

MODradio

Grzegorz Koperwas

13 stycznia 2022

1. Temat projektu:

Celem powstałego programu jest strumieniowanie piosenek z **trackerów** poprzez *Http Live Streaming*¹ w celu łatwego odsłuchu na urządzeniach mobilnych.

Zatem musi on dynamicznie łączyć kolejne pliki w jeden strumień, bez wcześniejszego wczytania ich wszystkich (Ilość tych plików wynosi 29 tysięcy, rozmiar około 50 gigabajtów w formie skompresowanej).

Opis problemu:

Archiwa strony `modarchive.org` są udostępniane w następującej formie:

1. Piosenki znajdują się w drzewie folderów rozróżniającym je ze względu na format, artystę czy rok dodania.

Jakiegokolwiek informacje zawarte w strukturze folderów mają być ignorowane, strumieniujemy piosenki w losowy sposób.

Odtwarzacz VLC może uzyskiwać dostęp po protokole SMB do serwera z plikami, lecz nie odtwarza ich losowo w prosty sposób.

2. Każda piosenka jest skompresowana jako archiwum ZIP.
3. Piosenki są przechowywane jako *moduły trackerów*, gdzie zamiast danych PCM przechowywane są w formie sampli i informacji jak je odtwarzać. Wynika to z architektury komputerów *Amiga*, gdzie ten rodzaj muzyki powstał.

Zatem program musi:

1. Przyjmować pliki audio w formie pozwalającej na utrzymywanie *bufora* następnych plików audio.
2. Przyjmować pliki, lub ścieżki do nich ze źródła łatwo dostępnego dla jakiegoś skryptu. Na przykład przez `stdin`.
3. Spełniać wymagania do łatwego zahostowania na moim klastrze *Docker Swarm*, brak GUI itp.

¹Dalej będę korzystał ze skrótu **HLS**

2. Opis pobieranych danych przez program:

Program pobiera przez standardowe wejście ścieżki do kolejnych plików audio. Pliki te są **usuwane** po zakończeniu ich strumieniowania.

Program po napotkaniu pliku, którego nie da się otworzyć lub zdekodować pomija dany plik.

Program stara się otwierać dwa pliki naraz. Plik, który jest właśnie strumieniowany, oraz plik, który jest następny w kolejce.

3. Opis otrzymanych rezultatów

Wydruk z programu

Program wypisuje do konsoli logi diagnostyczne, za przechowywanie ich w plikach odpowiada **systemd** lub **docker**.

Logi w omawianym przykładzie składają się z paru części:

- Logi ze znakami `<` oraz `>`, - Logi informujące jakie obiekty są tworzone przez program. Przykładowo:
 - `<Encoder for aac>` - Stworzono obiekt kodera formatu AAC, zawsze wyświetla się na początku programu.
 - `<Reader for /path/to/file>` - Stworzono obiekt demuxera, który czyta zawartość pliku. Jeżeli dany plik zawiera parę strumieni wideo lub audio, program wybiera pierwszy strumień audio. Powinien wspierać większość standardów (pewnie nawet zasoby sieciowe, zależy od wersji biblioteki `libav`).
 - `<Decoder for $codec>` - Stworzono obiekt dekodera kompresji, program *powinien* wspierać wiele różnych kodeków, jednak standard mp3 generuje niepoprawny dźwięk.
 - `<Resampler from $foo to $bar>` - Stworzono obiekt resamplera, który normalizuje częstotliwość próbkowania oraz zapis bitowy.
- Logi z znakami `[` oraz `]`², - Logi generowane przez bibliotekę `libav` - są zwykle w formie:

`[$źródło @ adres źródła] $wiadomość`

Zwykle są to logi o statusie muxera HLS, lecz w przypadku złego pliku wejściowego zawierają one dodatkowe informacje o błędzie. Program `ffmpeg`, który jest frontendem do biblioteki `libav` generuje te same logi, więc jego dokumentacja pomoże w diagnozowaniu problemu.

- Reszta:

Część logów w przykładzie pochodzi od skryptu realizującego przykładowe wykorzystanie programu. Jest on dołączony do kodu jako `./feeder.sh`.

²te kolorowe

Przykładowe logi znajdują się na załączniku 1.

```

1 [mpegts @ 0x7f0600355480] frame size not set
2 <Encoder for aac>
3 extracted /tmp/modfiles/lazertrack_heaven_2.mod
4 [aac @ 0x7f05f4005240] Estimating duration from bitrate, this may be inaccurate
5 <Reader for /tmp/modfiles/lazertrack_heaven_2.mod.aac>
6 <Decoder for aac>
7 <Resampler from 44100hz to 44100hz>
8 [hls @ 0x7f060034de00] Opening 'stream0.ts' for writing
9 [hls @ 0x7f060034de00] Opening 'stream.m3u8.tmp' for writing
10 extracted /tmp/modfiles/the_hardliner_-_whoronzon_gohonzon.xml
11 [aac @ 0x7f05ec001100] Estimating duration from bitrate, this may be inaccurate
12 <Reader for /tmp/modfiles/the_hardliner_-_whoronzon_gohonzon.xml.aac>
13 extracted /tmp/modfiles/the_savannus_never_never.669
14 [hls @ 0x7f060034de00] Opening 'stream1.ts' for writing
15 [hls @ 0x7f060034de00] Opening 'stream.m3u8.tmp' for writing
16 [hls @ 0x7f060034de00] Opening 'stream2.ts' for writing
17 [hls @ 0x7f060034de00] Opening 'stream.m3u8.tmp' for writing
18 extracted /tmp/modfiles/gustavo6046_-_trulix.it
19 extracted /tmp/modfiles/jabdah-cover.xml
20 extracted /tmp/modfiles/jason_ee-futurefuckballs2010_cover.it
21 extracted /tmp/modfiles/owcfullfrontal.it
22 [hls @ 0x7f060034de00] Opening 'stream3.ts' for writing
23 [hls @ 0x7f060034de00] Opening 'stream.m3u8.tmp' for writing
24 extracted /tmp/modfiles/skyline_-_boners.it
25 extracted /tmp/modfiles/ko0x_-_galaxy_guppy.it
26 [hls @ 0x7f060034de00] Opening 'stream4.ts' for writing
27 [hls @ 0x7f060034de00] Opening 'stream.m3u8.tmp' for writing
28 extracted /tmp/modfiles/pasyada_alex_-_decil.xml
29 extracted /tmp/modfiles/badboyremixhypnosis.mod
30 [hls @ 0x7f060034de00] Opening 'stream5.ts' for writing
31 [hls @ 0x7f060034de00] Opening 'stream.m3u8.tmp' for writing
32 Waiting for modradio to pickup data
33 [hls @ 0x7f060034de00] Opening 'stream6.ts' for writing
34 [hls @ 0x7f060034de00] Opening 'stream.m3u8.tmp' for writing
35 [hls @ 0x7f060034de00] Opening 'stream7.ts' for writing
36 [hls @ 0x7f060034de00] Opening 'stream.m3u8.tmp' for writing
37 ...

```

Załącznik 1: Przykładowe logi z programu.

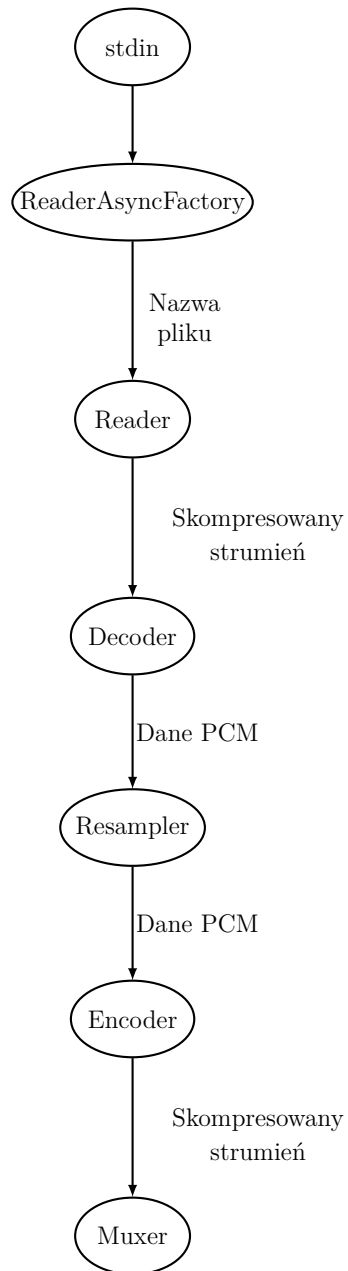
Pliki tworzone przez program:

Program tworzy w aktualnym katalogu strumień w formacie HLS jako pliki `stream$x.ts` oraz plik „spis” `stream.m3u8`. Te pliki powinny być hostowane przez serwer HTTP, jako pliki statyczne. Programy takie jak *VLC Media Player*, *ffplay*, *safari* czy przeglądarki internetowe na systemie android odtworzą je bez problemu, nawet z dysku. Dla przeglądarek na komputerach PC trzeba dostarczyć demuxer w *javascript'cie*, co nie jest przedmiotem tego projektu.

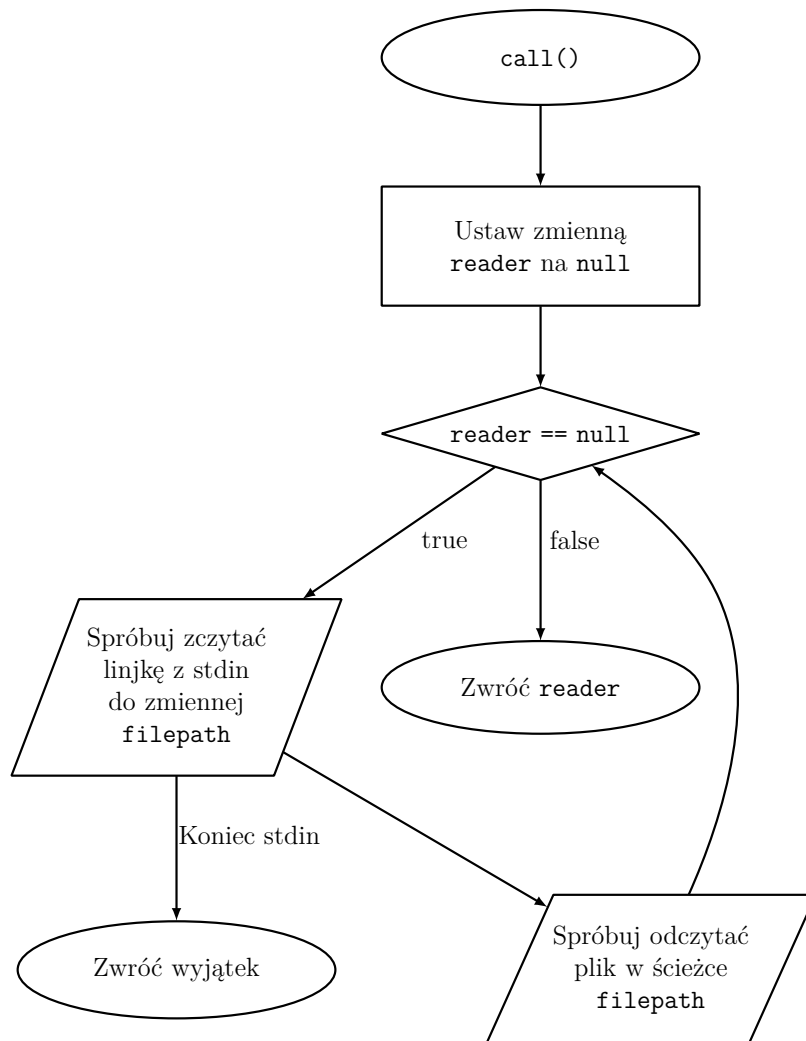
4. Zastosowane algorytmy:

Obieg danych jest przedstawiony na załączniku 2.

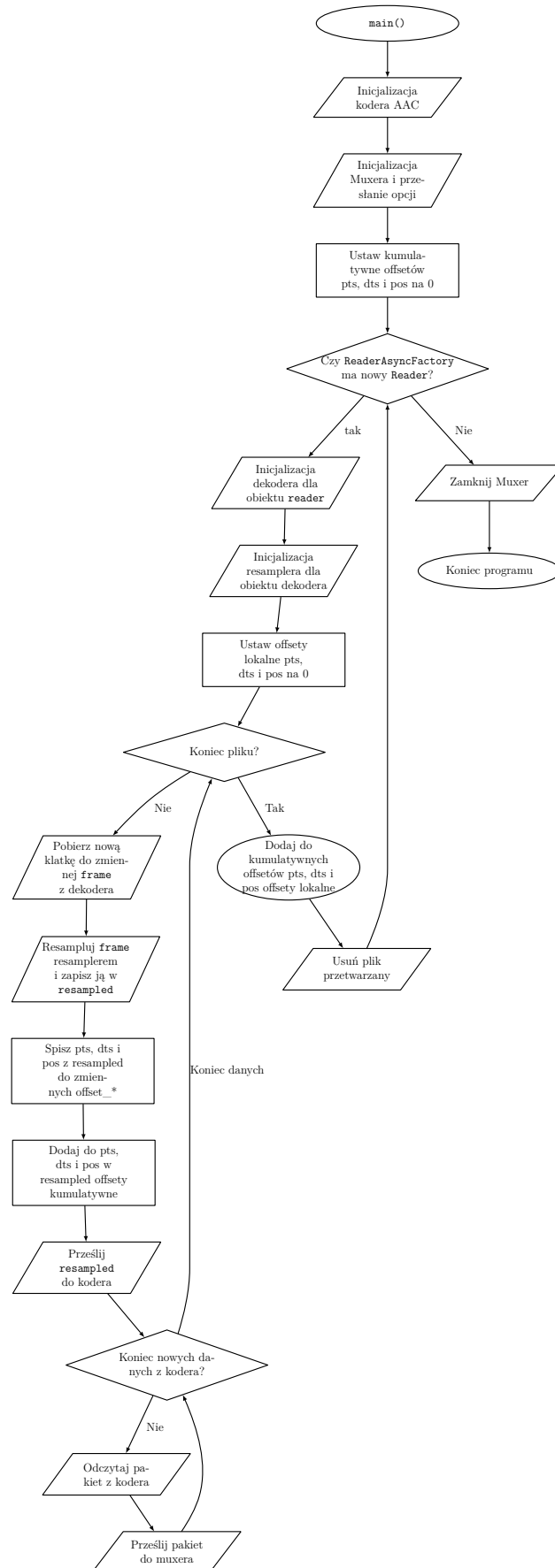
W programie został zastosowany mechanizm asynchroniczności poprzez klasę standardową `Future` oraz `ReaderAsyncFactory`.



Załącznik 2: Pipeline danych



Załącznik 3: Schemat blokowy działania klasy `ReaderAsyncFactory`



Załącznik 4: Ogólny schemat działania programu.

5. Testy na poprawność działania programu:

Poprawność działania była sprawdzana poprzez odsłuch strumieni generowanych przez program oraz hostowanie ich serwerem `http` z biblioteki standardowej języka *Python*, poprzez komendę `python -m http.server`.

6. Wnioski:

Po stworzeniu tego programu lepiej rozumiem architekturę biblioteki `libav`, gdyż wcześniej wykorzystywałem ją tylko do dekodowania audio w poprzednim projekcie z [programowania II](#). w języku C++ oraz w projekcie komercyjnym dla Gliwickiej firmy *APA Group*³ w języku *python*.

³Plakat rekrutacyjny „Pracuj w Czarnym Domu” wisi na czwartym piętrze wydziału Matematyki Stosowanej obok schodów.