

METHOD, STORAGE MEDIUM AND ELECTRONIC DEVICE FOR DYNAMICALLY LOADING SO

TECHNICAL FIELD

The present invention relates to the field of computer technologies, and in particular, to a method, a storage medium and an electronic device for loading so files by an Android application.

BACKGROUND ART

With the development of terminal devices and computer technologies, smartphone terminal devices have become indispensable electronic products for almost everyone. In the development process of a mobile application, how to reduce the pack size of an app, increase the install rate and reduce the uninstall rate is a problem that developers need to be aware of all the time. The download conversion rate is improved, the smaller an install pack is, the higher the conversion rate is; and in different distribution platforms, based on data statistics and analysis, a pack size limit is given, therefore, how to reduce the size of an app is an important measure to ensure the installation and operation of the application.

At present, in order to reduce the pack size of an application, a first consideration is to clean up useless resources and codes before packing, a second consideration is to perform confusion, compression, format conversion and on-demand dynamic loading of resource files and compression and on-demand dynamic loading of resources, fonts and audio files under assets during packing, and for an application with complex functions, third-party libraries with duplicate functionality are integrated. A so file under lib is also one of the main reasons for pack size increasing due to its function and compatibility with a cpu architecture. A conventional means is to store a so library in a preset server, monitor application startup and check whether a local so exists, alternatively, download the so library from the preset server to a local sandbox, and load the so library through System.load, wherein it is necessary to consider so association and loading order problems, as well as a problem of loading a shared library containing text relocations after Android6.0.

SUMMARY

In view of the deficiencies of the prior art, the present invention aims to provide a method, an apparatus, a storage medium and an electronic device for dynamically loading so.

In order to achieve the above purpose, the present invention adopts the following technical solution:

a method for dynamically loading so includes the following steps:

(I) importing a so library based on functions at application startup, wherein a loading time of the so library imported by a third-party jar is not controlled by a code logic,

and finding out so library files suitable for being asynchronously loaded according to requirements and scenarios, which is specifically implemented as follows:

1.1: intercepting a System.loadLibrary function;

1.2: printing a call chain;

1.3: distinguishing loading modes as required, including a mode that related files born on a home page are not loaded asynchronously and a mode that function module related files are required to be loaded asynchronously; and

1.4: outputting a loading priority array;

(II) before the so library files are downloaded completely, processing the update and iterative upgrade of application related function logics and versions, which is specifically implemented as follows:

2.1: intercepting a startActivity function;

2.2: reading a configuration; and

2.3: detecting whether it is necessary to load the so library files asynchronously; when asynchronous loading is not required, directly starting a target activity; intercepting a onCreate function of the target activity, and mounting a local path; and when the asynchronous loading is required, judging whether the so library files have been downloaded, if so, directly starting the target activity, intercepting the onCreate function and mounting the local path, and if not, downloading target so library files from a preset server to a sandbox path, then after the downloading is completed, starting the target activity, intercepting the onCreate function of the target activity and mounting the local path;

(III) deleting the so library files during a packing phase, which is specifically implemented as follows:

3.1: reading and parsing a json configuration file;

3.2: carrying out security verification, baseUrl compliance verification and soInfo compliance verification on the json configuration file;

3.3: when all the verifications have been passed, carrying out MD5 verification on a packing process and a file configuration so as to prevent developers from misconfiguring that results in mismatching; and

3.4: removing the so library files that need to be loaded dynamically and will be subsequently mounted based on the dynamic loading mode, so as to achieve the purpose of pack size reduction.

The present invention also provides a computer readable storage medium in which a computer program is stored, and when the computer program is executed by a processor, the above-mentioned method is implemented.

The present invention also provides an electronic device including a memory configured to store a computer program and a processor configured to implement the

method when the computer program is executed.

The present invention has the beneficial effects that: traditional so library files are uploaded to the network, then downloaded to a local sandbox, and loaded through `System.load`, in such a way, there will be an in-load dependency priority problem, a text relocations problem, and the like. According to the method provided by the present invention, a custom native library path is inserted into the front of `nativeLibraryDirectories` through reflection, so that even if there is a so with the same name in an install pack libs directory, an external so with a specified path is preferentially loaded to realize updating or downloading requirements, therefore, the method is less intrusive; and the method is used only by three steps: introducing, verifying and analyzing configurations, and dynamically issuing so according to service configuration needs, thereby solving problems caused by traditional loading methods.

BRIEF DESCRIPTION OF THE DRAWINGS

FIG. 1 is a flowchart of step (I) in Embodiment 1 of the present invention;

FIG. 1 is a flowchart of step (II) in Embodiment 1 of the present invention;

FIG. 1 is a flowchart of step (III) in Embodiment 1 of the present invention;

DETAILED DESCRIPTION OF THE EMBODIMENTS

The following further describes the present invention with reference to the accompanying drawings. It should be noted that detailed implementations and specific operation processes are given in the embodiment based on the technical solution, but the scope of protection of the present invention is not limited to the embodiment.

The embodiment provides a method for dynamically loading so, including the following steps:

(I) importing a so library based on functions at application startup, where a loading time of the so library imported by a third-party jar (aar) is not controlled by a code logic, and finding out so library files suitable for being asynchronously loaded according to requirements and scenarios, which, as shown in FIG. 1, is specifically implemented as follows:

1.1: intercepting a `System.loadLibrary` function;

1.2: printing a call chain;

1.3: distinguishing loading modes as required, including a mode that related files born on a home page are not loaded asynchronously and a mode that function module related files are required to be loaded asynchronously; and

1.4: outputting a loading priority array to provide reference for subsequent configurations;

(II) before the so library files are downloaded completely, processing the update and iterative upgrade of application related function logics and versions, which, as shown

in FIG. 2, is specifically implemented as follows:

2.1: intercepting a startActivity function;

2.2: reading a configuration; and

2.3: detecting whether it is necessary to load the so library files asynchronously; when the asynchronous loading is not required, directly starting a target activity; intercepting a onCreate function of the target activity, and mounting a local path; and when the asynchronous loading is required, judging whether the so library files have been downloaded, if so, directly starting the target activity, intercepting the onCreate function and mounting the local path, and if not, downloading target so library files from a preset server to a sandbox path, then after the downloading is completed, starting the target activity, intercepting the onCreate function of the target activity and mounting the local path;

Further, in the embodiment, in order to optimize the download process, it is necessary to provide a user interaction prompt to display a download progress dialog box.

(III) deleting the so library files during a packing phase, which, as shown in FIG. 3, is specifically implemented as follows:

3.1: reading and parsing a json configuration file;

3.2: carrying out security verification, baseUrl compliance verification and soInfo compliance verification on the json configuration file;

3.3: when all the verifications have been passed, carrying out md5 verification on a packing process and a file configuration so as to prevent developers from misconfiguring that results in mismatching; and

3.4: removing the so library files that need to be loaded dynamically and will be mounted subsequently based on the dynamic loading mode, so as to achieve the purpose of pack size reduction.

For those skilled in the art, various corresponding variations and deformations can be given according to the above technical solution and conceptions, and all these variations and deformations should be included in the scope of protection of the claims of the present invention.

CLAIMS

1. A method for dynamically loading so, wherein the method comprises the following steps:

(I) importing a so library based on functions at application startup, where a loading time of the so library imported by a third-party jar is not controlled by a code logic; and finding out so library files suitable for being asynchronously loaded according to requirements and scenarios, which is specifically implemented as follows:

1.1: intercepting a System.loadLibrary function;

1.2: printing a call chain;

1.3: distinguishing loading modes as required, including a mode that related files born on a home page are not loaded asynchronously and a mode that function module related files are required to be loaded asynchronously; and

1.4: outputting a loading priority array;

(II) before the so library files are downloaded completely, processing the update and iterative upgrade of application related function logics and versions, which is specifically implemented as follows:

2.1: intercepting a startActivity function;

2.2: reading a configuration; and

2.3: detecting whether it is necessary to load the so library files asynchronously; when asynchronous loading is not required, directly starting a target activity; intercepting a onCreate function of the target activity, and mounting a local path; and when the asynchronous loading is required, judging whether the so library files have been downloaded, if so, directly starting the target activity, intercepting the onCreate function and mounting the local path, and if not, downloading target so library files from a preset server to a sandbox path, then after the downloading is completed, starting the target activity, intercepting the onCreate function of the target activity and mounting the local path;

(III) deleting the so library files during a packing phase, which is specifically implemented as follows:

3.1: reading and parsing a json configuration file;

3.2: carrying out security verification, baseUrl compliance verification and soInfo compliance verification on the json configuration file;

3.3: when all the verifications have been passed, carrying out MD5 verification on a packing process and a file configuration so as to prevent developers from misconfiguring that results in mismatching; and

3.4: removing the so library files that need to be loaded dynamically and will be mounted subsequently based on the dynamic loading mode, so as to achieve the purpose of pack size reduction.

2. A computer readable storage medium, wherein a computer program is stored in the computer readable storage medium, and when the computer program is executed by a processor, the method according to Claim 1 is implemented.
3. An electronic device, comprising a memory configured to store a computer program and a processor configured to implement the method according to Claim 1 when the computer program is executed.