



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Информатика и вычислительная техника
БАКАЛАВРСКАЯ ПРОГРАММА 09.03.01/03 Вычислительные машины, комплексы,
системы и сети

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Тип практики Эксплуатационная практика

Название
предприятия «НУК ИУ МГТУ им. Н.Э.Баумана»

Студент ИУ6-63Б

08.09.2022 А.А. Бушев
(Подпись, дата) (И.О. Фамилия)

Руководитель практики

08.09.2022 М.В. Фетисов
(Подпись, дата) (И.О. Фамилия)

Оценка отлично

2022 г.

ЗАДАНИЕ на производственную практику

по теме создание и развитие системы развертывания для системы SIMODO

Студент группы ИУ6-63Б

Бушев Антон Алексеевич

(Фамилия, имя, отчество)

Направление подготовки 09.03.01 Информатика и вычислительная техника

Бакалаврская программа 09.03.01/03 Вычислительные машины, комплексы, системы и сети

Тип практики Эксплуатационная практика

Название предприятия НУК ИУ МГТУ им. Н.Э. Баумана

Техническое задание разработать и реализовать систему автоматизированного развертывания системы
SIMODO

Оформление отчета по практике:

Отчет на 15-25 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

«нет»

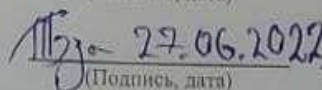
Дата выдачи задания «27» июня 2022 г.

Руководитель практики

Студент


(Подпись, дата)

М.В. Фетисов
(И.О. Фамилия)


(Подпись, дата)

А.А. Бушев
(И.О. Фамилия)

Примечание: Задание оформляется в двух экземплярах.



Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет
имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ ИНФОРМАТИКА И СИСТЕМЫ УПРАВЛЕНИЯ

КАФЕДРА КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)

НАПРАВЛЕНИЕ ПОДГОТОВКИ 09.03.01 Информатика и вычислительная техника

БАКАЛАВРСКАЯ ПРОГРАММА 09.03.01/03 Вычислительные машины, комплексы,
системы и сети

ОТЧЕТ ПО ПРОИЗВОДСТВЕННОЙ ПРАКТИКЕ

Тип практики Эксплуатационная практика

Название
предприятия «НУК ИУ МГТУ им. Н.Э.Баумана

Студент ИУ6-63Б

(Подпись, дата)

А.А. Бушев

(И.О. Фамилия)

Руководитель практики

(Подпись, дата)

М.В. Фетисов

(И.О. Фамилия)

Оценка _____

2022 г.

Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Московский государственный технический университет имени Н.Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н.Э. Баумана)

З А Д А Н И Е
на производственную практику

по теме создание и развитие системы развёртывания для системы SIMODO

Студент группы ИУ6-63Б

Бушев Антон Алексеевич

(Фамилия, имя, отчество)

Направление подготовки 09.03.01 Информатика и вычислительная техника

Бакалаврская программа 09.03.01/03 Вычислительные машины, комплексы, системы и сети

Тип практики Эксплуатационная практика

Название предприятия НУК ИУ МГТУ им. Н.Э. Баумана

*Техническое задание разработать и реализовать систему автоматизированного развёртывания системы
SIMODO*

Оформление отчета по практике:

Отчет на 15-25 листах формата А4.

Перечень графического (иллюстративного) материала (чертежи, плакаты, слайды и т.п.)

«нет»

Дата выдачи задания « 27 » июня 2022 г.

Руководитель практики

(Подпись, дата)

М.В. Фетисов

(И.О. Фамилия)

Студент

(Подпись, дата)

А.А. Бушев

(И.О. Фамилия)

Примечание: Задание оформляется в двух экземплярах.

Оглавление

Перечень сокращений и условных обозначений	4
Введение	5
Исследование видов установочных пакетов	6
Разработка автоматизированной системы развёртывания	8
Заключение	9
Список использованных источников	10
Приложение А. Листинг скрипта интеграции QtIFW в проект CMake	11
Приложение Б. Листинг скрипта непрерывного развёртывания для системы жизненного цикла DevOps Gitlab	16

Перечень сокращений и условных обозначений

ПО – программное обеспечение

RPM – Red Hat Package Manager – формат пакетов программного обеспечения, а также программа, созданная для управления этими пакетами, используемые в ряде Linux-дистрибутивов

DEB – расширение имён файлов «бинарных» пакетов для распространения и установки программного обеспечения в операционной системе проекта Debian, и других, использующих систему управления пакетами dpkg

NSIS – Nullsoft Scriptable Install System – система создания установочных программ для Microsoft Windows с открытым исходным кодом

MSI – Microsoft Installer – подсистема Microsoft Windows, обеспечивающая установку программ

QtIFW – Qt Installer Framework – фреймворк для разработки установщиков с графическим интерфейсом

DevOps – Методология автоматизации технологических процессов сборки, настройки и развёртывания программного обеспечения

Gitlab – Веб-инструмент жизненного цикла DevOps с открытым исходным кодом

Введение

На кафедре ИУ-6 МГТУ им. Баумана ведётся разработка системы SIMODO – адаптивной системы моделирования, предназначенной для упрощения создания моделей.

Система SIMODO решает проблему построения сложных моделей, которые зачастую объединяют несколько предметных областей. Для решения этой проблемы предлагается использование предметно-ориентированных языков, каждый из которых описывает ту или иную предметную область и должен быть интуитивно понятен специалистам этой области. Система SIMODO предоставляет способ унификации поддержки предметно-ориентированных языков [1].

Практическая значимость системы SIMODO заключается в возможности эффективно описывать и просчитывать сложные модели.

Система SIMODO предназначена для платформ Windows и Linux и не обладает автоматизированной системой развёртывания.

Развёртывание ПО – это все действия, которые делают программную систему готовой к использованию. Данный процесс является частью жизненного цикла программного обеспечения.

Целью данной эксплуатационной практики является создание и развитие системы развёртывания для системы SIMODO.

Задачами ставятся:

1. Исследование видов установочных пакетов приложений.
2. Выбор установочного пакета для системы SIMODO.
3. Разработка системы формирования установочного пакета.
4. Автоматизация формирования установочного пакета.

Исследование видов установочных пакетов

Целевыми платформами системы SIMODO являются Windows и Linux. Дистрибутив Linux существует великое множество. Решено было остановиться на одном из самых распространённых дистрибутиве Ubuntu версии Jammy [3], использующий установочные пакеты DEB и дистрибутиве отечественной разработки Alt Linux версии P10, использующий установочные пакеты RPM. Первый критерий к установочным пакетам – поддержка целевых платформ Ubuntu Jammy и Alt Linux P10.

Исходные тексты программ системы SIMODO расположены в репозитории системы DevOps Gitlab [2]. Для непрерывного развёртывания в данной системе используются виртуальные машины на основе Linux дистрибутивов, что определяет второй критерий для установочного пакета системы SIMODO – возможность сборки пакета на дистрибутиве Linux, в частности под целевую платформу Windows.

Если поставлять динамические библиотеки, необходимые для работы системы SIMODO, вместе с системой SIMODO, то размер установочного пакета будет значительно больше, чем установочный пакет, к которому операционная система сможет самостоятельно предоставить необходимые зависимости. Третьим и необязательным критерием выбора установочного пакета является наличие системы разрешения зависимостей.

Были выбраны и исследованы установочные пакеты RPM, DEB, NSIS, MSI, QtIFW. В таблице 1 представлены результаты исследования.

Таблица 1. Сравнение установочных пакетов

Установочный пакет	Целевые платформы	Возможность сборки под Windows в linux дистрибутиве	Наличие системы разрешения зависимостей
RPM	Linux дистрибутивы с пакетным менеджером rpm	-	+

Таблица 1 продолжение

DEB	Linux дистрибутивы с пакетным менеджером dpkg	-	+
NSIS	Windows	+	-
MSI	Windows	-	+
QtIFW	Windows, Linux	+	-

Из таблицы 1 видно, что установочный пакет QtIFW наиболее полно соответствует выставленным критериям.

Разработка автоматизированной системы развёртывания

Выбранный установочный пакет QtIFW собирается с помощью одноимённого фреймворка, который прежде, чем собирать установочный пакет, необходимо собрать из исходников [4]. Сам фреймворк требует аналогичной процедуры для статической библиотеки Qt. Для дистрибутивов Linux, в которых нет возможности установить утилиту `linuxdeployqt`, так же приходится собирать из исходников. Для сборки фреймворков под платформу Windows используется среда кросскомпиляции `mxe`, которую было решено заранее скомпилировать из исходников и разместить в репозитории Gitlab.

В приложении Б приведён листинг скрипта YAML, который используется для сборки установочного пакета в процессе непрерывного развёртывания Gitlab.

Чтобы собрать установочный пакет системы SIMODO, необходимо интегрировать данный процесс в проект системы SIMODO. В приложении А представлен листинг скрипта интеграции сборки установочного пакета в проект графического редактора системы SIMODO. С помощью переменных окружения контролируется вариант сборки под Windows или Linux.

Реализованная автоматизированная система развёртывания системы SIMODO формирует установочные пакеты системы SIMODO как артефакты конвейера системы непрерывного развёртывания Gitlab.

Заключение

В течении данной эксплуатационной практики была разработана автоматизированная система развёртывания для системы SIMODO. Автоматизированная система развёртывания формирует установочные пакеты под следующие целевые платформы: Windows, Ubuntu Jammy, Alt Linux p10.

Получаемые установочные пакеты позволяют автономно устанавливать систему SIMODO под целевые платформы.

Наработки, полученные в течении данной эксплуатационной практики могут быть применены и улучшены в новых версиях системы SIMODO, например, установка системы SIMODO из репозитория онлайн.

Список использованных источников

1. Иванова Г.С., Жильцов А.И., Фетисов М.В., Чулин Н.А., Юдин А.Е. Адаптивная система моделирования. – Автоматизация. Современные технологии, номер 11 за 2020 год, стр. 500.
2. SIMODO в репозитории МГТУ им. Н.Э. Баумана. [Электронный ресурс]. URL: <https://bmstu.codes/lxx/simodo> (дата обращения: 08.07.2022).
3. Рейтинг дистрибутивов Linux. [Электронный ресурс]. URL: <https://www.tecmint.com/top-most-popular-linux-distributions> (дата обращения: 07.07.2022).
4. Документация QtIFW. [Электронный ресурс]. URL: <https://doc.qt.io/qtinstallerframework/ifw-getting-started.html> (дата обращения: 05.07.2022).
5. Документация Gitlab. [Электронный ресурс]. URL: <https://doc.qt.io/qtinstallerframework/ifw-getting-started.html> (дата обращения: 10.07.2022).

Приложение А. Листинг скрипта интеграции QtIFW в проект CMake

```
cmake_minimum_required(VERSION 3.8)

project(edit LANGUAGES CXX)

if(${CROSS_WIN})
    enable_language("RC")
    set(ICON_RESOURCE ${CMAKE_CURRENT_SOURCE_DIR}/simodoedit.rc)
endif()

configure_file(${CMAKE_CURRENT_SOURCE_DIR}/src/MainWindow.cpp.in
               ${CMAKE_CURRENT_SOURCE_DIR}/src/MainWindow.cpp)

set(CMAKE_INCLUDE_CURRENT_DIR ON)

set(CMAKE_AUTOUIC ON)
set(CMAKE_AUTOMOC ON)
set(CMAKE_AUTORCC ON)

set(CMAKE_CXX_STANDARD 17)
set(CMAKE_CXX_STANDARD_REQUIRED ON)

find_package(
    QT NAMES Qt6 Qt5 REQUIRED
    COMPONENTS Widgets Charts
               DataVisualization 3DCore
               3DRender 3DInput 3DLogic 3DExtras
               3DAnimation)
find_package(
    Qt${QT_VERSION_MAJOR} REQUIRED
    COMPONENTS Widgets Charts
               DataVisualization 3DCore
               3DRender 3DInput
               3DLogic 3DExtras
               3DAnimation)

if(NOT (${CROSS_WIN}))
    set(TERM_WIDGET_CPP src/TermWidget.cpp)
    set(TERM_WIDGET_H src/TermWidget.h)
endif()

add_executable(
    ${PROJECT_NAME}
    ${ICON_RESOURCE}
    res/simgui.qrc
    src/BufferedListWidget.cpp
    src/BufferedListWidget.h
    src/ChartBuffer.cpp
    src/ChartBuffer.h
    src/ChartView.cpp
    src/ChartView.h
    src/cockpit/Cockpit3D.cpp
    src/cockpit/Cockpit3D.h
    src/cockpit/CockpitAviagorizont.h
    src/cockpit/CockpitBar.cpp
    src/cockpit/CockpitBar.h
    # src/cockpit/CockpitCBar.cpp
    # src/cockpit/CockpitCBar.h
    src/cockpit/CockpitHBar.h
    src/cockpit/CockpitVBar.h)
```

```

src/cockpit/CockpitVariometer.h
src/cockpit/CockpitView.cpp
src/cockpit/CockpitView.h
src/codeeditor/CodeEditor.cpp
src/codeeditor/CodeEditor.h
src/codeeditor/CodeEditorColorTheme.h
src/codeeditor/ErrorInfo.h
src/codeeditor/Highlighter.cpp
src/codeeditor/Highlighter.h
src/codeeditor/LineNumberArea.cpp
src/codeeditor/LineNumberArea.h
src/codeeditor/RuleData.cpp
src/codeeditor/RuleData.h
src/codeeditor/SpellChecker.cpp
src/codeeditor/SpellChecker.h
src/codeeditor/TextBlockData.cpp
src/codeeditor/TextBlockData.h
src/CommonConfig.cpp
src/CommonConfig.h
src/FindReplace.cpp
src/FindReplace.h
src/FindReplace.ui
src/GrammarTable.cpp
src/GrammarTable.h
src/Graph3D.cpp
src/Graph3D.h
src/Graph3DBuffer.cpp
src/Graph3DBuffer.h
src/ListReporter.cpp
src/ListReporter.h
src/ListReporterBuffer.cpp
src/ListReporterBuffer.h
src/main.cpp
src/MainWindow.cpp
src/MainWindow.h
src/MdiChild.cpp
src/MdiChild.h
src/ScriptC_NS_Chart.cpp
src/ScriptC_NS_Chart.h
src/ScriptC_NS_Cockpit.cpp
src/ScriptC_NS_Cockpit.h
src/ScriptC_NS_Q3DScatter.cpp
src/ScriptC_NS_Q3DScatter.h
src/SimodoRuler.cpp
src/SimodoRuler.h
${TERM_WIDGET_CPP}
${TERM_WIDGET_H}
src/WorkDirectory.cpp
src/WorkDirectory.h
src/WorkDirectoryGetName.cpp
src/WorkDirectoryGetName.h
src/WorkDirectoryGetName.ui
src/WorkObject.cpp
src/WorkObject.h
ts/simgui_ru_RU.ts)

target_include_directories(${PROJECT_NAME}
    PUBLIC ${CMAKE_SOURCE_DIR}/src/thirdparts/hunspell)

target_link_libraries(${PROJECT_NAME} Qt${QT_VERSION_MAJOR}::Widgets)
target_link_libraries(${PROJECT_NAME} Qt${QT_VERSION_MAJOR}::Charts)
target_link_libraries(
    ${PROJECT_NAME}
    Qt${QT_VERSION_MAJOR}::DataVisualization

```

```

Qt${QT_VERSION_MAJOR}::3DCore
Qt${QT_VERSION_MAJOR}::3DRender
Qt${QT_VERSION_MAJOR}::3DInput
Qt${QT_VERSION_MAJOR}::3DLogic
Qt${QT_VERSION_MAJOR}::3DExtras
Qt${QT_VERSION_MAJOR}::3DAnimation)
target_link_libraries(${PROJECT_NAME} SIMODOdsl)
target_link_libraries(${PROJECT_NAME} hunspell)
if(NOT (${CROSS_WIN}))
    target_link_libraries(${PROJECT_NAME} "qtermwidget5")
endif()
# target_link_libraries(${PROJECT_NAME} "stdc++fs")

set(PROJECT_OUTPUT_NAME "simodoedit")
set_target_properties(
    ${PROJECT_NAME}
    PROPERTIES MACOSX_BUNDLE_GUI_IDENTIFIER my.example.com
                MACOSX_BUNDLE_BUNDLE_VERSION ${SIMODO_VERSION}
                MACOSX_BUNDLE_SHORT_VERSION_STRING
                ${SIMODO_VERSION_MAJOR}.${SIMODO_VERSION_MINOR}
                MACOSX_BUNDLE TRUE
                WIN32_EXECUTABLE TRUE
                RUNTIME_OUTPUT_DIRECTORY ${CMAKE_SOURCE_DIR}/bin
                OUTPUT_NAME ${PROJECT_OUTPUT_NAME})

set(RELEASE_STRIP $<IF:$<CONFIG:Release>,$<CMAKE_STRIP>,>)
if(NOT (${RELEASE_STRIP}) STREQUAL "" AND (${CROSS_WIN}))
    set(PON_EXT .exe)
endif()
set(RELEASE_STRIP_ARGS

$<IF:$<CONFIG:Release>,"${CMAKE_SOURCE_DIR}/bin/${PROJECT_OUTPUT_NAME}${PON_EXT}"
,>
)
add_custom_command(
    TARGET ${PROJECT_NAME}
    POST_BUILD
    COMMAND ${RELEASE_STRIP} ARGS ${RELEASE_STRIP_ARGS})

set(CPACK_GENERATOR "IFW")

set(CPACK_PACKAGE_NAME ${PROJECT_OUTPUT_NAME})
set(CPACK_PACKAGE_VERSION ${SIMODO_VERSION})
set(CPACK_PACKAGE_RELEASE ${SIMODO_VERSION_BUILD})
set(CPACK_PACKAGE_CONTACT "Michael Fetisov")
set(CPACK_PACKAGE_VENDOR "BMSTU")
set(CPACK_PACKAGE_FILE_NAME
    "${CPACK_PACKAGE_NAME}-${CPACK_PACKAGE_VERSION}-
${CPACK_PACKAGE_RELEASE}.${CMAKE_SYSTEM_PROCESSOR}"
)
set(CPACK_IFW_COMPONENT_INSTALL ON)

set(CPACK_IFW_PACKAGE_NAME "SIMODO edit")
set(CPACK_IFW_PACKAGE_START_MENU_DIRECTORY "Simodo")
set(CPACK_IFW_PACKAGE_TITLE "SIMODO edit Installer")
set(CPACK_IFW_PACKAGE_START_MENU_DIRECTORY "SIMODO")
set(CPACK_IFW_TARGET_DIRECTORY "@HomeDir@/Simodo/SimodoEdit")
set(CPACK_IFW_PACKAGE_ICON "${CMAKE_SOURCE_DIR}/installer/logo-01.ico")
set(CPACK_IFW_PACKAGE_WINDOW_ICON "${CMAKE_SOURCE_DIR}/installer/logo-01.png")
set(CPACK_IFW_PACKAGE_LOGO "${CMAKE_SOURCE_DIR}/installer/logo-01-128.png")
set(CPACK_IFW_PACKAGE_WIZARD_STYLE "Modern")
set(CPACK_IFW_PACKAGE_ALLOW_NON_ASCII_CHARACTERS ON)
set(CPACK_IFW_PRODUCT_URL "https://bmstu.codes/lxx/simodo/stars")
set(CPACK_IFW_ROOT "${CMAKE_SOURCE_DIR}/../../Qt/QtIFW")

```

```

include(CPackIFW)

cpack_ifw_configure_component(
    simodo-edit FORCED_INSTALLATION
    NAME simodo.edit
    DISPLAY_NAME "SIMODO edit"
    DESCRIPTION "Install SIMODO edit."
    SCRIPT ${CMAKE_SOURCE_DIR}/installer/installscript.qs
    LICENSES "MIT License" ${CMAKE_SOURCE_DIR}/installer/LICENSE.MIT
    DEFAULT true
    DEPENDS support)

cpack_ifw_configure_component(
    support FORCED_INSTALLATION VIRTUAL
    NAME support
    DEFAULT true)

if(${CROSS_WIN})
    install(
        PROGRAMS ${CMAKE_SOURCE_DIR}/bin/${PROJECT_OUTPUT_NAME}.exe
        DESTINATION bin
        COMPONENT simodo-edit)

    install(
        DIRECTORY ${CMAKE_SOURCE_DIR}/bin/
        DESTINATION bin
        COMPONENT simodo-edit
        PATTERN ${PROJECT_OUTPUT_NAME}.exe EXCLUDE)
else()
    install(
        CODE "
            execute_process(
                COMMAND rm -rf tmp
                WORKING_DIRECTORY ${CMAKE_SOURCE_DIR}
                COMMAND_ERROR_IS_FATAL ANY
            )
            execute_process(
                COMMAND mkdir -p tmp/ifw-installer/bin
                WORKING_DIRECTORY ${CMAKE_SOURCE_DIR}
                COMMAND_ERROR_IS_FATAL ANY
            )
            execute_process(
                COMMAND cp bin/${PROJECT_OUTPUT_NAME} tmp/ifw-
installer/bin/${PROJECT_OUTPUT_NAME}
                WORKING_DIRECTORY ${CMAKE_SOURCE_DIR}
                COMMAND_ERROR_IS_FATAL ANY
            )
            execute_process(
                COMMAND linuxdeployqt ${PROJECT_OUTPUT_NAME} -bundle-non-qt-libs -
extra-
plugins=iconengines,platformthemes,renderers,geometryloaders,kf5,kf5/kio,styles
-unsupported-allow-new-glibc
                WORKING_DIRECTORY ${CMAKE_SOURCE_DIR}/tmp/ifw-installer/bin
                COMMAND_ERROR_IS_FATAL ANY
            )
        "
        COMPONENT simodo-edit)

    install(
        PROGRAMS ${CMAKE_SOURCE_DIR}/tmp/ifw-installer/bin/${PROJECT_OUTPUT_NAME}
        DESTINATION bin
        COMPONENT simodo-edit)

```



```

install(
  DIRECTORY ${CMAKE_SOURCE_DIR}/tmp/ifw-installer/
  DESTINATION .
  COMPONENT simodo-edit
  PATTERN bin/${PROJECT_OUTPUT_NAME} EXCLUDE)
endif()

if(${CROSS_WIN})
install(
  CODE "
    execute_process(
      COMMAND rm -rf tmp/data
      WORKING_DIRECTORY ${CMAKE_SOURCE_DIR}
      COMMAND_ERROR_IS_FATAL ANY
    )
    execute_process(
      COMMAND mkdir -p tmp
      WORKING_DIRECTORY ${CMAKE_SOURCE_DIR}
      COMMAND_ERROR_IS_FATAL ANY
    )
    execute_process(
      COMMAND cp -RL data tmp/data
      WORKING_DIRECTORY ${CMAKE_SOURCE_DIR}
      COMMAND_ERROR_IS_FATAL ANY
    )
  "
  COMPONENT support)

install(
  DIRECTORY "${CMAKE_SOURCE_DIR}/tmp/data"
  DESTINATION "."
  COMPONENT support)

else()
install(
  DIRECTORY "${CMAKE_SOURCE_DIR}/data"
  DESTINATION "."
  COMPONENT support)
endif()

if(${CROSS_WIN})
install(
  FILES "${CMAKE_SOURCE_DIR}/installer/icons/Simodo-simodoedit.ico"
  DESTINATION "icons"
  COMPONENT support)
else()
install(
  DIRECTORY "${CMAKE_SOURCE_DIR}/installer/icons/"
  DESTINATION "icons"
  COMPONENT support
  PATTERN *.ico EXCLUDE)

endif()

install(
  DIRECTORY "${CMAKE_SOURCE_DIR}/test/examples"
  DESTINATION "."
  COMPONENT support)

include(CPack)

```

Приложение Б. Листинг скрипта непрерывного развёртывания для системы жизненного цикла DevOps Gitlab

```
image: alt:p10
stages:
  - build
  - test
  - coverage
  - delivery
.remove_utils: &remove_utils
  - rm -rf ../../Qt ../../linuxdeployqt ../../cross /opt/mxe
.save_dir_to_env_var: &save_dir_to_env_var
  - __GITLAB_CI_last_location_dir__=$PWD
.return_to_saved_dir: &return_to_saved_dir
  - cd $__GITLAB_CI_last_location_dir__

.build_deploylinuxqt: &build_deploylinuxqt
  # Сборка linuxdeployqt
  - git clone https://github.com/probonopd/linuxdeployqt.git
  - cd linuxdeployqt && qmake && make && export PATH=$PWD/bin:$PATH && cd ..

.build_ifw_with_static_qt: &build_ifw_with_static_qt
  # Сборка Qt5 из исходников
  - mkdir Qt && cd Qt && mkdir Qt5-static
  - git clone git://code.qt.io/qt/qt5.git Qt5-source
  - cd Qt5-source
  - git checkout ifw-5.15.2
  - ./init-repository --module-
subset=qtbase,qtdeclarative,qttools,qttranslations,qtwayland,qtqmlextras
  - sed -i 's/#ifdef __cplusplus/#ifdef __cplusplus\n# include <limits>/'
qtbase/src/corelib/global/qglobal.h
  - ./configure -prefix $PWD/../Qt5-static -opensource -confirm-license -release
-static -static-runtime -platform linux-clang -accessibility -qt-zlib -qt-libpng
-qt-libjpeg -qt-pcre -no-glib -no-cups -no-sql-sqlite -no-qml-debug -no-opengl -
no-egl -no-sm -no-icu -nomake examples -nomake tests -no-libudev
  - make -j8 module-qtbase module-qtdeclarative module-qttools module-
qttranslations module-qtwayland module-qtqmlextras
  - make install
  - cd ../Qt5-static
  - export PATH=$PWD/bin:$PATH
  - cd ../../
  # Сборка IFW из исходников
  - cd Qt
  - git clone https://github.com/qtproject/installer-framework.git QtIFW
  - cd QtIFW
  - echo "CONFIG += libarchive"'\n' > installerfw.pri.tmp
  - cat installerfw.pri >> installerfw.pri.tmp
  - rm -f installerfw.pri
  - cp installerfw.pri.tmp installerfw.pri
  - qmake && make -j8
  - cd ../../
  - PATH=$(echo $PATH | sed 's/.*Qt5-static\/bin:\/\/')
# Устанавливаем общие программы для сборки
.download_build_programs: &download_build_programs
  # Обновляем список пакетов в репозиториях
  - apt-get -y update
  # Устанавливаем g++ и Qt
  - apt-get -y install qt-creator
  - gcc --version
  # Устанавливаем пакеты
  - apt-get -y install ninja-build clang boost-devel cmake make
  - cmake --version
```

```

# Устанавливаем библиотеки для edit
- apt-get -y install qt5-charts-devel qt5-datavis3d-devel libqtermwidget-devel
qt5-3d qt5-3d-devel
# Задаём команды, которые будут выполнены до прогона скриптов
before_script:
# Создаём рабочие каталоги
- ./configure
# - export DEBIAN_FRONTEND=noninteractive
# - export TZ=Europe/Moscow

deploy-alt:
  dependencies: []
  stage: delivery
  script:
    - *remove_utils
    - *download_build_programs
    # Устанавливаем пакеты
    - apt-get -y install rpm-build rpm-build-ninja sudo linuxdeployqt
    - apt-get -y install git qt5-tools-devel bzip-devel zlib-devel liblzma-devel
libexpat-devel
    - apt-get -y install qt5-declarative-devel
    - apt-get -y install fontconfig-devel libfreetype-devel libX11-devel libxcb-
devel libxcbutil-devel
    - apt-get -y install libxcbutil-icccm libxcbutil-icccm-devel libxcbutil-image-
devel libqt5-xcbqpa
    - apt-get -y install libxcb-render-util-devel libxcbutil-cursor-devel
libxcbutil-keysyms-devel libxcbutil-xrm-devel
    - apt-get -y install libxcbwin-devel libXext-devel libXfixes-devel libXi-devel
libXrender-devel libxcb libXrandr-devel
    - apt-get -y install libshape-devel libXinerama-devel libxkbcommon-devel
libxkbcommon-x11-devel
    - apt-get -y install libxshmfence-devel libGLX libGLX-mesa
    # Плагины для ifw
    - apt-get -y install libkf5iconthemes kf5-plasma-integration kde5-kio-extras
plasma5-oxygen plasma5-breeze
    - *save_dir_to_env_var
    # Сборка утилит и фреймворков
    - cd ../..
    - *build_ifw_with_static_qt
    # Возврат в директорию проекта
    - *return_to_saved_dir
    - ./gomake
    - ./godeploy
    - *remove_utils
    # Установка gitlab-release-cli
    # - apt-get -y install curl
    # - curl --location --output /usr/local/bin/release-cli
"https://gitlab.com/api/v4/projects/gitlab-org%2Frelease-
cli/packages/generic/release-cli/latest/release-cli-linux-amd64"
    # - sudo chmod +x /usr/local/bin/release-cli
    # - release-cli -v
    # release:
    #   tag_name: '0.2.0'
    #   name: 'Release 0.2.0'
    #   description: 'Release created using the release-cli.'
  artifacts:
    paths:
      - packages/*
    expire_in: 120 days
  only:
    - prod

deploy-ubuntu:
  dependencies: []

```

```

stage: delivery
image: ubuntu:jammy
script:
- *remove_utils
# Устанавливаем пакеты
- apt update
- apt -y install build-essential libgl1-mesa-dev cmake ninja-build clang
- apt -y install qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools
- apt -y install libqt5charts5-dev libqt5datavisualization5-dev qt3d5-dev
- apt -y install libboost-all-dev libqtermwidget5-0-dev patchelf
- apt -y install git g++
- apt -y install libghc-bzlib-dev liblzma-dev
- apt -y install qtdeclarative5-dev libxkbcommon-dev
- apt -y install libqt5waylandclient5-dev libqt5waylandclient5
libqt5waylandcompositor5-dev libqt5waylandcompositor5
- apt -y install libxkbcommon-x11-dev libxcb-xkb-dev libxcb-icccm4-dev libxcb-
icccm4
- apt -y install xutils-dev xutils xdg-utils x11-utils libxcb-util0-dev
libxcb-util-dev libxcb-util1 libx11-xcb-dev libx11-xcb1 libxcb1-dev libxcb1 xcb
- apt -y install libxcb-shm0 libxcb-shm0-dev libclang1 libclang-dev xwayland
- apt -y install $(apt list | grep libwayland | sed 's/\././g')
- apt -y install $(apt list | grep libqt5wayland | sed 's/\././g')
- apt -y install $(apt list | grep qtwayland | sed 's/\././g')
- apt -y install $(apt list | grep libegl | sed 's/\././g')
- apt -y install $(apt list | grep libegl | sed 's/\././g')
- apt -y install libfontconfig1-dev libfreetype6-dev libx11-dev libx11-xcb-dev
libxext-dev libxfixes-dev libxi-dev libxrender-dev libxcb1-dev libxcb-glx0-dev
- apt -y install libxcb-keysyms1-dev libxcb-image0-dev libxcb-shm0-dev libxcb-
icccm4-dev libxcb-sync0-dev libxcb-xfixes0-dev libxcb-shape0-dev libxcb-randr0-
dev
- apt -y install libxcb-render-util0-dev libxcb-xinerama0-dev libxkbcommon-dev
libxkbcommon-x11-dev
- apt -y install qt5-style-plugins qt5-styles-ukui qt5-style-kvantum qt5-
style-kvantum-themes qt5-style-kvantum-l10n
# Плагины ifw
- apt -y install libkf5iconthemes5 plasma-integration kio-extras plasma-theme-
oxygen breeze qt3d-defaultgeometryloader-plugin
- *save_dir_to_env_var
# Сборка утилит и фреймворков
- cd ../..
- *build_deploylinuxqt
- *build_ifw_with_static_qt
# Возврат в директорию проекта
- *return_to_saved_dir
- ./gomake
- ./godeploy
- *remove_utils
# Установка gitlab-release-cli
# - apt-get -y install curl
# - curl --location --output /usr/local/bin/release-cli
"https://gitlab.com/api/v4/projects/gitlab-org%2Frelease-
cli/packages/generic/release-cli/latest/release-cli-linux-amd64"
# - sudo chmod +x /usr/local/bin/release-cli
# - release-cli -v
# release:
#   tag_name: '0.2.0'
#   name: 'Release 0.2.0'
#   description: 'Release created using the release-cli.'
artifacts:
  paths:
    - packages/*
  expire_in: 120 days
only:
- prod

```

```

deploy-windows:
  dependencies: []
  stage: delivery
  image: ubuntu:jammy
  script:
    - *remove_utils
    - apt -y update && apt -y install git cmake ninja-build libxcb-glx0 libx11-
xcb1 libxcb-icc4 libxcb-image0 libxcb-keysyms1 libxcb-randr0 libxcb-render-
util0 libxcb-shape0 libxcb-sync1 libxcb-xfixes0 libxcb-xinerama0 libxcb-xkb1
libxkbcommon-x11-0 libegl1 libpcre2-16-0 libgl1
    # Устанавливаем пакеты
    # - apt update
    # - apt -y install build-essential libgl1-mesa-dev cmake ninja-build clang
    # Плагины ifw
    # - apt -y install libkf5iconthemes5 plasma-integration kio-extras plasma-
theme-oxygen breeze qt3d-defaultgeometryloader-plugin
    - *save_dir_to_env_var
    # Скачивание утилит и фреймворков
    - cd ../../
    - git clone http://bmstu.codes/a.bushev/cross-linux-win-boost-qt5-qtifw.git
cross
    - __cross_dir__=$PWD/cross
    - cd /opt
    - tar -zxf $__cross_dir__/mxe.tar.gz
    - chmod -R +x .
    - export PATH=$PWD/mxe/usr/bin:$PATH
    # Возврат в директорию проекта
    - *return_to_saved_dir
    - cd bin
    - tar -zxf $__cross_dir__/runtime.tar.gz
    - cd ../../..
    - mkdir -p Qt/QtIFW/bin
    - cd Qt/QtIFW/bin
    - echo \#!/bin/bash > binarycreator
    - echo /opt/mxe/usr/bin/x86_64-pc-linux-gnu-binarycreator -t
/opt/mxe/usr/x86_64-w64-mingw32.static/qt5/bin/installerbase.exe \$_@ >>
binarycreator
    - chmod +rx binarycreator
    - *return_to_saved_dir
    - ./crossmake
    - ./godeploy
    - cd packages
    - for i in *; do mv "$i" "${i}/run/exe"; done;
    - cd ..
    - *remove_utils
    # Установка gitlab-release-cli
    # - apt-get -y install curl
    # - curl --location --output /usr/local/bin/release-cli
"https://gitlab.com/api/v4/projects/gitlab-org%2Frelease-
cli/packages/generic/release-cli/latest/release-cli-linux-amd64"
    # - sudo chmod +x /usr/local/bin/release-cli
    # - release-cli -v
    # release:
    #   tag_name: '0.2.0'
    #   name: 'Release 0.2.0'
    #   description: 'Release created using the release-cli.'
artifacts:
  paths:
    - packages/*
  expire_in: 120 days
only:
  - prod

```