FFmpeg auf Android portieren

Originalartikellink (CSDN)

Der Ursprung

Für diejenigen, die FFmpeg noch nicht kennen, können Sie sich zunächst FFmpeg Common Basic Commands ansehen und dessen Funktionen genau studieren – wie z.B. die Kombination von Audio und Video, das Abspielen verschiedener Codecs, das Schneiden von Videos, das Zusammenfügen mehrerer Bilder zu einem Video und das Mischen von Audio, sowie die Konvertierung von Formaten. Es kann viele Anwendungsszenarien mit leistungsstarken und interessanten Funktionen bereichern.

Am Beispiel von Apps wie "□□□" (Voice Dubbing Show) zeigt sich, dass das Synchronisieren von Stimmen nicht nur äußerst unterhaltsam ist, sondern oft auch die Attraktivität eines Produkts steigern kann. Wir planten die Entwicklung eines Dubbing-Moduls und wurden dabei auf FFmpeg aufmerksam, das wir auf die Android-Plattform portieren wollten. Nach etwa 3 bis 4 Tagen und mehreren Versuchen mit verschiedenen Versionen, bei denen viele Online-Artikel keinen Erfolg brachten, stießen wir schließlich auf ein Tutorial mit dem Titel "android □□ ffmpeg □□□□□" (Verwendung von FFmpeg in Android und Aufruf der Schnittstellen), das uns zum Ziel führte. Praktische Tests zeigten, dass nur die Kombination aus FFmpeg 1.2 und ndk-r9 erfolgreich portiert werden konnte.

Ansatz

FFmpeg ist ein in C geschriebenes Projekt, das eine `main()`-Funktion enthält. Unser Ziel ist:

1. Verwenden Sie das Android NDK, um `libffmpeg.so` zu kompilieren.
2. Nutzen Sie diese Bibliothek, um die `ffmpeg.c` Datei zu kompilieren und zu modifizieren. Ändern Sie die ursprüngliche `main()` Funktion in `video_merge(int argc, char **argv)` um, damit sie direkt über JNI aufgerufen werden kann, um Aufgaben wie das Zusammenführen von Videos durchzuführen.

Zum Beispiel kann die Videosynthese auf folgende Weise implementiert werden (entspricht dem Befehlszeilenbefehl `ffmpeg -i src1 -i src2 -y output`):

```
video_merge(5, argv); // wobei argv die Befehlszeilenparameter simuliert
```

Umgebung

- Betriebssystem: Ubuntu 12.04
- FFmpeg-Version: 1.2
- NDK-Version: ndk-r9

Bevor Sie mit der Arbeit beginnen, wird empfohlen, einige relevante Tutorials zu konsultieren. Wenn Sie auf Probleme stoßen, können Sie zurückkommen und diesen Artikel vergleichen, um zu vermeiden, zu viele Umwege zu machen.

Ändern der Schnittstelle und der Android.mk

Beim Schreiben von JNI-Schnittstellen für FFmpeg muss eine `Android.mk`-Datei erstellt werden, um Bibliotheken zu verlinken und somit eine nutzbare `.so`-Datei zu generieren. Einige Beispiel-`Android.mk`-Dateien können in verschiedenen Umgebungen möglicherweise nicht direkt erfolgreich ausgeführt werden. Ihre Aufgabe besteht darin, dem NDK mitzuteilen, welche Quelldateien kompiliert und mit welcher Bibliothek verlinkt werden sollen.

Ich habe eine Methode des “zweimaligen Kompilierens und anschließenden Linkens” verwendet:

1. Zuerst wird eine Shared Library namens `myffmpeg` kompiliert.
2. In einem anderen Modul `ffmpeg-jni` wird dann `myffmpeg` eingebunden, um schließlich das gewünschte `.so`-File zu generieren.

Außerdem muss die kompilierte Datei `libffmpeg.so` im `jni`-Verzeichnis abgelegt werden, um sicherzustellen, dass sie während des Linkens gefunden werden kann.

Debugging von FFmpeg

Nach der Portierung von FFmpeg ist es oft notwendig, Funktionen über JNI aufzurufen und auf der C-Ebene zu debuggen. Wenn man detaillierte Logausgaben wie in der Befehlszeile sehen könnte, wäre es viel einfacher, Probleme zu lokalisieren.

In Eclipse können Sie durch Drücken von Strg und Klicken auf eine Aufrufstelle wie `av_log` zur Implementierung der Funktion `av_log_default_callback` in `ffmpeg/libavutil/log.c` gelangen. Diese Funktion ruft `__android_log_print` von Android auf, um die Ausgabe in Logcat zu drucken. Durch die Betrachtung dieser Ausgaben können Sie den internen Zustand von FFmpeg ermitteln, was bei der Fehlerbehebung von Problemen wie fehlgeschlagenen Synthesen oder der Nichtunterstützung bestimmter Codecs hilfreich ist.

Manchmal kann FFmpeg eine Ausnahme auslösen, die zum Absturz der App führt. Um dies zu lokalisieren, können Sie den folgenden Befehl verwenden:

```
adb shell logcat | ndk-stack -sym obj/local/armeabi
```

Hinweis: Der obige Befehl bleibt auf Englisch, da es sich um einen spezifischen technischen Befehl handelt, der in der Regel nicht übersetzt wird.

Wenn die ursprüngliche `main()`-Funktion von FFmpeg am Ende ein `exit(0)` enthält, denken Sie daran, dies auszukommentieren, da dies sonst zum Beenden der Anwendung führen würde.

Speicherlecks und Service-Lösungen

Nach Abschluss der Synthese kann es bei erneutem Aufruf zu einem Fehler wie `"INVALID HEAP ADDRESS IN dlfree ffmpeg"` kommen, was meist auf eine unvollständige Speicherfreigabe durch FFmpeg zurückzuführen ist. Ein möglicher Workaround besteht darin, den Syntheseprozess in einem separaten `Service` auszuführen und diesen Service nach Abschluss der Synthese zu beenden, um die Ressourcen freizugeben.

```
<!-- AndroidManifest.xml -->
<service android:name=".FFmpegService" />
```

Durch die Registrierung von `Receiver` und ähnlichen Methoden kann der Service nach Abschluss der Synthese selbstständig beendet werden, wodurch Speicherprobleme bei wiederholten Aufrufen vermieden werden können.

Mögliche Probleme

- **AAC-Dateiabspielprobleme**
  Einige Gerätemodelle (z. B. Xiaomi 2s) können möglicherweise AAC-codierte Audiodateien nicht mit dem standardmäßigen `MediaPlayer` abspielen.

- **Unzureichende Encoder-Unterstützung**
  Um Formate wie AMR-NB und MP3 zu unterstützen, müssen die entsprechenden Optionen beim Kompilieren von FFmpeg manuell aktiviert werden. Wenn das Kompilierungsskript die erforderlichen Bibliotheken oder Header-Dateien nicht findet, wird es mit einer Fehlermeldung beendet.

- **Synthesizer-Geschwindigkeit**
  Die Synthese eines 10-Sekunden-Videos mit einer Auflösung von 1280×720 und gemischtem Audio kann zwischen einigen Dutzend Sekunden und einer Minute dauern. Aus Benutzersicht könnte es besser sein, den Benutzern zunächst eine Vorschau zu ermöglichen, bevor sie sich für die endgültige Synthese entscheiden.

In der konkreten Implementierung von Dubbing-Apps ist es üblich, wie folgt vorzugehen: 1. Laden Sie das Originalvideo, die Untertitel und die Audiodatei, aus der die zu synchronisierenden Abschnitte entfernt wurden, im Voraus herunter. 2. Nehmen Sie die Stimme des Benutzers auf. Bei der Synthese müssen Sie lediglich die Aufnahme mit den stummen Abschnitten zusammenführen. 3. Wenn Sie mit der lokalen Synthesezeit nicht zufrieden sind, können Sie die Audio- und Videodaten auf den Server hochladen, wo die Synthese serverseitig durchgeführt wird. Anschließend können Sie das fertige Ergebnis herunterladen.

Verwendung des NDK in Eclipse

Es ist nicht notwendig, `ndk-build` in der Befehlszeile einzugeben. Klicken Sie einfach mit der rechten Maustaste auf das Projekt in Eclipse und wählen Sie **Android Tools → Add Native Support**. Danach wird `ndk-build` automatisch jedes Mal ausgeführt, wenn Sie auf “Run” klicken.

JNI-Headerdatei mit einem Klick generieren

Das Schreiben von JNI-Funktionsheaderdateien kann mühsam sein, aber es kann automatisch mit dem `javah`-Befehl generiert werden.
In Eclipse kann dies als externes Tool konfiguriert werden, um die Headerdateien mit einem ähnlichen Befehl wie folgt zu generieren:

```
javah -jni -classpath bin/classes -d jni com.example.ffmpeg.MyFFmpeg
```

Hinweis: Der obige Befehl ist ein Java Native Interface (JNI)-Befehl und wird in der Regel nicht übersetzt, da er spezifisch für die Entwicklungsumgebung und die Programmiersprache ist. Der Befehl generiert Header-Dateien für native Methoden in der angegebenen Java-Klasse.

Nach der Ausführung wird im `jni`-Verzeichnis eine Datei ähnlich wie `com_example_ffmpeg_MyFFmpeg.h` erzeugt. Anschließend müssen Sie diese Datei in Ihrem C-Code mit `#include` einbinden und die entsprechenden Funktionen implementieren.

Zusammenfassung

Die Portierung von FFmpeg auf Android umfasst verschiedene Wissensbereiche, darunter die Konfiguration der NDK-Umgebung, die Kompilierung und Verknüpfung von C/C++-Code, JNI-Aufrufe sowie die Audio- und Videocodierung und -decodierung. Wenn Probleme wie fehlgeschlagene Synthese, nicht unterstützte Formate oder Linker-Fehler auftreten, ist eine sorgfältige Überprüfung der Konfiguration und der Log-Ausgaben erforderlich. Ich hoffe, dass dieser Artikel Ihnen helfen kann, einige Fallstricke zu vermeiden. Wenn Sie ebenfalls FFmpeg verwenden, teilen Sie gerne Ihre Erfahrungen oder Probleme in den Kommentaren, um sich gegenseitig auszutauschen und voneinander zu lernen.