и транслировать их в векторное семантическое пространство невысокой размерности. В настройках *Settings* > *RAG Settings* выставить *nomic-embed-text* в качестве *Embedding Model*.

- В меню *Settings* > *Manage Knowledge* > *Add New Knowledge* загрузить нужные pdf/doc/txt, и дождаться их обработки.
- На главной странице, нажав снизу на *"квадратики"* выбрать нужную базу знаний. После чего запросы к LLM будут производиться с учётом данных из файлов.

Примечание по RAG технологии

Следует помнить, что на 2024г технология новая, отработана слабо, поэтому стоит учитывать, что, к примеру, данные могут выгрузиться в базу данных не полностью, или с ошибками, это особенно часто случается при работе с pdf, где разметка страницы может прерывать текст, а формулы без OCR вообще не вытащить. Существуют модели (и использующие их программы), которые могут не только выдёргивать тексты из pdf, но и умеют OCR pdf, картинок и видео, например Qwen2-VL. Но для их применения необходимы другие, более сложные программы, например vLLM, SillyTavern или quivr.

Примечание по LLM chain технологии

Создать одну универсальную модель гораздо сложнее, чем создать несколько небольших специализированных моделей заточенных на конкретную задачу. По технологии цепочки моделей, данные, полученные на выходе из одной модели, передаются на вход другой модели, формируя конвейер обработки данных.

8.11 Алгоритм работы нейросети-трансформера

Ссылки:

- wiki: Большая языковая модель
- wiki: Transformer
- Трансформер
- Объясняем простым языком, что такое трансформеры
- Transformer в картинках

Алгоритм трансформера

Несколько упрощённое описание алгоритм работы нейросети типа трансформер применяемой в LLM.

1. Входной текст разбивается на токены с помощью токенизатора, основанного на заранее обученном словаре (обычно 10–50 тыс. элементов). Этот шаг не является частью (слоем) нейросети, а является предварительной обработкой текста.

2. С помощью входного эмбеддинг-слоя (тензор token_embd) нейросети каждый токен преобразуется в числовой[24] вектор (эмбеддинг/embedding) фиксированной длины (например, 512), являющийся точкой в многомерном семантическом пространстве. Также к этим векторам-эмбеддингам добавляется *позиционное кодирование*, чтобы модель могла учитывать порядок слов. Над этим набором эмбеддингов проводится вся дальнейшая работа трансформера, причём, т.к. *позиционное кодирование* уже встроено в сами эмбеддинги, то порядок обработки эмбеддингов не важен, что позволяет эффективно распараллеливать обработку.

3. Алгоритм "механизм внимания" (self-attention) вычисляет, какие части входной последовательности наиболее важны для понимания каждой конкретной её позиции, т.е. устанавливает смысловые связи между всеми (каждый с каждым) точками семантического пространства (эмбеддингами). Для этого для каждого эмбеддинга создаются три вектора: Query (Q), Key (K) и Value (V), по которым вычисляется матрица внимания: *Attention(Q, K, V) = softmax(QK^T / sqrt(d_k)) * V* [25], которая, собственно, и описывает связи между емебеддингами и показывает, какие части входа наиболее важны при обработке конкретного токена. Полученная матрица внимания *Attention(Q,K,V)* (тензоры attn) описывает *представление* каждого эмбеддинга, учитывающее его контекст.

4. Полученные представления *Attention(Q,K,V)* проходят через Feed-Forward Network (FFN) (тензоры ffn), которая применяется независимо к каждому эмбеддингу (т.е. возможна параллельная обработка). После этого формируется новый набор эмбеддингов, уже с более глубоким семантическим смыслом; при этом количество эмбеддингов не изменяется.

5. Этот процесс повторяется несколько раз: последовательность эмбеддингов проходит через множество уровней, состоящих из пары слоёв: self-attention (п.3) + FFN (п.4). Обычно таких уровней от 12 до 48. На каждом уровне модель углубляет понимание контекста и взаимосвязей между элементами последовательности.

6. На выходе из сети получается последовательность эмбеддингов. Для предсказания следующего токена ответа используется специальный выходной эмбеддинг, вычисляемый в зависимости от заданного алгоритма: последний в последовательности эмбеддинг; вычисленный методом beam search; или др. методы.

7. Для сопоставления выходному эмбеддингу конкретного токена, этот выходной эмбеддинг проходит через дополнительный выходной *линейный слой* НС (тензор output), который преобразует его в вектор[26] размером со словарь токенов (см. п.1), а затем через функцию

[24] floating point, FP16
[25] *d_k = length(Q) = length(K)*
[26] называется logits

softmax — в вероятностное распределение для каждого токена из словаря. Следующий токен ответа выбирается согласно этому распределению вероятностей в зависимости от заданного алгоритма: жадный (greedy, temperature=0) — самый вероятный; случайный выбор среди наиболее вероятных с учётом температуры (шума); top-k sampling, или nucleus sampling (top-p) и др.

8. Выбранный токен ответа добавляется в конец к исходной последовательности, и весь процесс повторяется заново, начиная с п.2, пока не будет достигнут конец ответа (получен токен <EOS>), или превышена максимальная длина генерации.

Структура трансформера

Несколько приближённо, но можно сказать, что каждый слой НС представляет собой набор тензоров (матриц)[27], и прохождение информации через слой является операцией умножения этих матриц на каждый ембеддинг из набора, представляющего запрос плюс текущую часть ответа.

В зависимости от степени квантования непосредственный размер блоков будет различным, но размерности (*shape*) тензоров будут одинаковы для одной LLM в любой степени квантования.

Характерный список блоков (тензоров), а также их размеры, на примере квантованной *MoE* модели *Qwen3-30B-A3B-UD-Q4_K_XL*, для второго (.1.) слоя, плюс включены входные и выходные тензоры token_embd.weight, output_norm.weight, output.weight:

Таблица 3: Непосредственные размеры блоков LLM *Qwen3-30B-A3B-UD-Q4_K_XL*

Name	Shape	Length	Q10n, bits	Size, bytes	Size
output.weight	2048 x 151936	311164928	Q6_K	233373696	222 MB
output_norm.weight	2048	2048	F32	8192	8 KB
token_embd.weight	2048 x 151936	311164928	Q4_K	155582464	148 MB
blk.1.attn_k.weight	2048 x 512	1048576	Q5_K	655360	640 KB
blk.1.attn_k_norm.weight	128	128	F32	512	0.5 KB
blk.1.attn_norm.weight	2048	2048	F32	8192	8 KB
blk.1.attn_output.weight	4096 x 2048	8388608	Q5_K	5242880	5 MB
blk.1.attn_q.weight	2048 x 4096	8388608	Q5_K	5242880	5 MB
blk.1.attn_q_norm.weight	128	128	F32	512	0.5 KB
blk.1.attn_v.weight	2048 x 512	1048576	Q5_K	655360	640 KB
blk.1.ffn_down_exps.weight	768 x 2048 x 128	201326592	Q5_K	125829120	120 MB
blk.1.ffn_gate_exps.weight	2048 x 768 x 128	201326592	Q4_K	100663296	96 MB
blk.1.ffn_gate_inp.weight	2048 x 128	262144	F32	1048576	1 MB
blk.1.ffn_norm.weight	2048	2048	F32	8192	8 KB
blk.1.ffn_up_exps.weight	2048 x 768 x 128	201326592	Q4_K	100663296	96 MB

[27] называются блоками

Далее приведены некоторые пояснения о размерах тензоров (матриц) для ключевых слоёв в типичной Transformer-модели, информация от языковой модели Qwen:

1. Embedding Layer
    - Вход: индекс слова (или токена)
    - Выход: векторное представление токена
    - Матрица весов token_embd: (vocab_size × d_model)
        - vocab_size: ~30,000–100,000 (размер словаря)
        - d_model: ~512–8192 (размерность скрытого состояния)
    - *Пример: Для GPT-2 small (*d_model=768*,* vocab_size=50257*) — матрица будет иметь размер* ~50k × 768
2. Self-Attention (Q, K, V проекции). Каждый из трёх проекторов (Query, Key, Value):
    - Матрицы blk.attn_q, blk.attn_k, blk.attn_v: (d_model × d_k) или (d_model × d_v)
    - Обычно d_k = d_v = d_model // num_heads
    - *Пример: GPT-2 small имеет* d_model=768*,* num_heads=12 => d_k = d_v = 64 *Тогда каждая матрица Q/K/V:* 768 × 64 *Иногда эти три матрицы объединяются в одну:* d_model × (3*d_k)
3. Проекция после Self-Attention (Dense). После конкатенации голов внимания:
    - Матрица blk.attn_output: (d_model × d_model)
4. Feed-Forward Network (FFN). Обычно состоит из двух линейных слоёв:
    - Первый blk.ffn_up: (d_model × d_ff)
    - Второй blk.ffn_down: (d_ff × d_model)
        - d_ff: обычно в 2–8 раз больше d_model (например, 2048–30720)
    - *Пример: Для GPT-2 small* d_model=768*,* d_ff=3072 => *Матрицы:* 768×3072 *и* 3072×768
5. Output (Head / LM Head)
    - Проекция на логиты output: (d_model × vocab_size)
    - Может быть тем же весом, что и embedding (weight tying)

Таблица 4: Примеры приблизительных размеров матриц для популярных языковых моделей

Модель	d_model	num_ heads	d_ff	Embedding / Output	Q/K/V (на head)	FFN
GPT-2 small	768	12	3072	50k × 768	768 × 64	768 × 3072
GPT-2 medium	1024	16	4096	50k × 1024	1024 × 64	1024 × 4096
LLaMA 7B	4096	32	11008	32k × 4096	4096 × 128	4096 × 11008
GPT-3 (175B)	12288	96	49152	12k × 12k	12k × 128	12k × 49k

Note: Размеры матриц растут квадратично с увеличением d_model.

8.12 О квантизации моделей

В зависимости от степени квантования непосредственный размер блоков будет различным, но размерности (*shape*) тензоров будут одинаковы для одной LLM любой степени квантования.

Q означает, что используется метод квантизации GWQ (Gradient-Aware Weight Quantization).

Число после Q означает, что веса модели были округлены, *в среднем*, до ***x*** бит на один коэффициент, причём веса после квантизации хранятся в виде *целочисленных* (*Integer*) значений соответствующей длины ***x*** бит.

_K_ означает, что при квантизации веса были сгруппированы по 64 и более коэффициентов (*quatization groups*). Для каждой группы весов ищутся два числа: масштабный коэффициент (*scale*) и смещение (*zero point*), которые обеспечивают следующее соотношение: weight[i] = scale * quant[i] + zero_point для всех весов в группе; здесь weight, scale, zero_point, это *FloatingPoint* числа, а quant, это *целочисленные* значения. Коэффициент масштабирования (*scale*) и нулевая точка (*zero point*) используются как при округлении (квантовании), так и при операции инференса (применения LLM). scale и zero_point вычисляются взвешенным методом наименьших квадратов для каждой группы весов, и основную сложность представляет поиск "взвешенности" в зависимости от "важности" того или иного веса.

Q8_0 означает, что *все* коэффициенты в весах модели были без группировки квантованы до 8 бит на вес, или, аналогично, Q4_0 — квантованы до 4 бит на вес. Можно сказать, что это устаревающие методы.

Суффиксы _S, _M, _L в Qx_K_S, Qx_K_M, Qx_K_L означают, что некоторые тензоры были квантованы с большей битностью, чем основная доля блоков. В методе Qx_K_S все блоки квантованы в ***x*** бит. В Qx_K_M блоки blk.*.attn_v.weight, blk.*.attn_output.weight, blk.*.ffn_down_exps.weight квантованы до ***x+1*** бит. А в Qx_K_L ещё больше блоков квантовано в ***x+1*** бит, а некоторые и в ***x+2*** бит.

Квантизация IQx_ проводится с использованием матриц важности (*imatrix*), которые составляются из ключевых фраз, участков кода, в общем той текстовой информации, которую требуется сохранить в модели при квантизации. Те блоки тензоров, что слабо реагируют на матрицу важности округляются сильнее, чем блоки, которые "замечают" *imatrix*. Такой способ квантизации позволяет в значительной степени управлять информацией, которая остаётся в квантованной модели. У квантованных этим методом моделей, у некоторых блоков битность квантования может быть ниже, чем общая объявленная битность ***x*** в IQx_.

Динамическая квантизация _XL в Qx_K_XL означает, что тензоры attn и ffn квантуются в различном качестве в зависимости от степени их влияния на качество инференса, в результате степень квантования варьируется от блока к блоку, причём, степень квантования подбирается непосредственно для *каждого* блока для каждой модели. У всех блоков степень квантования x будет не ниже объявленной Qx_, у большей части блоков степень квантования будет Qx_. Пример вариативности степени квантования можно заметить в таблице Таблица 3 непосредственных размеров блоков, приведённой для модели *Qwen3-30B-A3B-UD-Q4_K_XL*, где часть блоков имеет размер Q5_K вместо Q4_K, а некоторые блоки имеют и Q6_K (не представлены в таблице); для примерного представления, количество блоков по степеням квантования для

этой конкретной модели: Q4_K: 290, Q5_K: 37, Q6_K: 11, среди них большая часть блоков с битностью более Q4_K это attn блоки. *Note:* Подход динамической квантизации может применяться и в отношении метода квантизации IQx_.

Ещё существует перспективная динамическая квантизация _R4 для квантования с низкими степенями битности больших LLM моделей, для выполнения на CPU без GPU, доступна только в виде одного инструмента ik_llama.cpp.

Чисто для справки: квантизации TQ1_0 и TQ2_0 предназначены, в первую очередь, для троичных моделей (ternary models), таких как *BitNet-b1.58*, к которых в качестве весов используются числа на основе тритов[28]. Но также предпринимаются попытки приспособить эти методы как альтернативу квантизациям Q1_K_ и Q2_K_.

9 *Приложение:* Консольные терминалы и приложения

Глава для продвинутых пользователей. В этой главе установка и настройка следующего:

- консольный терминал Windows Terminal + Clink
- консольный терминал Conemu+Clink / Cmder
- Neovim
- управление Docker'ом
- Backup/Restore WSL систем

Note: Далее в тексте будет применяться сокращение *WT — Windows Terminal*.

9.1 Ограничения командной строки Windows

В Windows * (wildcard) не раскрывается перед запуском программ, а передаётся прямо как звёздочка, и, скажем, обычная Linux команда ls * скажет, что нет такого файла *. У этого ограничения есть несколько решений:

- Использовать BusyBox, где в каждую команду встроен свой глоббинг и правильное распознавание \ | / в путях, и, тот же ls, из состава BusyBox, сработает как в Linux. BusyBox, это альтернатива сборкам Git-for-Windows/MSYS2. Этот вариант подходит для использования с *WT*/*Conemu*.
- Использовать доработанный *Conemu* — Cmder, где глоббинг встроен в консольные скрипты, которые перед запуском команд осуществляют подмену *wildcards* на аргументы, как в Linux shell.
- Другие варианты — это использовать настоящий Linux shell, например, в виде терминала с WSL в WT/Conemu/Cmder, или, использовать bash/zsh из проектов Git-for-Windows/MSYS2 или BusyBox.

См. также пункт Отступление об экранировани символов.

[28] Триты, это числа, которые могут принимать три значения, а не два, как биты

Отступление о WT/Conemu vs. Cmder

Cmder — это донастроенный **Conemu**, со встроенным набором стандартных Linux утилит, плюс Clink. Т.е. Cmder всё необходимое ~~носит с собой~~ при установке скачивает заодно. При запуске Cmder автоматически ищет установленные утилиты из *Git-for-Windows* и прописывает их в пути (контролируется опцией */nix_tools*). В случае с WT/Conemu эти утилиты изначально необходимо доустанавливать, и, возможно, прописывать вручную пути. Ещё у *Cmder* есть некоторое преимущество — это его мобильность (portable), т.е., что его можно носить и запускать преднастроенным с флешки. Из минусов Cmder — настройки по умолчанию для интерфейса весьма сомнительные, их обязательно придётся править.

Но, Cmder, при стандартной установке с Git-for-Windows, не умеет распознавать направление слешей: прямой и обратный, и, например, не сможет отработать такую команду:

```
fd --hidden "\.vhdx$" %HOME% | xargs -I@ cp -v @ E:\backup\wsl\
```

Поэтому для нормальной работы Cmder потребуется доустанавливать BusyBox, и использовать опцию загрузки /nix_tools 0, чтобы по умолчанию не дописывались пути с Git-for-Windows. Но в таком случае Cmder становится практически идентичен Conemu.

> В общем, Cmder — это достаточно противоречивая утилита, подкупающая портативностью и преднастроенностью. Но, при должной настройке, Windows Terminal или Conemu будут даже лучше, чем Cmder, за исключением момента с портативностью.

В данном руководстве будут рассмотрены оба варианта настройки терминала: и *WT/Conemu+Clink+BusyBox*, и *Cmder*. Если Cmder уже установлен и настроен, то предлагается настроить Windows Terminal на его использование. Если Cmder не установлен, то значит он вам не нужен и не надо с ним связываться. В этом случае предлагается установить и настроить Clink и BusyBox, чтобы получить универсальную командную оболочку, которая будет работать и в WT, и в Conemu. *В целом, Главы про настройку Conemu и Cmder приведены, в большей мере, по историческим причинам, чтобы не утерять данную информацию.*

9.2 Консольная оболочка Clink и утилиты BusyBox

Clink — это дополнение оболочки (shell) для *Cmd.exe*, интегрирующее следующие возможности: автодополнение, подсветка синтаксиса, история команд и поиск по ней, поиск файлов и папок, история переходов по папкам, алиасы.

BusyBox — это набор Linux *команд*, скомпилированных для Windows, которые поддерживают wildcard globbing, как в Linux. Будет использоваться для настройки *Windows Terminal* (или *Conemu*).

***Note**: Clink* не работает внутри *Far*, т.е, в терминале *Far* по <Ctrl+O> все супер-пупер автодополнения и поиск от *Clink* не сработают — для работы *Clink* должна использоваться отдельная консоль.

Note: ls из состава BusyBox не работает с русскими буквами, решение описано чуть ниже.

9.2.1 Установка Clink+BusyBox

Установить:

```
scoop install busybox
scoop install clink
scoop install clink-completions
scoop install clink-flex-prompt
scoop install oh-my-posh
scoop bucket add nerd-fonts
scoop install nerd-fonts/IosevkaTerm-NF-Mono
```

Note: ls из состава BusyBox не умеет показывать файлы с русскими буквами в имени файла. Поэтому, её следует заменить на утилиту ls из состава приложения git:

```
scoop shim add ls %UserProfile%\scoop\apps\git\current\usr\bin\ls.exe
```

Пока не пофиксят этот баг, данную процедуру придётся повторять после каждого обновления BusyBox, благо такие обновления могут быть проведены только вручную командой scoop update --all.

9.2.2 Настройка Clink

Настройка Clink для нечёткого (fuzzy) поиска, автоподстановки ~ и переменных окружения, интеграции с VSCode: скопировать соответствующие скрипты в отдельную постоянную папку и настроить Clink на их использование:

```
mkdir %UserProfile%\.config\clink-scripts
cd %UserProfile%\.config\clink-scripts

wget https://raw.githubusercontent.com/chrisant996/clink-gizmos/refs/heads/main/fzf.lua
wget https://raw.githubusercontent.com/chrisant996/clink-gizmos/refs/heads/main/fuzzy_history.lua
wget https://raw.githubusercontent.com/chrisant996/clink-gizmos/refs/heads/main/cwdhistory.lua
wget https://raw.githubusercontent.com/chrisant996/clink-gizmos/refs/heads/main/tilde_autoexpand.lua
wget https://raw.githubusercontent.com/chrisant996/clink-gizmos/refs/heads/main/vscode_shell_integration.lua

clink installscripts %UserProfile%\.config\clink-scripts
clink set fzf.default_bindings true
clink set autosuggest.strategy match_prev_cmd history completion fuzzy_history

clink set history.max_lines 100000
```

```
clink set clink.logo none
```

Алиасы для Clink

См. соответвующую главу *Алиасы и переменные окружения в консоли*.

9.2.3 Настройка внешнего вида CommandPrompt для Clink

Clink поставляется с несколькими вариантами темы оформления командной строки, их можно попробовать не устанавливая:

```
clink config prompt list
clink config prompt show <prompt_name>
```

Для постоянного использования включить тему оформления командой:

```
clink config prompt use <prompt_name>
```

Существует проект oh-my-posh для настройки вида командной строки для различных терминалов (CMD, PowerShell) с широкими возможностями настройки и большим списком уже преднастроенных вариантов. Включение oh-my-posh и выбор темы оформления:

```
clink config prompt use oh-my-posh
clink set ohmyposh.theme %UserProfile%\scoop\apps\oh-my-posh\current\themes\peru.omp.json
```

Выбранная тема оформления peru почти идеальна, единственный недостаток — не отображает текущую папку в заголовке окна. Это исправляется модификацией темы оформления: сначала скопировать файл исходной темы в новый файл peru-wtitle.omp.json:

```
mkdir %UserProfile%\.config\ohmyposh-themes
cd %UserProfile%\.config\ohmyposh-themes
cp %UserProfile%\scoop\apps\oh-my-posh\current\themes\peru.omp.json peru-wtitle.omp.json
```

затем добавить следующую строчку в файл peru-wtitle.omp.json почти в самый конец файла перед строкой "version": 3:

```
"console_title_template": "{{ .Shell }}: {{ if .Segments.Session.SSHSession }}
{{ .UserName }}@{{ .HostName }}{{ end }}{{ if .WSL }}WSL: {{ end }}{{ .PWD }}",
```

(это одна строка, только она не помещается на ширину формата текста, поэтому записана в две строки).

Переключить на новую созданную тему:

```
clink config prompt use oh-my-posh
clink set ohmyposh.theme %UserProfile%\.config\ohmyposh-themes\peru-wtitle.omp.json
```

9.2.4 Использование Clink

Списки горячих клавиш Clink; ниже приведены наиболее часто используемые.

Автодополнение Clink работает как в Linux:

- Tab — непосредственно дополнить в случае отсутствия неоднозначностей. В случае наличия вариантов они будут показаны, выбрать между ними вводом соответствующего символа и продолжить нажав Tab
- <Right> или End — принять предложенное в строке автодополнение до конца строки
- Alt+<Right> — принять предложенное в строке автодополнение на одно слово

В *Clink* консоли доступны удобные навигация и поиск по истории команд, навигация и поиск файлов:

- Ctrl+R — интерактивный поиск по истории команд
- Ctrl+T — интерактивный поиск по файлам и директориям в текущей директории
- Alt+C — быстрый переход из текущей директории в поддиректорию или к файлу
- Alt+B — интерактивный поиск по доступным горячим клавишам, и применение этих команд
- Ctrl+Space — интерактивный поиск по текущему автодополнению
- поиск по Tab после ** — рекурсивному поиску по директориям и файлам в текущей директории
- F7 — открыть окно со списком последних команд
- Up/Down — вверх/вниз по истории команд
- PgUp/PgDown — вверх/вниз по истории команд с учётом уже набранной части команды
- Shift+PgUp — интерактивное меню с историей посещённых директорий (при установленном *cwdhistory.lua*)
- Ctrl+Alt+U — эквивалентно cd .. — перейти в директорию выше
- .. — эквивалентно cd ..
- ... — эквивалентно cd ../..
- cd / или cd \ — перейти в корневую директорию
- ~ — перейти в домашнюю директорию
- c:, d: — переключиться на диск *C*, *D* и т.д.
- Ctrl+PgUp/Ctrl+PgDown — скрол консоли вверх/вниз.

Также удобное редактирование команд в командной строке:

- Ctrl+W — удалить слово слева от курсора
- Ctrl+U — удалить всю команду слева от курсора и положить в “буфер”
- Ctrl+Y — вставить из буфера (удалённое) по месту курсора
- End, Home, Ctrl+<Left/Right> — навигация по тексту
- Shift+End, Shift+Home, Shift+Ctrl+<Left/Right> — выделение текста в командной строке

- Shift+Mouse — выделение мышкой в неприспособленных для этого программах

9.3 Консольный терминал WindowsTerminal + Clink

После установки и настройки Clink останется только настроить профиль в WindowsTerminal.

Профиль для Clink в WindowsTerminal проще всего получить сдублировав профиль для *cmd.exe*, после чего в новом профиле в качестве запускаемой команды прописать:

```
%SystemRoot%\System32\cmd.exe /k "clink inject"
```

9.4 Консольный терминал Cmder/Conemu + Clink

Conemu/**Cmder** — это универсальный консольный терминал с открытым исходным кодом, позволяющий запускать все консольные приложения: *Far*, *Putty*, *CMD*, *PowerShell*, *WSL*, и пр., в одинаково выглядящем терминальном текстовом окне, с одинаково настроенным типом и размером *шрифта*, поведением горячих клавиш, размером окна.

Git-for-Windows — это *git*, скомпилированный для Windows; также содержит набор Linux команд, но эти команды не поддерживают wildcard globbing. Устанавливается автоматически при работе со *Scoop*. Используется в *Cmder*.

Some links: WSL, Pageant, Msys, 256colors.

После установки *Clink* и *BusyBox*, установить *Cmder* или *Conemu*:

```
scoop install cmder
```

```
scoop bucket add extras
scoop install extras/conemu
scoop install extras/conemu-color-themes
```

Для использования *Clink* в *Conemu* необходимо добавить вызов самого Clink: в настройках Conemu *Settings* > *Startup* > *Tasks:* модифицировать строку запуска Задачи/Task *{Cmd}*, заменив на

```
cmd.exe /k "clink inject && %ConEmuBaseDir%\CmdInit.cmd"
```

Настройки Cmder/Conemu

- В Windows11 консольные программы в Cmder/Conemu могут неправильно отображать цвета, глючить — в таком случае надо снять галочку *Inject ConEmuHk* в Cmder/Conemu в *Settings* > *Features: In-console options*; и выбрать цветовую схему *Monokai*.

- Для показа иконок в *Taskbar*'е в Windows должна быть включена опция *Show badges on taskbar buttons* в *Settings* > *Personalization* > *Taskbar*.
- Одним из интересных вариантов интеграции Cmder в Windows будет настройка *Панели задач* (*Taskbar*) без группировки окон — чтобы каждый терминал был в отдельном окне Windows; при этом надо установить опции для открытия каждого терминала в новом окне в *Settings* > *General* > *Appearance: Generic* убрать первые две галочки — *Single instance, Multiple consoles*.
- Сменить фон на непрозрачный: *Settings* > *Features* > *Transparency*.
- Настроить шрифт: *Settings* > *General* > *Fonts*, требуется шрифт с набором иконок NerdFonts, любой из них можно установить с помощью Scoop.
- Каждый терминал в своём окне: *Settings* > *General* > *Appearance: Generic* > *Single instance mode* — отключить.
- Разрешить мышку в *Far*: *Settings* > *Keys&Macro* > *Mouse: Mouse Options* > *Send mouse events to console*.
- При желании Табы можно расположить сверху, а не снизу: *Settings* > *General: Tabs* > *Tabs on bottom*.
- Настроить вид курсора: *Settings* > *Features* > *Text cursor*.
- Отключить проверку обновлений: *Settings* > *General* > *Update*.
- Комбинации Ctrl+C, Ctrl+V, Ctrl+W, Ctrl+T для доступности в консольных приложениях следует перенастроить (в настройках в *Cmder(Conemu)* > *Settings* > *Keys & Macro*) на варианты с Shift, т.е.: Ctrl+Shift+C, Ctrl+Shift+V, Ctrl+Shift+W, Ctrl+Shift+T и т.п.
- Действие клика правой кнопкой мыши настраивается в *Settings* > *Keys&Macro* > *Mouse: Mouse button actions* -> сделать *Paste*.
- Cmder/Conemu имеет большое количество настроек, поэтому рекомендуется, разок его настроив, сохранить файл настроек ConEmu.xml, и в дальнейшем, применять его на других своих компьютерах.
- mc под Windows: сменить диск: Alt+D

Использование Cmder/Conemu

Списки горячих клавиш: Cmder (Conemu) и Clink; ниже приведены наиболее часто используемые.

Управление терминалами Cmder/Conemu:

- Ctrl+` — глобальная комбинация клавиш вызова Cmder, не работает в поле ввода
- Ctrl+Shift+T — диалог открытия новой вкладки/окна (по умолчанию Ctrl+T, см. *Настройки Cmder* ниже)
- Ctrl+Shift+W — закрыть вкладку/окно (по умолчанию Ctrl+W, см. *Настройки Cmder* ниже)
- Ctrl+D — завершить терминальную сессию, эквивалентно команде *exit*
- Shift+Alt+#Number — открыть новую вкладку/окно по быстрому набору из меню открытия новых терминалов, см. Win+Alt+T
- Alt+Enter, Ctrl+Win+Enter — Full Screen

Управление вкладками Cmder/Conemu работает когда все консоли открыты в одном окне Cmder/Conemu, а не когда каждый терминал в своём собственном окне. Управление вкладками:

- Ctrl+#Number — переключиться на вкладку номер #Number
- Ctrl+Tab — переключиться на следующую вкладку
- Ctrl+Shift+Tab — переключиться на предыдущую вкладку

9.5 Алиасы и переменные окружения в консоли

Алиасы для Clink

Алиасы в Windows создаются командой doskey. В стартап-файле для Clink %LOCALAPPDATA%\clink\clink_start.cmd можно прописать соответствующие doskey команды:

```
@echo off

set HOME=%UserProfile%
set LANG=en_US.utf8

doskey ll=ls --color=auto -lF $*
doskey l=ls --color=auto -lrtF $*
doskey ls=ls --color=auto -F $*

doskey sort=%UserProfile%\scoop\shims\sort.exe $*
doskey time=%UserProfile%\scoop\shims\time.exe $*
```

В результате в командной строке появятся привычные для Linux алиасы. В данном случае определены:
ls — выводит простой список файлов и папок,
ll — выводит подробный список файлов и папок,
l — подробный список, отсортированный по времени модификации файлов/папок.

В этом стартап файле так же можно прописать другие настройки, например, переменные окружения.

Алиасы для Cmder

В настройках в *Settings* > *Startup* > *Environment* есть поле для стартовых настроек. В это поле можно добавить свои переменные окружения и алиасы:

```
set LANG=en_US.utf8
```

```
unalias ll
unalias l
unalias ls

alias ll=ls --show-control-chars -F -l --color --ignore={"NTUSER.DAT*","ntuser.dat*"} $*
alias l=ls --show-control-chars -CFGNhplrt --color --ignore={"NTUSER.DAT*","ntuser.dat*"} $*
alias ls=ls --show-control-chars -CFGNhp --color --ignore={"NTUSER.DAT*","ntuser.dat*"} $*
```

(Ещё потребуется удалить соответствующие алиасы из файла %HOME%\scoop\apps\cmder\current\config\user_aliases.cmd)

Алиасы для Conemu

Аналогичные алиасы для Conemu, в *Settings* > *Startup* > *Environment*:

```
set HOME=%HOMEDRIVE%%HOMEPATH%
set LANG=en_US.utf8
alias ll=ls --color=auto -lF $*
alias l=ls --color=auto -lrtF $*
alias ls=ls --color=auto -F $*
```

А можно использовать те алиасы, что настроены в *Clink*.

9.5.1 Замечание про *HOME*

- Для *Clink* переменную *HOME* можно прописать в стартап файле %LOCALAPPDATA%\clink\clink_start.cmd, см. раздел Алиасы для Clink.
- В случае с *WindowsTerminal* переменную *HOME* можно задать в переменных среды окружения для пользователя:

  ```
  setx HOME "%UserProfile%"
  ```

  , либо, при использовании *Clink*, она в нём была прописана.
- *Cmder* самостоятельно создаёт переменную *HOME*.
- В *Conemu* переменную *HOME* можно определить в настройках *Startup* > *Environment* как HOME=%HOMEDRIVE%%HOMEPATH% или HOME=%UserProfile%.

9.6 Neovim

Neovim это обновлённый *Vim*, конфигурации для которого пишутся на вполне понятном Lua.

9.6.1 Установка

Установка полноценного Neovim в Windows. Возможно какие-то пакеты уже были установлены — scoop пропустит повторную установку:

```
scoop install llvm
scoop install gcc
scoop install busybox
scoop install curl
scoop install luarocks
scoop bucket add versions
scoop install versions/lua51
scoop install python
scoop install ripgrep
scoop install fzf
scoop install fd
scoop install iconv
scoop install make
scoop install nodejs-lts
scoop install tree-sitter
scoop install yarn
scoop install neovim
```

Выйти и зайти заново в терминал, чтобы прописались пути для *python* и проч., после выполнить:

```
python -m pip install --upgrade pynvim
npm install -g neovim
```

Установить настроенную конфигурацию kickstart.nvim с предустановленным набором плагинов, и набор словарей для проверки правописания:

```
git clone https://github.com/denius/kickstart.nvim.git %LOCALAPPDATA%\nvim

mkdir %LOCALAPPDATA%\nvim\spell
cd %LOCALAPPDATA%\nvim\spell

wget https://ftp.nluug.nl/pub/vim/runtime/spell/ru.utf-8.spl
wget https://ftp.nluug.nl/pub/vim/runtime/spell/en.utf-8.spl
wget https://ftp.nluug.nl/pub/vim/runtime/spell/en.utf-8.sug
wget https://ftp.nluug.nl/pub/vim/runtime/spell/ru.utf-8.sug

cd %HOME%
```

Запустить:

```
nvim
```

Note: если в Windows11 в Cmder/Conemu *nvim* после запуска покажет чёрный экран, то надо снять галочку *Inject ConEmuHk* в Cmder/Conemu в *Settings* > *Features: In-console options*. И выбрать цветовую схему <*Monokai*>. *Note:* для 256-цветного терминала, возможно, потребуется донастройка.

> *Пояснение о внутренней механике Neovim.*
> В Neovim за *синтаксис* отвечает плагин *treesitter*, это парсер; в установленной конфигурации kickstart.nvim он уже установлен; treesitter самостоятельно заботится о парсинге всевозможных форматов. За *семантику* отвечает встроенный плагин *lsp.nvim*; *LSP* — это аббревиатура для *Language-Server-Protocol*, универсального сервиса, который обеспечивает: проверку корректности кода и поиск ошибок, автодополнение, переход к определению, подсказки и т.д.; LSP — это, де факто, стандарт в отрасли разработки, применяется во всех современных IDE и редакторах, в VSCode в том числе. Сервисы LSP для необходимых форматов устанавливаются отдельно, см. далее.

В процессе первого запуска nvim загрузит с интернета плагины в соответствии с конфигурацией kickstart.nvim. Для загрузки и обновления плагинов вручную набрать :Lazy и U.

После выйти и заново войти в nvim, и набрать :Mason для запуска менеджера *LSP*-сервисов для настройки поддерживаемых языков программирования и форматов. В нём найти (/) строчку clangd и нажать i для установки поддержки языка C/C++; аналогично установить fortls для Fortran, python-lsp-server и pyright для Python. Для установки других форматов их следует поискать в списке существующих LSP серверов.

После всех настроек можно проверить, как всё это установилось, командой :checkhealth. Возможно будет указано несколько Warning: про pyenv, про недостающие языки типа ruby или php, но это не имеет принципиального значения. Основное — надо убедиться что: а) плагин *treesitter* (парсер) нормально заработал, и, что: б) плагин *LSP* для нужного типа файлов работает — LSP будет выводить отдельную диагностику для каждого типа файлов.

9.6.2 Использование

9.6.2.1 *which-key*

В установленной конфигурации kickstart.nvim предустановлен плагин which-key.nvim, показывающий справку по клавишам при нажатии на клавишу <Пробел> (в kickstart.nvim на <Пробел> настроен <Leader>). То есть после нажатия на пробел будет показан список команд с ведущим <Leader>. Нажав *BackSpace* (возврат) будет показан общий список комбинаций клавиш, причём некоторые из них многосимвольные (напр. *g*, *z*, *[* и пр.), их списки буду раскрыты при нажатии на соответствующий символ.

9.6.2.2 Поиск

Также, среди предустановленных есть плагин Telescope.nvim, позволяющий в интерактивном режиме производить *поиск* по всему что есть в редакторе: файлу, файлам, истории, поиск файлов; посмотреть состояние настроек и переменных, клавиатурные сочетания; позволяет сменить тему оформления: :Telescope colorscheme (например *shine*, *wildcharm*). В целом, все данные, которые есть в редакторе доступны через этот плагин. Запускается через команду :Telescope <TAB>, высветится список возможностей *Telescope*, выбрать и применить. Список преднастроенных в kickstart.nvim комбинаций клавиш (в режиме NORMAL) для *Telescope*:

- <leader>sh — [S]earch [H]elp
- <leader>sk — [S]earch [K]eymaps
- <leader>sf — [S]earch [F]iles
- <leader>ss — [S]earch [S]elect Telescope
- <leader>sw — [S]earch current [W]ord
- <leader>sg — [S]earch by [G]rep
- <leader>sd — [S]earch [D]iagnostics
- <leader>sr — [S]earch [R]esume
- <leader>s. — [S]earch Recent Files (“.” for repeat)
- <leader><leader> — [ ] Find existing buffers
- <leader>/ — [/] Fuzzily search in current buffer
- <leader>s/ — [S]earch [/] in Open Files
- <leader>sn — [S]earch [N]eovim files

9.6.2.3 Русский язык и правописание

В установленной конфигурации, включение-выключение подсветки проверки правописания будет происходить по команде <LocalLeader>s = \s, а переключение языка по Ctrl-6 (в документации обозначается как Ctrl-^) в режиме редактирования (INSERT) или режиме команд (COMMAND), в нормальном режиме (NORMAL) команда Ctrl-^ не работает. Для режима команд индикатор текущей кодировки не отображается.

9.6.2.4 Некоторые команды и настройки Neovim

- В меню выбора из командной строки, стрелки влево/вправо бегают по вариантам, например, набрать :colorscheme и <Tab>, появится меню, перемещаться по которому следует клавишами влево-вправо, Ctrl-Y или <Enter> — выбрать.
- В меню *autocomplete*, при редактировании в режиме INSERT, выбор варианта производится стрелками *вверх*/*вниз*, а окончательный выбор — Ctrl-Y (*Y*, это от *Yes*).
- Alt-1, Alt-2, ... — переключиться на таб №; <LocalLeader>h, <LocalLeader>l — переключение между табами (это кастомная настройка).
- * в режиме NORMAL — поиск слова под курсором.

- gcc/gc — закоментировать-раскоментировать строку или выбранную область, в нормальном и визуальном режимах, соответственно.
- [d и]d — переход к предыдущему или следующему диагностическому сообщению LSP. <C-W>d — открыть окошко под курсором для вывода сообщения LSP полностью.
- LSP не только подсвечивает ошибки, но также может некоторые из них исправлять — при этом он высветит сообщение *fix available*. Комбинация <Leader>ca (вызывает *:vim.lsp.buf.code_action()*) производит исправление.
- На <LocalLeader>p настроен показ истории yank (буфера копи-пасты).
- Для перманентной смены схемы подсветки её необходимо прописать в конце файла *%HOME%\AppData\Local\nvim\init.lua* в виде строки vim.cmd.colorscheme 'wildcharm' (здесь *wildcharm* — это название схемы), плюс определить вариант light/dark, если у схемы есть такой выбор: vim.o.background = 'light'.
- В Windows также можно настроить работу Neovim через WSL, и запускать его: wsl vi somefile.

9.6.2.5 Документация

https://neovim.io/doc/user/

Команды nvim на stackoverflow.

Есть множество шпаргалок по командам Vim/Neovim, например: https://devhints.io/vim.

Если что-то непонятно в Vim, то всегда можно спросить ИИ, например, DeepSeek или Qwen. Также, они всегда могут подсказать команды для того или иного действия.

10 *Приложение:* Виртуалки

10.1 Управление Docker'ом

Docker работает с контейнерами (виртуальными машинами) (containers), образами (images), и томами (volumes). *Контейнер* — это набор из *образ* + *том*.

При скачивании или создании виртуальной машины (контейнера) Docker создаёт *образ* (*image*), который не изменяется, он *read-only*. *Образ* — это несколько файлов, со специальной внутренней структурой в виде файловой системы OverlayFS, т.е. внутри нескольких файлов, из которых состоит *образ*, хранится множество других файлов. Файловая система OverlayFS характерна тем, что она может хранить несколько версий одного и того же файла, но показываться будет только самая "верхняя" версия файла. *Образ* состоит из слоёв — при создании или обновлении виртуалки на него накатываются новые слои, может какие-то слои откатываются и переписываются, или вообще стираются. В нижележащем слое образа, как правило, хранится слепок операционной системы в самом минимальном виде, какой только может быть. На последующих слоях

образа хранятся файлы тех или иных программ и библиотек, которые были добавлены при создании образа. docker images — вывести список всех хранимых образов.

При запуске *контейнера* все созданные в процессе работы данные записываются в *том* (*volume*). Если при запуске контейнера была указана опция --rm, то использовавшийся при работе *том* будет удалён после окончания работы контейнера. docker volume ls — список сохранённых томов. Данные постоянно хранятся в томах, в том числе между запусками контейнеров; хранятся пока не будет удалён содержащий их контейнер — docker container rm <container id>, либо не будет удалён вручную — docker volume rm <volume id>.

docker container ls — список запущенных контейнеров, docker container ls -a — список всех контейнеров в системе.

В Windows и образы и тома хранятся в одном файле:
C:\Users\user\AppData\Local\Docker\wsl\disk\docker_data.vhdx — удобно для контроля занимаемого места (см. так же утилиту *WinDirStat*).

10.1.1 Запуск Docker контейнеров

Самостоятельно формировать команды запуска Docker контейнеров вряд ли придётся, но для понимания запускаемых строчек краткое пояснение опций запуска команды docker run:

- -t — запуск в консольном режиме;
- -i — запуск в интерактивном режиме;
- -d — запустить виртуалку в бэкграунде, т.е. она продолжит работать фоновом режиме;
- -v path1:path2 — смонтировать локальный путь path1 внутри контейнера в точке path2; это требуется чтобы работать с локальными данными изнутри контейнера;
- --rm — удалить созданные контейнером данные после завершения работы контейнера, т.е. удаляется соответствующий том (данные из папок, смонтированных с -v, не затрагиваются);
- --name some-container-name — задаёт имя для созданного контейнера.

После создания контейнера командой docker run все опции запуска будут сохранены внутри контейнера, их более не придётся где-либо вводить.

Используя имя созданного контейнера его в дальнейшем можно запускать и останавливать короткими командами:

```
docker start some-container-name
```

```
docker stop some-container-name
```

10.2 Backup/Restore WSL систем

Посмотреть какие WSL системы установлены, какая основная:

```
wsl -l -v
```

Скопировать/забэкапить WSL систему, здесь *Ubuntu* — название WSL системы, которая будет забэкаплена:

```
wsl --export Ubuntu D:\backup\ubuntu-24.04-20241117.tar
```

Восстановить из бэкапа, здесь *ubuntu2* название новой WSL системы, путь C:\wsl\ubuntu — место где будет лежать файл-контейнер *.vhdx* с новой восстановленной системой (чтобы далеко не прятался и можно было глянуть сколько места занимает):

```
wsl --import ubuntu2 C:\wsl\ubuntu D:\backup\some-file.tar
```

После восстановления из бэкапа новой машины необходимо указать пользователя по-умолчанию: ubuntu2.exe config --default-user aero, где *ubuntu2.exe*, это название новой VM, для которой Windows автоматически создаст приложение для запуска.

Сделать *ubuntu2* основной, запускаемой по умолчанию: wsl -s ubuntu2

В дальнейшем можно будет удалить старую WSL машину: wsl --unregister Ubuntu, при этом все данные внутри неё будут потеряны. Для безопасности лучше сначала импортировать в новое имя, установить её по-умолчанию, поработать с новой, а потом уже удалять старую — при работе через команду wsl название виртуалки не имеет значения.

10.2.1 Установка произвольных Linux

Аналогичным способом, из архива, можно создать новую Linux систему на компьютере, скопировав и установив соответствующий дистрибутивный файл Linux-системы с интернета, например со страницы проекта wsldl.

Fedora

<TBD>

https://fedoraproject.org/wiki/Changes/FedoraWSL

Fedora имеет преимущество перед *Ubuntu*, так как *RedHat*/*Fedora* — это изначально ориентированный на *HPC* Linux дистрибутив, и в нём лучше поддержка параллелизации, отладки Fortran, и прочим разделам высокопроизводительных вычислений.

10.2.2 Импортирование локальных образов с Docker

Docker также может импортировать бэкапы полноценных Linux систем (не WSL) в виде *.tar.gz* архива:

```
docker import <filename> <repository>:<tag>
```

```
docker import some-linux-backup.tar.gz linux-test:1.0
```

С последующим запуском

```
docker run -i -t <image id> <commands>
```

где в качестве <*commands*> для Linux может выступать */bin/bash*; <*image id*> посмотреть командой docker images:

```
docker images
docker run -i -t linux-test:1.0 /bin/bash
```

Таким методом можно создавать/копировать и запускать виртуалки с Linux (например, с менеджерами лицензий, и т.п.) из бэкапов, созданных с помощью команды tar изнутри копируемой Linux системы:

```
tar --one-file-system -cvpzSf /mnt/backup/some-ubuntu-backup.tar.gz / /boot --exclude="/mnt/*"
```

11 *Приложение:* Дистрибуция продуктов Microsoft

11.1 BitLocker

BitLocker — это встроенный в Windows механизм шифрования дисков для обеспечения безопасности информации. По умолчанию, при использовании учётной записи Microsoft, при установке Windows диск C: будет автоматически зашифрован. При форматировании новых дисков (например D:), они тоже будут зашифрованы, если C: был зашифрован.

Windows в процессе загрузки открывает ключ шифрования и начинает прозрачно для пользователя расшифровывать данные на диске, т.е. пользователь не замечает, что файлы зашифрованы. Но, если, этот диск вынуть, и подключить к другому компьютеру, то без ключей шифрования считать информацию с него не удастся — он зашифрован.

Ключи шифрования хранятся: на диске C:, в модуле TPM (если есть), в облаке учётной записи пользователя Microsoft; так же их можно экспортировать в разделе *Управление BitLocker* (48 цифр).

При создании только локальной учётной записи BitLocker автоматически не включается, и шифрование не проводится (хотя его можно активировать вручную).

При переустановке Windows важно учитывать, что если на компьютере, помимо системного диска C:, были другие зашифрованные диски, то без входа в учётную запись Microsoft они так и останутся зашифрованы. Если в планах использование только локальной учётной записи, перед переустановкой необходимо, либо отключить шифрование BitLocker (что приведёт к дешифровке дисков), либо убедиться, что ключи шифрования сохранены и доступны для использования.

11.2 OEM, Retail или Volume?

Лицензии OEM — это лицензии на ПО, поставляемые производителями и сборщиками техники вместе с новыми ноутбуками и компьютерами. Microsoft выделяет производителям техники целые пулы OEM лицензий по небольшой цене. И, по Европейским законам, производители могут продавать не только продукт целиком, но и комплектующие к нему, и, в данном случае, они могут продавать эти лицензии отдельно от компьютерной техники. Каждая OEM лицензия входит в какой-то конкретный пул лицензий, который закреплён за конкретным производителем, т.е., OEM, это не какая-то абстрактная безликая лицензия, а выпущенная определённым производителем. В целом, покупка OEM лицензии достаточно безопасна, случаев отзыва пулов лицензий очень и очень немного. Из особенностей OEM лицензии следует отметить, что она, в отличии от Retail или Volume лицензий, "одноразовая", то есть, активировав её один раз, *ключ* нельзя применить второй раз, и программу нельзя будет переустановить или перенести на другой компьютер. Не может быть активирована сервисом KMS.

Retail (розничная) лицензия — это привычная "коробочная" (FPP) версия из магазина, сейчас может поставляться в виде электронного ESD ключа. Эту лицензию можно переустановить (для MSOffice не более трёх раз) на другой компьютер (на старом, естественно, лицензия перестанет работать). Должна быть активирована в течении месяца с момента установки. Не может быть активирована сервисом KMS.

Volume (VL), это (корпоративные) лицензии для большого количества лицензий (от 5 штук) — поштучно не продаются. Помимо коммерческих предприятий часто используются в образовательных учреждениях. Volume лицензии активируются с помощью KMS сервиса — службы предоставляемой сервером. Активации с помощью KMS сервиса по локальной сети, это совершенно стандартная процедура, которая происходит автоматически для продуктов с Volume лицензией при правильной настройке свойств локального сетевого подключения (при получении IP-адреса, по DHCP, в качестве DNS сервера будет прописан адрес KMS сервиса). В случае подсоединения через интернет, возможность KMS активации ограничена.

- OEM — может быть установлена только один раз. Эту лицензию нельзя перенести на другой компьютер. Должна быть активирована в течении месяца с момента установки.
- Retail — лицензию можно переустановить на другой компьютер (MSOffice не более трёх раз). Должна быть активирована в течении месяца с момента установки.
- Volume — Использует не ключи, а сервисы лицензий KMS. Нет ограничений на перенос на другой компьютер.

11.3 Выбор версии MSOffice

Аргументы против обновления версии MSOffice: Все последние версии MSOffice, начиная с 2016, это всё одна версия 16.0, т.е., по сути, все новые Офисы, это 16 версия с сервис-паками, ответственными за улучшения интерфейса, но не принципиальные изменения. Единственное препятствие к работе в прошлых Офисах, это окончание срока поддержки и прекращение обновлений (безопасности). Поэтому, если никакой MSOffice не установлен, то следует устанавливать самую новую версию, если же MSOffice версий 2016 (или новее) уже установлен и устраивает, то пусть и стоит.

Аргументы за обновление версии MSOffice: Изменения пользовательского интерфейса, а значит и пользовательского удобства могут улучшаться с новыми версиями; например, в PowerPoint, появилось выравнивание элементов на слайде (я не помню с какой версии), и это значительно улучшило удобство использования. Поэтому, по возможности, следует опробовать новые версии MSOffice на предмет улучшения удобства использования, и, если офис стал лучше, то переустанавливать на более новый.

LTSC (Long-Term Servicing Channel) — это версия Офиса для корпоративного сегмента, где после установки в процессе эксплуатации не должен как-либо меняться функционал или предварительные настройки; поэтому, *LTSC*, это версия Офиса с обновлениями безопасности, но без обновления функционала. Поэтому, при возможности выбора, лучше установить полноценную версию с *Current* канала; но, если *LTSC*-версия уже установлена, то и ладно.

Office 365 — это офис с ежемесячной или ежегодной подпиской, требует постоянного подключения к интернету. Сейчас, в отсутствии возможности оплачивать онлайн, ни в коем случае не стоит связываться с *Office 365*. *Note:* Сейчас Microsoft старается продвигать только *Office 365*, а все упоминания об обычном бессрочном автономном Офисе удалены со страниц сайта Microsoft.

11.3.1 Дистрибутивы автономного MSOffice:

- Office Home & Student — Word, Excel, PowerPoint, OneNote;
- Office Home & Business — + Outlook;
- ~~Office Standard — + Publisher, Web Apps;~~ сейчас не продаётся;
- ~~Office Professional — + Access, SharePoint Workspace;~~ сейчас не продаётся;
- Office Professional Plus — + InfoPath, Skype for Business или Lync или Microsoft Teams, OneDrive for Business (ранее был SharePoint Workspace).

Note: Раньше (2003) редактор математических формул был только в Professional версии, как обстоит дело сейчас давно никто не проверял. *CHECKME: <TBD>*

11.3.2 Итого по выбору *MS Office*:

- **если не используется MS Access, то Office Home или Student *абсолютно достаточно*!**

- **если установлен офис версии 2016 (или новее) и он устраивает, то обновляться не требуется;**
- **если офис не установлен, то следует устанавливать самую последнюю версию (2024) с канала *Current*.**

11.4 UUP (Unified Update Platform) — загрузка MS Windows

> Загрузка с портала *Microsoft Unified Update Platform*, это официальный способ получения установочных пакетов и обновлений для продуктов Microsoft.

Для установки рекомендуется использовать версию *build 22631.3296* (*23H2*). Это версия, с одной стороны, достаточно свежая, чтобы были драйвера для новых ноутов, а с другой стороны, эта версия всё ещё позволяет при установке создать локальную учётную запись без необходимости онлайн регистрации учётной записи Microsoft (можно использовать и более новые версии при настройке с помощью *Rufus*, см. далее в *Создание загрузочной флешки*).

11.4.1 uupdump.net

https://uupdump.net/ — это сайт для создания скрипта, который скачает с сайта Microsoft и создаст загрузочный iso-образ Windows. Сайт https://uupdump.net/, возможно, недоступен в России, в этом случае следует воспользоваться другим вариантом.

Использование: на сайте выбрать желаемую версию, понажимать далее, в процессе отказаться от скачивания последних обновлений, и, скачать zip-архив (небольшой). Этот архив распаковать в папку на диске с достаточным количеством свободного места (12 ГБ?), и запустить скрипт из распакованного архива, uup_download_windows.cmd (выполняется около часа). Если есть сомнения какую версию выбрать, то вот ссылка на Windows11 23H2 (4.8 ГБ). (Руководство.)

11.4.2 UUPMediaCreator

UUPMediaCreator, кроссплатформенная[29] утилита, которая позволяет скачать и создать установочный ISO-образ с операционной системой Windows. Этот iso-образ создаётся по официальной методике получения образов и обновлений с официального сайта Microsoft Unified Update Platform.

Установка:

```
scoop bucket add hoilc_scoop-lemon https://github.com/hoilc/scoop-lemon
scoop install hoilc_scoop-lemon/uupmediacreator
```

[29] можно загрузить и в Linux/MacOS

Использование. Перейти в папку, где будет создаваться загрузочный iso-образ, выполнить загрузку[30] (22631.3296 соответствует 23H2):

```
UUPDownload -s Professional -v 10.0.22621.1 --r Retail -b Retail -c ni_release -t amd64 -l en-us -y 10.0.22631.3296
```

Запустить[31] сборку iso-диска (здесь надо подставить путь куда скачались файлы):

```
sudo UUPMediaConverter -l en-us -u "path-to-downloaded-files" -i Win11-23H2.iso -e Professional
```

11.4.3 Создание загрузочной флешки

Подготовить загрузочную флешку из загруженного iso-образа можно с помощью утилиты **Rufus** под Windows, или **mkusb** под Linux.

Установка *Rufus*:

```
winget install --id=Rufus.Rufus -e
```

Rufus позволяет при создании сделать *Customize Windows Installation*, а именно *запрет* на: проверку ограничений, сбор данных (телеметрию), онлайн регистрацию. При выборе типа загрузки следует выбирать *GPT/SecureBoot*, а не *MBR/Legacy* (который предназначен для очень старых компьютеров).

11.5 ODT (Office Deployment Tool) — загрузка MS Office

ODT — *Microsoft Office Deployment Tool*, *Средство Развертывания Office*, официальный инструмент для: выбора конфигурации MSOffice для загрузки, загрузки MSOffice, и установки. Официальный обзор.

Получение утилиты развёртывания *ODT*:

- Либо скачать по ссылке *officedeploymenttool.exe* и распаковать этот самораспаковывающийся архив в папку куда будет загружаться офис;
- Либо установить

  ```
  winget install --id=Microsoft.OfficeDeploymentTool -e
  ```

  и скопировать файлы из *"C:\Program Files\OfficeDeploymentTool"* в нужную папку.

[30] Потребуется до 12 ГБ свободного места
[31] UUPMediaConverter выполняется полчаса

В любом случае в папке для закачки будут необходимы два файла: *setup.exe* и *configuration-Office365-x64.xml*.

Поставляемый configuration-Office365-x64.xml скопировать в configuration.xml, и отредактировать спецификацию загружаемого офиса, инструкция.

- При выборе загружаемой версии офиса потребуется его идентификатор, который можно найти на сайте "Список идентификаторов продуктов, поддерживаемых средством развертывания Office для запуска".
- Список возможных исключений: *"Access"*, *"Bing"*, *"Excel"*, *"Groove"*, *"Lync"*, *"OneDrive"*, *"OneNote"*, *"Outlook"*, *"OutlookForWindows"*, *"PowerPoint"*, *"Publisher"*, *"Teams"*, *"Word"*.
- Опция *<Remove All="True" />* указывает перед установкой удалить все установленные версии MSOffice.

Пример конфигурации configuration.xml для загрузки дистрибутива версии *ProPlus2024Volume*, но без всех корпоративных или облачных опций, в результате в дистрибутиве должны остаться только *Word*, *Excel*, *PowerPoint*, *Visio*, *MSProject*:

```
<Configuration>
 <Add OfficeClientEdition="64" Channel="Current">
  <Product ID="ProPlus2024Volume">
   <Language ID="ru-ru" />
   <Language ID="en-us" />
   <!-- <ExcludeApp ID="Access "/> -->
   <ExcludeApp ID="Bing "/>
   <ExcludeApp ID="Groove "/>
   <ExcludeApp ID="Lync "/>
   <ExcludeApp ID="OneDrive "/>
   <ExcludeApp ID="OneNote "/>
   <ExcludeApp ID="Outlook "/>
   <ExcludeApp ID="OutlookForWindows "/>
   <ExcludeApp ID="Publisher "/>
   <ExcludeApp ID="Teams "/>
  </Product>
  <Product ID="VisioPro2024Volume">
   <Language ID="ru-ru" />
   <Language ID="en-us" />
  </Product>
  <Product ID="ProjectPro2024Volume">
   <Language ID="ru-ru" />
   <Language ID="en-us" />
  </Product>
 </Add>
 <Remove All="True" />
 <!-- <RemoveMSI All="True" /> -->
 <!-- <Display Level="None" AcceptEULA="TRUE" /> -->
```

```
 <!-- <Property Name="AUTOACTIVATE" Value="1" /> -->
</Configuration>
```

Пример конфигурации configuration.xml для загрузки дистрибутива с непосредственно выбранными *Word*, *Excel* и *PowerPoint* без облачных опций *OneDrive*:

```
<Configuration>
 <Add OfficeClientEdition="64" Channel="Current">
  <Product ID="Word2024Volume">
   <Language ID="ru-ru" />
   <Language ID="en-us" />
   <ExcludeApp ID="OneDrive "/>
  </Product>
  <Product ID="PowerPoint2024Volume">
   <Language ID="ru-ru" />
   <Language ID="en-us" />
   <ExcludeApp ID="OneDrive "/>
  </Product>
  <Product ID="Excel2024Volume">
   <Language ID="ru-ru" />
   <Language ID="en-us" />
   <ExcludeApp ID="OneDrive "/>
  </Product>
 </Add>
 <Remove All="True" />
 <!-- <RemoveMSI All="True" /> -->
 <!-- <Display Level="None" AcceptEULA="TRUE" /> -->
 <!-- <Property Name="AUTOACTIVATE" Value="1" /> -->
</Configuration>
```

Перед скачиванием изменить регион для *ODT* на *US*, иначе из-за санкций утилита не начнёт загрузку:

```
reg add "HKCU\Software\Microsoft\Office\16.0\Common\ExperimentConfigs\Ecs" /v "CountryCode" /t REG_SZ /d "std::wstri
```

Запустить загрузку:

```
setup.exe /download configuration.xml
```

Установить загруженный офис:

```
setup.exe /configure configuration.xml
```

После установки открыть любое приложение MSOffice, Word например; снизу слева будет меню *Учетная запись*, в нём выбрать *Активировать...*.

11.5.1 Office Tools Plus

Ещё есть программа с графическим интерфейсом *Office Tool Plus* https://otp.landian.vip/en-us/, в которой все вышеописанные действия можно сделать в графическом интерфейсе. Также, умеет активировать Офис с помощью KMS сервисов, расположенных в интернете.

```
scoop install extras/office-tool-plus
```

12 *Приложение:* Документация и ссылки

Windows 10/11 Guide. Including Windows Security tools, Encryption, Nextcloud, Graphics, Gaming, Virtualization, Windows Subsystem for Linux (WSL 2), Software Apps, and Resources.

Большая база заметок по работе в PowerShell и смежным командам на русском языке.

Заметки по работе с системными командами и консольными утилитами Linux для Windows пользователей.

Огромный гайд по настройке рабочего окружения: Linux, VScode, Python.

Введение в машинное обучение и искусственные нейронные сети.