Настройка Windows окружения

MSYS2

1. Установка пакетного менеджера MSYS2, который упрощает управление компонентами сборочного окружения. Последняя версия MSYS2 с поддержкой Windows 7 здесь.
   - Установку рекомендуется производить в каталог `c:/opt/MSYS64`, что позволит устанавливать в единый каталог c:/opt/ и другие пакеты, например, Python и Qt.
   - При проблемах с фокусом в Windows 11 (отсутствует клавиатурный ввод после передачи фокуса мышью окну терминала MSYS2) предлагается отключить/переключить визуальные эффекты в системе.
2. Обновление списка пакетов и основных пакетов оболочки MSYS2:

   ```
   pacman -Syu
   ```

   Справка по параметрам pacman'a доступна по ссылке.

3. После основного этапа обновления необходимо ещё раз проверить наличие дополнительных обновлений:

   ```
   pacman -Syu
   ```

   При проблемах с доступом к тому или иному зеркалу репозитория программ можно отредактировать список зеркал в файлах `/etc/pacman.d/mirrorlist.*`

4. Поиск программ в репоизтории MSYS2

   ```
   pacman -Ss <package_name>
   ```

5. Установка программ из репозитория MSYS2

   ```
   pacman -S <package_name>
   ```

Python

1. Загрузить установщик с официального сайта по ссылке. Для запуска Python 3.9+ в Windows 7 можно использовать неофициальный установщик PythonWin7.
2. Установить Python из установщика.
   - Установку рекомендуется производить в каталог `c:/opt/python/v3.xy`
   - По окончании установки рекомендуется согласится с отключением ограничения на длину переменной PATH в 260 символов.
3. Установка переменных окружения Python внутри MSYS2:
   - Добавить в файл `~/.bashrc` пути до исполняемого файла Python и каталога Scripts

     ```
     export PATH="/c/opt/Python/v3.12:/c/opt/Python/v3.12/Scripts:$PATH"
     ```

4. Ручная при необходимости установка переменных окружения в Windows:
   - Для Windows 10: Параметры → Система → О программе → Дополнительные параметры системы → Переменные среды... → Системные переменные → Path [Изменить].
     Добавить два соответствующих каталога:

```
C:\opt\Python\v3.12
C:\opt\Python\v3.12\Scripts
```

Qt over aqtinstall

Установка библиотеки Qt с помощью aqtinstall.

- При установленном Python и pip:

```
pip install -U pip
pip install aqtinstall
```

- Просмотр доступных версий:

```
aqt list-qt windows desktop
```

Долгое ожидание и получение ошибки вида `ERROR : Failed to download checksum for the file 'Updates.xml' from mirrors '['https://download.qt.io']` говорит о том, что доступ к скачиванию скорее всего запрещён по ip адресу. Для решения проблемы необходимо либо изменить запрашиваемое зеркало в настройках программы, либо производить запросы с иного ip адреса. Также на Хабре представлено решение с использованием зеркал Яндекса:

```
[aqt]
baseurl: https://qt-mirror.dannhauer.de/

[requests]
max_retries_on_checksum_error: 1
max_retries_to_retrieve_hash: 1
INSECURE_NOT_FOR_PRODUCTION_ignore_hash: True

[mirrors]
fallbacks:
    https://mirrors.ocf.berkeley.edu/qt
    https://qt.mirror.constant.com/
    https://ftp.acc.umu.se/mirror/qt.io/qtproject/
    https://qtproject.mirror.liquidtelecom.com/
    https://ftp.jaist.ac.jp/pub/qtproject
    http://ftp1.nluug.nl/languages/qt
    https://mirrors.dotsrc.org/qtproject
    https://mirror.yandex.ru/mirrors/qt.io
```

В данном случае программу запускать следующим образом

```
aqt -c aqt.cfg ...
```

- Просмотр доступных сборок библиотеки:

```
aqt list-qt windows desktop --arch 6.6.2
win64_mingw win64_msvc2019_64 win64_msvc2019_arm64 wasm_singlethread
wasm_multithread
```

- Установка Qt SDK версии 6.6.2 для MinGW со всеми доступными модулями:

```
aqt install-qt windows desktop 6.6.2 win64_mingw -m all --outputdir /c/opt/Qt
```

- Для использования Qt SDK совместно с окружением MSYS2/MinGW64, необходимо установить переменную окружения, например, с помощью `~/.bashrc`

```
export SETUPTOOLS_USE_DISTUTILS=stdlib
```

  В противном случае возможно появление ошибки `VC6.0 is not supported`

- Также необходимо установить переменную PATH:

```
export PATH="/c/opt/Qt/6.6.2/mingw_64/bin:$PATH"
```

Qt from source

Установка из исходного кода
Проблема
Идея состоит в том, чтобы использовать статическую линковку стандартных библиотек в создаваемой программе. Тогда её распространение не потребует наличия библиотек MinGW. Всё хорошо работает для программ, не использующих внешние библиотеки. Однако, если подключить внешние библиотеки, то уже им требуются стандартные библиотеки и программа не работает. Для решения этой проблемы необходимо скомпилировать библиотеку Qt с параметрами `-static-libgcc -static-libstdc++`. Инструкция по сборке доступна по ссылке.

- Установка Perl. Установщик сам прописывает все необходимые переменные окружения и Perl работоспособен сразу после установки.
- Установка Node.js. В среде MSYS2 надо добавить данные в PATH:

```
export PATH="/c/opt/Node.js:$PATH"
```

- Установка html5lib:

```
pip install html5lib
```

  В среде MSYS2 задать переменную в которых python ищет модули:

```
export PYTHONPATH="/c/opt/Python/v3.12/lib/site-packages"
```

- Установка GPerf:

```
pacman -S gperf
```

- Установка Bison:

```
pacman -S bison
```

- Установка Flex:

```
pacman -S flex
```

- Установка Clang для QDoc
  1. Скачать исходный код /длительная операция/ релиза с версией 18.1.2:

     ```
     git clone --branch llvmorg-18.1.2 https://github.com/llvm/llvm-
     project.git source
     cd source
     ```

     Список релизов доступен по ссылке.

  2. Компиляция:

     ```
     mkdir build
     cd build
     cmake ../llvm -G Ninja -DLLVM_ENABLE_PROJECTS=clang -
     DCMAKE_BUILD_TYPE=Release -DCMAKE_INSTALL_PREFIX=c:\opt\LLVM\release
     ninja -j8
     ```

  3. Проверка сборки:

     ```
     ninja -j8 check-all
     ```

  4. Установка в указанный ранее каталог `c:\opt\LLVM\release`:

     ```
     ninja install
     ```

  5. Для обнаружения установленного Clang'а в системе необходимо добавить в переменную Path путь до утилиты `llvm-config`, которая используется для получения основных каталогов Clang

     ```
     export PATH="$PATH:/c/opt/LLVM/release/bin"
     ```

     Также, чтобы сборка не была ограничена средой MSYS2, в системную переменную Path необходимо добавить каталог `C:\opt\LLVM\release\bin`.

- Клонирование Open-Source репозитория Qt:

  ```
  git clone git://code.qt.io/qt/qt5.git src
  ```

- Переключение на требуемую ветку. Метки веток можно посмотреть по ссылке:

  ```
  cd src
  git switch 6.6.2
  ```

- Подключение модулей Qt с помощью скрипта init-repository (будет загружено ~15ГБ данных):

  ```
  perl init-repository
  ```

- Конфигурирование сборки Qt SDK в среде MSYS2:

  ```
  mkdir build
  cd build
  ../configure -cmake-generator Ninja -prefix /opt/Qt/6.6.2/mingw_64-libstd-
  static -qt-zlib -qt-libb2 -no-zstd
  ```

  - Здесь же производится конфигурирование набора возможностей библиотеки путём

указания директив либо 'qt', либо 'system', например, `-qt-zlib`, `-qt-libb2`, `–no-zstd`. Подробности по ссылке.
  - Сводная информация конфигурирования сохраняется и доступна в файле `build/config.summary`.
  - При возникновении ошибок или необходимости произвести повторную конфигурацию модулей необходимо удалить файл `CMakeCache.txt` из каталога `build`.
- Сборка и установка Qt SDK:

```
cmake --build . --parallel
cmake --install .
```

MinGW

- Установка полного набора инструментов компилятора (compiler toolchains). Внимание, размер установленных файлов более 900МБ:

```
pacman -S mingw-w64-ucrt-x86_64-toolchain
```

Будет предложено установить следующие пакеты:

```
  - mingw-w64-ucrt-x86_64-binutils
  - mingw-w64-ucrt-x86_64-crt-git
  - mingw-w64-ucrt-x86_64-gcc
  - mingw-w64-ucrt-x86_64-gdb
  - mingw-w64-ucrt-x86_64-gdb-multiarch
  - mingw-w64-ucrt-x86_64-headers-git
  - mingw-w64-ucrt-x86_64-libmangle-git
  - mingw-w64-ucrt-x86_64-libwinpthread-git
  - mingw-w64-ucrt-x86_64-make
  - mingw-w64-ucrt-x86_64-pkgconf
  - mingw-w64-ucrt-x86_64-tools-git
  - mingw-w64-ucrt-x86_64-winpthreads-git
  - mingw-w64-ucrt-x86_64-winstorecompat-git
```

- Здесь же из репозитория MSYS2 можно установить утилиты сборки `make` и `cmake`:

```
pacman -S make
```

```
pacman -S cmake
```

При необходимости изменить генератор cmake, используемый по умолчанию, необходимо в файле `/.bashrc` установить соответствующую переменную среды:

```
export CMAKE_GENERATOR="MSYS Makefiles"
```

Для статической сборки стандартных библиотек C/C++ и библиотеки libwinpthread из MinGW необходимо установить некоторые параметры линковщика (подробности по ссылке):

```
export LDFLAGS="-static-libgcc -static-libstdc++ -Wl,-Bstatic,--whole-archive
```

```
-lwinpthread -Wl,-Bdynamic,--no-whole-archive"
```

- Для работы в оболочке MSYS2 добавить в файл `~/.bashrc` путь до исполняемых файлов компилятора:

```
export PATH="/c/opt/MSYS64/mingw64/bin:$PATH"
```

- При необходимости добавить в системную переменную Path операционной системы Windows дополнительное значение для обнаружения исполняемых файлов MinGW:

```
C:\msys64\mingw64\bin
```

- Здесь же логичным является установка Git:

```
pacman -S git
```

Deploying your application

Для развёртывания приложения в операционной системе Windows служит утилита `windeployqt`, которая копирует необходимые для работы приложения библиотеки из Qt и рантайма компилятора:

```
windeployqt application.exe
```

В случае, если рантайм компилятора не требуется, то используется параметр `-no-compiler-runtime`:

```
windeployqt application.exe -no-compiler-runtime
```

Подробности по ссылке https://doc.qt.io/qt-6/windows-deployment.html