

Санкт-Петербургский политехнический университет Петра Великого  
Институт компьютерных наук и технологий  
**Высшая школа программной инженерии**

## **КУРСОВАЯ РАБОТА**

“Преобразование форматов видео и аудио файлов”

по дисциплине «Архитектура ЭВМ»

Выполнил  
студент гр.  
3530904/00005

Самойленко К.И.

Руководитель /  
преподаватель

Молодяков С.А.

Санкт-Петербург  
2022

# 1. Описание программы

Программа представляет собой исполняемый файл на языке программирования высокого уровня Python. Основная ее цель - преобразование форматов видео и аудио файлов.

Для запуска программы требуется установленная библиотека FFMPEG, как раз-таки с ее помощью и происходит вся “магия” конвертации файлов. Также требуется библиотеку PyQt5 для корректной работы пользовательского интерфейса.

После запуска программы пользователю становится доступен интерфейс для работы с программой, изначально на экране продемонстрирована краткая инструкция по использованию:

- 1) Программа способна преобразовывать видео -> видео;
- 2) Программа способна преобразовывать аудио -> аудио;
- 3) Программа способна преобразовывать видео -> аудио;  
(С потерей визуальных данных)
- 4) Программа способна преобразовывать аудио -> видео;  
(Без добавления визуальных данных)
- 5) Доступные форматы видео:  
asf | avi | mp4 | mpg | mov | m4v | wmv | dvd | mpeg;
- 6) Доступные форматы аудио:  
mp3 | wav | wma | aiff | au | m4a | mp4;
- 7) Пользователю на выбор предоставляется два варианта исполнения программы:
  - а) Преобразование одного файла;  
Требуемый пользовательский ввод: имя файла, расширение входного файла, расширение выходного файла
  - б) Преобразование файлов из текущего каталога;  
Требуемый пользовательский ввод: расширение входных файлов, расширение выходных файлов
- 8) Вся работа пользователя с программой происходит через графический интерфейс.

## 2. Алгоритм работы программы

1. Импортирование используемых библиотек:  
FFMPEG - основа программы, преобразование файлов  
PyQt5 - используется для генерации пользовательского интерфейса

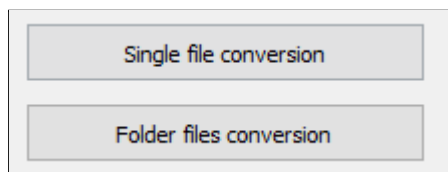
OS - используется для получения файлов в текущем каталоге  
SYS - используется для корректной работы программы с интерфейсом

2. Печать краткого описания функционала (описан выше):

```
You can convert
Video -> video
Audio -> audio
Video -> audio | lose of video data
Audio -> video | videofile without video data

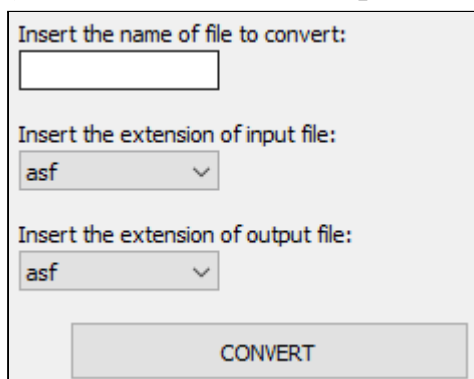
Available in/out video formats
asf | avi | mp4 | mpg | mov | m4v | wmv | dvd | mpeg
Available in/out audio formats
mp3 | wav | wma | aiff | au | m4a | mp4
```

3. Предоставлению пользователю выбора варианта программы:  
(ввод происходит через нажатие соответствующей кнопки)



A screenshot of a software interface showing two buttons stacked vertically. The top button is labeled "Single file conversion" and the bottom button is labeled "Folder files conversion". Both buttons are light gray with black text.

4. В зависимости от выбранного варианта срабатывает:  
а) Одиночное преобразование файла  
(в зависимости от выбранных вариантов из всплывающего списка)



A screenshot of a file conversion dialog box. It contains three input fields: "Insert the name of file to convert:" (a text box), "Insert the extension of input file:" (a dropdown menu with "asf" selected), and "Insert the extension of output file:" (a dropdown menu with "asf" selected). At the bottom right is a button labeled "CONVERT".

б) Преобразования всех файлов в текущем каталоге с указанным расширением:  
(в зависимости от выбранных вариантов из всплывающего списка)

С помощью библиотеки `os`, а именно функции `walk(...)`, мы можем получить кортеж из (пути, папки, файлов) и использовать файлы для перебора в цикле, однако `walk(...)` требует каталог - его мы получаем вызывая функцию, возвращающую текущий каталог, `getcwd()` из той же библиотеки.

Таким образом, мы элегантно перебираем в цикле каждый файл текущего каталога.

5. Основными функциями в программе являются функции `convert(...)` и `folder_convert(...)`

Внутри функций происходит конкатенация строк для получения полных имен файлов и их передачи внутрь функций FFMPEG.

`ffmpeg.input(...)` - принимает имя файла, который будет преобразован

`ffmpeg.output(...)` - принимает имя файла, который будет на выходе

`ffmpeg.run()` - запускает цепочку вызовов функций для преобразования

Два варианта вызова преобразований:

а) (

`ffmpeg`

`.input(ifilename)`

`.output(ofilename)`

`.run()`

)

б) `ffmpeg.input(ifilename).output(ofilename).run()`

В данной программе представлен второй вариант.

6. При некорректном вводе имени файла текущего каталога программа экстренно завершается с ошибкой.

### 3. Список используемых функций

#### Библиотека FFMPEG

input(...)	Передача файла в поток FFMPEG
output(...)	Указание имени выходного файла
run(...)	Запуск процесса FFMPEG с параметрами, скомпилированными для заданного потока

#### Библиотека OS

walk(...)	Генерирует имена файлов в дереве каталогов, обходя дерево сверху вниз
getcwd()	Возвращает строку представляющую текущий каталог

#### Библиотека PyQt5

QApplication(...)	Создать приложение с пользовательским интерфейсом
Window()	Создать окно программы
setWindowTitle(...)	Назвать окно программы
setGeometry(...)	Выставить изначальный размер окна программы
QLabel(...)	Создать элемент с текстом
setText(...)	Изменить текст на элементе
move(...)	Переместить элемент

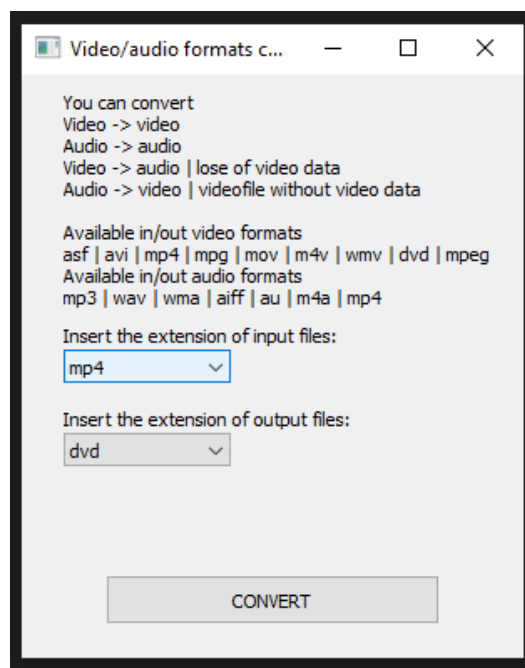
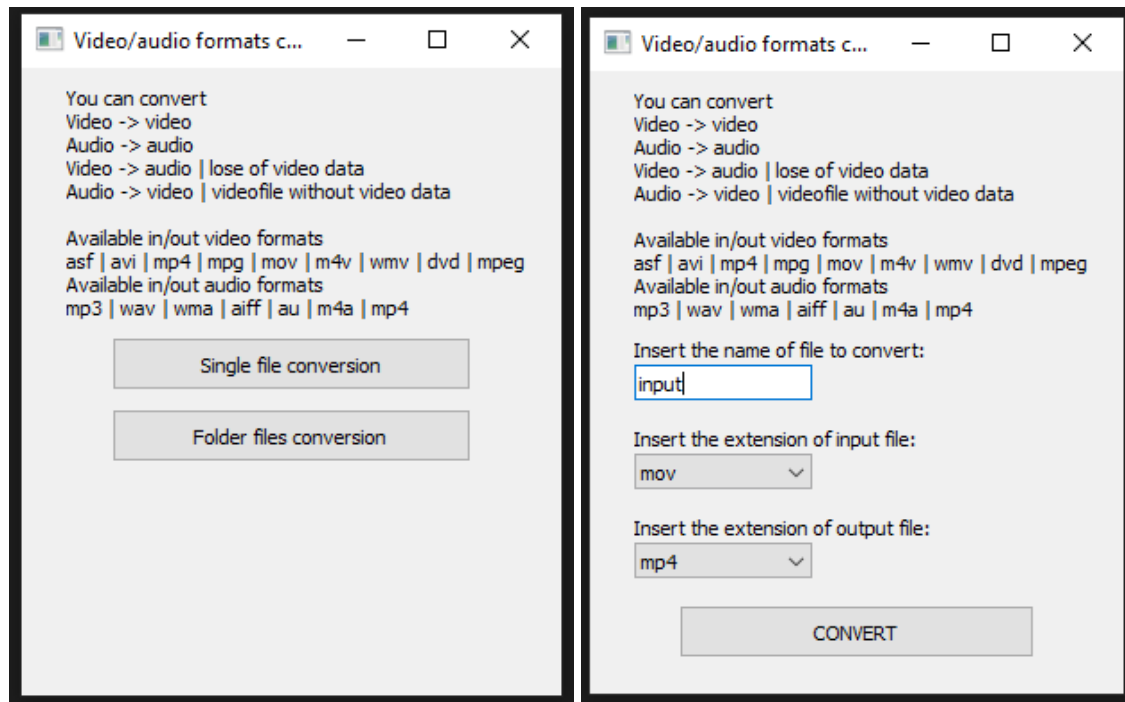
adjustSize(...)	Подстроить размер элемента под размер текста на нем
QLineEdit(...)	Создать элемент поля для ввода
QComboBox(...)	Создать элемент выпадающего списка
addItem(...)	Добавить варианты в выпадающий список
hide(...)	Скрыть элемент
QPushButton(...)	Создать элемент кнопки
setFixedWidth(...)	Зафиксировать размер элемента
clicked.connect(...)	Соединить событие нажатия кнопки с функцией
resize(...)	Изменить размер элемента
text(...)	Получить текст из QLineEdit
currentText(...)	Получить текст из QComboBox

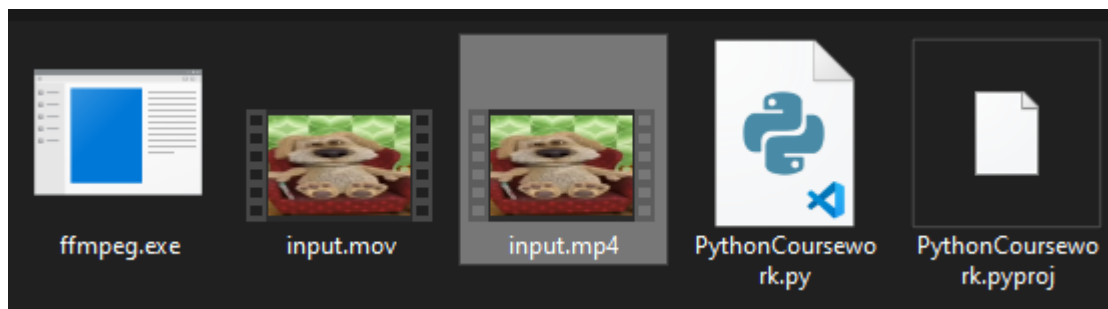
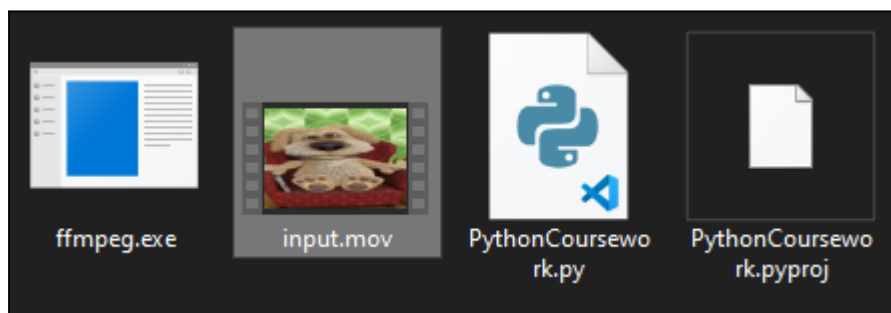
## 4. Вывод

С помощью мощной и универсальной библиотеки FFMPEG была создана простая, но при этом безумно эффективная программа, позволяющая быстро конвертировать аудио/видео файл или даже каталог файлов в нужный пользователю формат.

Для упрощения работы пользователя с программой был реализован графический интерфейс на основе библиотеки PyQt5.

## 5. Демонстрация работы





## 6. Листинг программы

```
import ffmpeg
import os
from PyQt5 import QtWidgets
from PyQt5.QtWidgets import QApplication, QMainWindow
import sys

class Window(QMainWindow):
    def __init__(self):
        super(Window, self).__init__()

        self.setWindowTitle('Video/audio formats converter')
        self.setGeometry(300, 250, 300, 350)

        self.main_text = QtWidgets.QLabel(self)
        self.main_text.setText('You can convert\nVideo -> video\nAudio -> audio\nVideo ->
audio | lose of video data\n' +
                                'Audio -> video | videofile without video
data\n\nAvailable in/out video formats\n' +
                                'asf | avi | mp4 | mpg | mov | m4v | wmv | dvd |
mpeg\nAvailable in/out audio formats\n' +
                                'mp3 | wav | wma | aiff | au | m4a | mp4\n')

        self.main_text.move(25, 10)
        self.main_text.adjustSize()

        self.name_text = QtWidgets.QLabel(self)
```



```

self.iextension_text = QtWidgets.QLabel(self)
self.oextension_text = QtWidgets.QLabel(self)

self.format_array = ['asf', 'avi', 'mp4', 'mpg', 'mov', 'm4v', 'wmv', 'dvd', 'mpeg', 'mp3',
'wav', 'wma', 'aiff', 'au', 'm4a']
self.name_line = QtWidgets.QLineEdit(self)
self.iextension_cb = QtWidgets.QComboBox(self)
self.oextension_cb = QtWidgets.QComboBox(self)
self.iextension_cb.addItem(self.format_array)
self.oextension_cb.addItem(self.format_array)

self.name_line.hide()
self.iextension_cb.hide()
self.oextension_cb.hide()

self.btn1 = QtWidgets.QPushButton(self)
self.btn1.move(50, 150)
self.btn1.setText('Single file conversion')
self.btn1.setFixedWidth(200)

self.btn2 = QtWidgets.QPushButton(self)
self.btn2.move(50, 190)
self.btn2.setText('Folder files conversion')
self.btn2.setFixedWidth(200)

self.btn3 = QtWidgets.QPushButton(self)
self.btn3.hide()
self.btn3.move(50, 300)
self.btn3.setText('CONVERT')
self.btn3.setFixedWidth(200)

self.input_filename = '0'
self.input_extension = '0'
self.output_extension = '0'
self.ifilename = '0'
self.ofilename = '0'

self.btn1.clicked.connect(self.variant1)
self.btn2.clicked.connect(self.variant2)

```

```

def variant1(self):

```

```

    self.btn1.hide()
    self.btn2.hide()

    self.name_text.setText('Insert the name of file to convert:')
    self.name_text.move(25, 150)
    self.name_text.adjustSize()

```

```

self.iextension_text.setText('Insert the extension of input file:\n')
self.iextension_text.move(25, 200)
self.iextension_text.adjustSize()

self.oextension_text.setText('Insert the extension of output file:')
self.oextension_text.move(25, 250)
self.oextension_text.adjustSize()

self.name_line.move(25, 165)
self.name_line.resize(100, 20)
self.name_line.show()

self.iextension_cb.move(25, 215)
self.iextension_cb.resize(100, 20)
self.iextension_cb.show()

self.oextension_cb.move(25, 265)
self.oextension_cb.resize(100, 20)
self.oextension_cb.show()

self.btn3.show()
self.btn3.clicked.connect(self.convert)

```

**def variant2(self):**

```

self.btn1.hide()
self.btn2.hide()

self.iextension_text.setText('Insert the extension of input files:\n')
self.iextension_text.move(25, 150)
self.iextension_text.adjustSize()

self.oextension_text.setText('Insert the extension of output files:')
self.oextension_text.move(25, 200)
self.oextension_text.adjustSize()

self.iextension_cb.move(25, 165)
self.iextension_cb.resize(100, 20)
self.iextension_cb.show()

self.oextension_cb.move(25, 215)
self.oextension_cb.resize(100, 20)
self.oextension_cb.show()

self.btn3.show()
self.btn3.clicked.connect(self.folder_convert)

```

```

#         Function of conversion
def convert(self):
    self.input_filename = self.name_line.text()
    self.input_extension = self.iextension_cb.currentText()
    self.output_extension = self.oextension_cb.currentText()

    ifilename = self.input_filename + '.' + self.input_extension
    ofilename = self.input_filename + '.' + self.output_extension
    print('Starting of conversion ' + self.ifilename + '...')
    ffmpeg.input(ifilename).output(ofilename).run()
    print('End of conversion')
    exit()

def folder_convert(self):
    self.input_extension = self.iextension_cb.currentText()
    self.output_extension = self.oextension_cb.currentText()
    for path, folder, files in os.walk(os.getcwd()):
        for file in files:
            if file.endswith(self.input_extension):
                ffmpeg.input(file).output(str(file).split('.')[0] + '.' +
self.output_extension).run()
            else:
                pass

    exit()

def application():
    app = QApplication(sys.argv)
    window = Window()

    window.show()
    sys.exit(app.exec_())

if __name__ == "__main__":
    application()

```