

Многоголосое озвучивание текста

Введение

Перед запуском скриптов ознакомьтесь с текстом этого руководства и ознакомьтесь с вложенными примерами.

Создавалось и проверялось на Mint 19.2.

Должны быть установлены текстовый редактор xed, ffmpeg, SOX, sed, медиа плеер MPV, xclip.

На свежей системе Mint для установки программ можно выполнить:

```
apt-get install mpv && apt-get install xclip && apt-get install sox && apt-get install ffmpeg
```

В скрипте **1-Say-this-text** сохранены запросы к Yandex Cloud, в которые могут быть включены произвольные тексты. Поддерживаются обычные текстовые запросы и запросы в формате SSML.

Для того, чтобы скрипт увидел текст, достаточно его выделить мышью, а затем запустить скрипт, двойным кликом или перетаскиванием в окно терминала, в котором открыт рабочий каталог (правой кнопкой на пустом месте в каталоге, открыть в Терминале).

Но прежде, необходимо внести значения строк FOLDER_ID и OAuth_TOKEN (см. Авторизация).

Такая операция приведёт к озвучиванию текста голосом по умолчанию. При этом в каталоге появятся файлы

1. **data_time_clip.mp3** - аудиозапись выделенного ранее и озвученного текста.

2. **clip.txt**, **CONtROL.txt** и **a.txt**. Это служебные файлы, которые будут перезаписываться при каждом новом запуске скрипта.

Для управления процессом озвучивания необходимо подготовить текст, применив ключи перед текстовыми строками.

Подготовка текста

Исходный текст необходимо поместить в текстовый файл с именем без пробелов, например, **text.txt**. Открыв терминал из рабочего каталога, перетащите в его окно файл **2-create_work_file** и **text.txt**.

В рабочем каталоге появится файл **data_time_work_file.txt**, в котором будет находиться текст, перед каждым абзацем которого будет установлен стандартный ключ:

```
{{ 1 0.9 3 0 -3 0 0 0 [ Комментарий ] }}
```

Вместо слова «комментарий» можно вносить произвольные комментарии в процессе подготовки текста к озвучиванию.

Авторский текст и диалоги персонажей следует снабдить соответствующими ключами. В файле **08-11-2019_19-32-46_work_file.txt** можно увидеть ключи для четырёх голосов, которые обозначены первой цифрой ключа.

Все компоненты ключа:

голос, темп речи, эмоция, язык, громкость, эффект, вид эффекта, кодирование SSML.

Голоса: 0-zhenya 2-oksana 3-jane 7-omazh 1-ermil 4-zahar 5-nastya 9-sasha 8-alyssUS 6-kostya 10-silaerkanTR 11-erkanyavas (TR) 12-nick (US) 13-filipp 14-alena. Последние два «premium» хорошо управляются через SSML.

Темп речи: от 0,1 до 3,0.

Эмоция: 1-good, 3-neutral, 2-evil. Лучше всего выражена для голоса 3-jane. Для остальных голосов

надо проверять каждый случай. Для «premium» эмоции не применяются, - эти голоса сами определяют эмоцию по контексту.

Язык текста: 0-ru-RU, 1-en-US, 2-tr-TR. Некоторые голоса специально предназначены для определённого языка. Иноязычные голоса не читают кириллицу.

Громкость: от -30 до +30 Дб, но необходимо следить за качеством результата. Некоторые эффекты требуют увеличения громкости.

Эффект: 1-эхо, 2-аудио закладка перед фразой. Закладка содержит текст: "Следующий текст очень важен. 3-реверс произношения, 4-подключение аудио вставки перед фразой, 5-подключение аудио вставки после фразы.

вид эффекта (условный код аудио файла для вставки): 0-a1.opus 1-a2.opus , 2-a3.opus , 3-a4.opus , 4-a5.opus , 5-a6.opus. Файлы необходимо готовить. Вложены примеры. Формат аудио строго Opus.

Кодирование SSML: 0-запрещено, 1-разрешено. Если разрешено, то голос считывает теги ssml, например паузу. Не требуется в тексте указывать открывающие и закрывающие теги ssml, они включаются автоматически при выполнении запроса.

Допустимые коды SSML

Коды показаны на примерах ключей и кодирования в тексте. Все фразы можно прослушать, выделив их и запустив скрипт.

```
{{ 1 0.9 3 0 -1 0 0 1 [ Комментарий ] }}
```

Пауза в тексте равная двум секундам. старт:<break time="2s"/> финиш!

```
<p><s>Возможна разметка на параграфы</s><s>Тогда паузы между предложениями удлинятся</s></p>
```

```
{{ 1 0.9 3 0 -1 0 0 1 [ Комментарий ] }}
```

Можно заменить любое слово в тексте на другое.

— _{Ag} мой любимый химический элемент, потому что он блестит.

```
{{ 1 0.9 3 0 -1 0 0 1 [ Комментарий ] }}
```

В разных регионах России по-разному произносят букву

```
<phoneme alphabet="ipa" ph="o">O</phoneme> в словах.
```

Международный фонетический алфавит (IPA) позволяет обозначать корректировать произношение.

Где-то говорят <phoneme alphabet="ipa" ph="mələko">молоко</phoneme> ,

а где-то <phoneme alphabet="ipa" ph="mələko">молоко</phoneme> .

Есть ещё один фонетический алфавит,

```
{{ 12 0.9 3 1 -1 0 0 1 [ Комментарий ] }}
```

Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA)

```
{{ 1 0.9 3 0 -1 0 0 1 [ Комментарий ] }}
```

Extended Speech Assessment Methods Phonetic Alphabet (X-SAMPA)

```
{{ 1 0.9 3 0 -1 0 0 1 [ Комментарий ] }}
```

Он тоже позволяет управлять произношением.

Где-то говорят <phoneme alphabet="x-sampa" ph="k@rtos`ka">кормошка</phoneme> ,

а где-то <phoneme alphabet="x-sampa" ph="kortos`ka">кормошка</phoneme> .

Авторизация

Скрипт может брать строки FOLDER_ID и OAuth_TOKEN из отдельного файла, но если вы не собираетесь свой экземпляр скрипта кому-либо передавать, то можно эти строки записать прямо в тело скрипта в раздел «Авторизация».

В первой строке должно быть значение FOLDER_ID, а во второй строке значение OAuth_TOKEN.

Вместо строк 126 и 128

```
FOLDER_ID=$(head -n 1 ~/YSK_ID/FID_OAT | tail -n 1)
```

```
OAuth_TOKEN=$(head -n 2 ~/YSK_ID/FID_OAT | tail -n 1)
```

следует написать:

```
FOLDER_ID=`значение FOLDER_ID в прямых кавычках`
```

```
OAuth_TOKEN=`значение OAuth_TOKEN в прямых кавычках`
```

Это единственный случай, когда требуется редактирование скрипта!

Можно не редактировать, но создать отдельный файл со строками без кавычек. В первой строке FOLDER_ID, а во второй OAuth_TOKEN.

Файл должен находиться по адресу ~/YSK_ID/ и иметь имя FID_OAT без расширения.

Ключ IAM_TOKEN, который получается при авторизации, действителен несколько часов. Если работа длится дольше, то необходимо получать новый ключ, когда истекло время действия предыдущего.

Все процедуры авторизации, контроля истечения времени действия токена и получения нового токена выполняются автоматически.

Полученный токен хранится в скрытом файле, который заменяется при устаревании ключа.

Важно! Перед запуском скриптов дать им права на запуск (позволять выполнение файла как программы).

В качестве примера вложены 08-11-2019_19-32-46_work_file.txt и 08-11-2019_21-29-10_clip.mp3 с обработанным текстом и выполненным озвучиванием.

Исходный текст в файле text.txt.

При следующем запуске будут удалены:

- **a.txt** в котором видно, какие параметры (основные) были переданы голосу скриптом.

- **clip.txt** и **CONtROL.txt**, в которых содержится текст сформированный скриптом для передачи.

Чаще всего содержание clip.txt и CONtROL.txt идентично. Но есть режим построочного формирования текста, когда структура текста в этих файлах окажется разной.

Этот режим включается комментированием строки 32 в **1-Say-this-text**, если вы не

редактировали этот файл самостоятельно. Но необходимость в этом режиме почти никогда не возникает.

Тестирование на чистой системе

1. Установка VM

2. Установка Mint 19.2

3.