Android Java代码Overlay机制

Last edited by **caoquanli** 1 month ago

Android Java代码Overlay机制

Table of Contents

简述

- Android Java代码Overlay机制，指将我们在Android原生Java代码上修改过的文件，单独放在一个Overlay目录下，Overlay目录下目录路径和原生目录路径相同；而在编译代码的时候就会去检查Overlay目录下的文件，如果Overlay目录下有与原生相同目录路径的文件，那么就将这个文件放入编译的源文件中参与编译，而将原生目录路径下相同的文件从编译的源文件中去除。

用途

- 在相同Android固件版本情况下，进行代码移植会比较方便。

原理步骤

- 建立用于进行代码Overlay的Overlay代码仓库和目录，此Overlay仓库内的目录结构与被Overlay的原生代码仓库内目录结构保持一致，用于Overlay的修改后文件命名也与未修改的原生文件一致；
- 编译时，在将未修改的原生文件放入编译的源文件中后，检查Overlay目录下是否存在需要参与编译的文件，若有，就进入代码Overlay操作；若无，就继续进行未修改的原生文件编译；
- 进入代码Overlay操作后，将需要参与编译的Overlay文件加入编译源文件，然后查找编译源文件中是否存在与刚才加入的Overlay文件路径结构、包名文件名相同的原生代码文件，若有就将它从编译的源文件中去除；
- 至此已完成代码Overlay操作，继续之后的编译操作；
- 由于Overlay文件和原生代码文件编译出的包名文件名类名都一样，编译时代码中对被替换的原生代码文件的调用，都会变为对对应Overlay文件的调用，实现Android Java代码Overlay机制。

实现效果

- 修改原生java文件或aidl文件后，保持原生文件不变，将修改过的文件按原生文件在AOSP_SRC仓库目录下的路径保存在${AOSP_SRC}/vendor/iauto/目录下相同路径；
- 在Overlay目录下新增需要参与编译的aidl接口文件时，需要在对应的Android.mk文件中,将Overlay目录下新增的aidl文件相对路径添加进LOCAL_SRC_FILES中（java文件不需要）；
- 然后进行全代码编译，就会用Overlay目录中的修改后文件替换原生文件进行编译了。

实现方法

1. 准备工作

- 在AOSP_SRC的vendor/iauto/目录下创建与Android原生代码仓库目录对应的Overlay代码仓库目录。

2. 修改Android.mk文件

2.1 添加java文件的Overlay操作及aidl文件Overlay操作

2.1.1 一般apk及单个子目录下的java文件Overlay及aidl文件Overlay

```
 # LOCAL_PATH 代表当前目录路径
LOCAL_PATH := $(call my-dir)

.......

# LOCAL_SRC_FILES 为参与编译的编译源文件，原生代码中的大部分参与编译的java文件和aidl文件被加入其中
LOCAL_SRC_FILES := $(call all-java-files-under, src)

# xxx 代表模块名，不同mk文件模块名不同，建议按实际情况命名修改
# 进入 java overlay
$(warning xxx java overlay start)

# 定义java文件所在目录名，一般apk的目录名通常为“src”，非apk模块（单个子目录）目录名通常为“java”，请按实际
xxx_overlay_javafile_dirname := src

# 自定义空值
xxx_overlay_empty :=

# 以LOCAL_PATH作为获取当前目录层级的基准
xxx_cur_path := $(LOCAL_PATH)
# $(warning $(xxx_cur_path))
# 替换LOCAL_PATH中的“/”为空字符串，变为空格分割的各级目录名字符串
xxx_pack_list := $(strip $(subst /, $(xxx_overlay_empty),$(xxx_cur_path)))
# $(warning $(xxx_pack_list))

xxx_overlay_one_back_dir := ../

# 按目录名个数添加“../”
xxx_overlay_back_dir_prefix_list := $(strip $(foreach xxx_pack,$(xxx_pack_list),$(xxx_overl
# $(warning $(xxx_overlay_back_dir_prefix_list))

# 自定义空格变量，注意两个空字符串变量中间夹一个空格，才能成功赋值空格变量
xxx_overlay_space := $(xxx_overlay_empty) $(xxx_overlay_empty)

# 去除空格，将所有“../”合并，组成相对路径前缀
xxx_overlay_back_dir_prefix := $(strip $(subst $(xxx_overlay_space),$(xxx_overlay_empty),$(
# $(warning $(xxx_overlay_back_dir_prefix))

xxx_overlay_root_dir := vendor/iauto/

# 组成Overlay目录路径前缀，得到当前目录到Overlay目录的相对路径
xxx_overlay_root_prefix := $(addprefix $(xxx_overlay_back_dir_prefix), $(xxx_overlay_root_d

# 利用warning log输出LOCAL_PATH信息
$(warning $(LOCAL_PATH))
# 在当前目录路径上添加Overlay目录路径前缀，组合成对应的Overlay目录路径
xxx_overlay_subdirs := $(addprefix $(xxx_overlay_root_prefix), $(LOCAL_PATH)/)
$(warning $(xxx_overlay_subdirs))
# $(warning $(xxx_overlay_subdirs)$(xxx_overlay_javafile_dirname)/)

# 查找vendor/iauto/目录下所有java文件
# all-java-files-under 为查找目录下所有java文件的函数
xxx_ext_overlay_java_list := $(call all-java-files-under, $(xxx_overlay_root_prefix))
# $(warning $(xxx_ext_overlay_java_list))

# vendor/iauto/目录下java文件列表判空，不为空继续，为空则结束Overlay操作
ifneq ($(xxx_ext_overlay_java_list), $(xxx_overlay_empty))
$(warning xxx_ext_overlay_java_list not empty)

# 获取vendor/iauto/目录下的java文件目录结构
xxx_ext_overlay_dir_list := $(dir $(xxx_ext_overlay_java_list))
# $(warning $(xxx_ext_overlay_dir_list))

# vendor/iauto/目录下的java文件目录结构列表判空，不为空继续，为空则结束Overlay操作
```

```
ifneq ($(xxx_ext_overlay_dir_list), $(xxx_overlay_empty))
$(warning xxx_ext_overlay_dir_list not empty)

# 在vendor/iauto/目录下存在的java文件目录中筛选出需要Overlay的目录路径，并进行去重复的处理
xxx_ext_overlay_subdir := $(sort $(strip $(filter $(xxx_overlay_subdirs)$(xxx_overlay_javaf
# $(warning $(xxx_ext_overlay_subdir))

# 需要Overlay的目录路径列表判空，不为空继续，为空则结束Overlay操作
ifneq ($(xxx_ext_overlay_subdir), $(xxx_overlay_empty))
$(warning xxx_ext_overlay_subdir not empty)

# 查找需要Overlay目录路径下的java文件
xxx_ext_overlay_java_files := $(call all-java-files-under, $(xxx_overlay_subdirs)$(xxx_over
# $(warning $(xxx_ext_overlay_java_files))

# 将查找到的需要Overlay目录路径下的java文件添加进参与编译的源文件
LOCAL_SRC_FILES += $(xxx_ext_overlay_java_files)

# 将Overlay java文件路径中的Overlay目录路径去除，变成被Overlay的原生java文件路径
xxx_orign_ext_overlayed_files := $(strip $(subst $(xxx_overlay_subdirs), $(xxx_overlay_empt
# $(warning $(xxx_orign_ext_overlayed_files))

# 将参与编译的源文件中的被Overlay的原生java文件去除
LOCAL_SRC_FILES := $(filter-out $(xxx_orign_ext_overlayed_files), $(LOCAL_SRC_FILES))

# java overlay 完成
$(warning xxx java overlay ok)

endif

endif

endif

# java overlay 结束
$(warning xxx java overlay end)

# 进入 aidl overlay
$(warning xxx aidl overlay start)

# LOCAL_AIDL_INCLUDES中的目录路径是用来查找aidl接口中import的类的
# LOCAL_PATH、LOCAL_PATH/src、FRAMEWORKS_BASE_JAVA_SRC_DIRS属于默认的查找路径
# 将三个默认路径对应的Overlay目录下的路径也作为默认路径添加进LOCAL_AIDL_INCLUDES
LOCAL_AIDL_INCLUDES += \
      $(addprefix $(xxx_overlay_root_dir), $(LOCAL_PATH)) \
      $(addprefix $(xxx_overlay_root_dir), $(LOCAL_PATH)/src) \
      $(addprefix $(xxx_overlay_root_dir), $(FRAMEWORKS_BASE_JAVA_SRC_DIRS))
# $(warning $(LOCAL_AIDL_INCLUDES))

# 查找vendor/iauto/目录下所有符合I*.aidl类型（匹配大小写）的文件
# all-Iaidl-files-under 为查找目录下所有I*.aidl类型文件的函数
xxx_ext_overlay_aidl_list := $(call all-Iaidl-files-under, $(xxx_overlay_root_prefix))
# $(warning $(xxx_ext_overlay_aidl_list))

# vendor/iauto/目录下I*.aidl类型文件列表判空，不为空继续，为空则结束Overlay操作
ifneq ($(xxx_ext_overlay_aidl_list), $(xxx_overlay_empty))
$(warning xxx_ext_overlay_aidl_list not empty)

# $(warning $(xxx_overlay_subdirs))
# 将vendor/iauto/目录下I*.aidl类型文件列表转换为对应的原生目录下的路径
xxx_ext_overlay_aidl_to_orign_list := $(strip $(subst $(xxx_overlay_subdirs), $(xxx_overlay
# $(warning $(xxx_ext_overlay_aidl_to_orign_list))

# 转换后的原生路径列表判空，不为空继续，为空则结束Overlay操作
ifneq ($(xxx_ext_overlay_aidl_to_orign_list), $(xxx_overlay_empty))
$(warning xxx_ext_overlay_aidl_to_orign_list not empty)

# 利用转换后的原生路径列表从LOCAL_SRC_FILES中筛选出需要Overlay的原生aidl文件
xxx_orign_overlayed_aidl := $(filter $(xxx_ext_overlay_aidl_to_orign_list), $(LOCAL_SRC_FIL

# 需要Overlay的原生aidl文件列表判空，不为空继续，为空则结束Overlay操作
ifneq ($(xxx_orign_overlayed_aidl), $(xxx_overlay_empty))
$(warning xxx_orign_overlayed_aidl not empty)
# $(warning $(xxx_orign_overlayed_aidl))
```

```
# 将LOCAL_SRC_FILES中需要Overlay的原生aidl文件去除
LOCAL_SRC_FILES := $(filter-out $(xxx_orign_overlayed_aidl), $(LOCAL_SRC_FILES))

# $(warning $(xxx_overlay_subdirs))
# 利用Overlay目录路径和需要Overlay的原生aidl文件列表组合成Overlay aidl文件列表，并将其添加进参与编译的源
LOCAL_SRC_FILES += $(addprefix $(xxx_overlay_subdirs), $(xxx_orign_overlayed_aidl))
# $(warning $(addprefix $(xxx_overlay_subdirs), $(xxx_orign_overlayed_aidl)))

# aidl overlay 完成
$(warning xxx aidl overlay ok)

endif

endif

endif

# aidl overlay 结束
$(warning xxx aidl overlay end)

......
```

2.1.2 多子目录下的java文件Overlay及aidl文件Overlay

此处以frameworks/base/Android.mk为例

```
 # LOCAL_PATH 代表当前目录路径，此处 LOCAL_PATH 为 frameworks/base/
LOCAL_PATH := $(call my-dir)

......

# FRAMEWORKS_BASE_SUBDIRS comes from build/core/pathmap.mk
# LOCAL_SRC_FILES 为参与编译的编译源文件，原生代码中的大部分java文件被加入其中
LOCAL_SRC_FILES := \
        $(call find-other-java-files,$(FRAMEWORKS_BASE_SUBDIRS)) \
        $(call all-proto-files-under, core/proto)

# frameworks java overlay
$(warning frameworks java overlay start)
# 不同模块最好修改变量名前缀，易于区分
# 自定义空值
frameworks_overlay_empty :=

# 以LOCAL_PATH作为获取当前目录层级的基准
frameworks_cur_path := $(LOCAL_PATH)
# $(warning $(frameworks_cur_path))
# 替换LOCAL_PATH中的“/”为空字符串，变为空格分割的各级目录名字符串
frameworks_pack_list := $(strip $(subst /, $(frameworks_overlay_empty),$(frameworks_cur_pat
# $(warning $(frameworks_pack_list))

frameworks_overlay_one_back_dir := ../

# 按目录名个数添加“../”
frameworks_overlay_back_dir_prefix_list := $(strip $(foreach frameworks_pack,$(frameworks_p
# $(warning $(frameworks_overlay_back_dir_prefix_list))

# 自定义空格变量，注意两个空字符串变量中间夹一个空格，才能成功赋值空格变量
frameworks_overlay_space := $(frameworks_overlay_empty) $(frameworks_overlay_empty)

# 去除空格，将所有“../”合并，组成相对路径前缀
frameworks_overlay_back_dir_prefix := $(strip $(subst $(frameworks_overlay_space),$(framewo
# $(warning $(frameworks_overlay_back_dir_prefix))

frameworks_overlay_root_dir := vendor/iauto/

# 组成Overlay目录路径前缀，得到当前目录到Overlay目录的相对路径
frameworks_overlay_root_prefix := $(addprefix $(frameworks_overlay_back_dir_prefix),$(frame

# 利用warning log输出当前目录路径
$(warning $(LOCAL_PATH))

# 利用当前目录路径，组合成目录深度与当前目录一致的Overlay文件目录路径前缀
# addprefix 为增加前缀的函数
frameworks_overlay_prefix := $(addprefix frameworks_overlay_root_prefix, $(LOCAL_PATH)/)
```

```
$(warning $(frameworks_overlay_prefix))

# 利用Overlay文件目录路径前缀和Android大部分frameworks/base/原生java文件子目录路径，组合成Overlay文件
frameworks_overlay_subdir := $(addsuffix /, $(addprefix $(frameworks_overlay_prefix), $(FRA

# 在vendor/iauto/目录下查找java文件
# all-java-files-under 为查找目录下所有java文件的函数
frameworks_ext_overlay_java_list := $(call all-java-files-under, $(frameworks_overlay_root_

# vendor/iauto/目录下的java文件列表判空，不为空继续，为空则结束Overlay操作
ifneq ($(frameworks_ext_overlay_java_list), $(frameworks_overlay_empty))
$(warning frameworks_ext_overlay_java_list not empty)

# 获取vendor/iauto/目录下的java文件目录结构
frameworks_ext_overlay_dir_list := $(dir $(frameworks_ext_overlay_java_list))

# vendor/iauto/目录下java文件目录结构列表判空，不为空继续，为空则结束Overlay操作
ifneq ($(frameworks_ext_overlay_dir_list), $(frameworks_overlay_empty))
$(warning frameworks_ext_overlay_dir_list not empty)

# 在vendor/iauto/目录下java文件目录结构列表中筛选出需要Overlay的java文件目录，并进行去重复的处理
frameworks_ext_overlay_subdir := $(sort $(strip $(foreach frameworks_dirname, $(frameworks_

# 需要Overlay的java文件目录列表判空，不为空继续，为空则结束Overlay操作
ifneq ($(frameworks_ext_overlay_subdir), $(frameworks_overlay_empty))
$(warning frameworks_ext_overlay_subdir not empty)

# 根据需要Overlay的java文件目录列表查找java文件
# find-other-java-files 为查找java文件的函数
frameworks_ext_overlay_java_file := $(call find-other-java-files, $(frameworks_ext_overlay_

# 将查找出来的Overlay java文件路径添加进参与编译的源文件
LOCAL_SRC_FILES += $(frameworks_ext_overlay_java_file)

# 将Overlay java文件路径中的Overlay文件目录路径前缀用空字符串替换掉，形成被Overlay的原生java文件路径
# subst 为字符串替换函数
frameworks_orign_overlayed_java_files := $(strip $(subst $(frameworks_overlay_prefix), $(fr

# 将 LOCAL_SRC_FILES 中符合被Overlay原生java文件路径的部分去除
# filter-out 为反过滤函数
LOCAL_SRC_FILES := $(filter-out $(frameworks_orign_overlayed_java_files), $(LOCAL_SRC_FILES

$(warning frameworks java overlay ok)

endif

endif

endif

$(warning frameworks java overlay end)

......

# 原生aidl文件被加入参与编译的编译源文件
LOCAL_SRC_FILES += \
        lowpan/java/android/net/lowpan/ILowpanEnergyScanCallback.aidl \
        lowpan/java/android/net/lowpan/ILowpanNetScanCallback.aidl \
        lowpan/java/android/net/lowpan/ILowpanInterfaceListener.aidl \
        lowpan/java/android/net/lowpan/ILowpanInterface.aidl \
        lowpan/java/android/net/lowpan/ILowpanManagerListener.aidl \
        lowpan/java/android/net/lowpan/ILowpanManager.aidl

# frameworks aidl overlay
$(warning frameworks aidl overlay start)

# LOCAL_AIDL_INCLUDES中的目录路径是用来查找aidl接口中import的类的
# LOCAL_PATH、LOCAL_PATH/src、FRAMEWORKS_BASE_JAVA_SRC_DIRS属于默认的查找路径
# 将三个默认路径对应的Overlay目录下的路径也作为默认路径添加进LOCAL_AIDL_INCLUDES
LOCAL_AIDL_INCLUDES += \
      $(addprefix $(frameworks_overlay_root_dir), $(LOCAL_PATH)) \
      $(addprefix $(frameworks_overlay_root_dir), $(LOCAL_PATH)/src) \
      $(addprefix $(frameworks_overlay_root_dir), $(FRAMEWORKS_BASE_JAVA_SRC_DIRS))
# $(warning $(LOCAL_AIDL_INCLUDES))
```

```
# 查找vendor/iauto/目录下所有符合I*.aidl类型（匹配大小写）的文件
$(warning $(frameworks_overlay_root_prefix))
frameworks_ext_overlay_aidl_list := $(call all-Iaidl-files-under, $(frameworks_overlay_root
# $(warning $(frameworks_ext_overlay_aidl_list))

# vendor/iauto/目录下I*.aidl类型文件列表判空，不为空继续，为空则结束Overlay操作
ifneq ($(frameworks_ext_overlay_aidl_list), $(frameworks_overlay_empty))
$(warning frameworks_ext_overlay_aidl_list not empty)

# 将vendor/iauto/目录下I*.aidl类型文件列表转换为对应的原生目录下的路径
frameworks_ext_overlay_aidl_to_orign_list := $(strip $(subst $(frameworks_overlay_prefix),
# $(warning $(frameworks_ext_overlay_aidl_to_orign_list))

# 转换后的原生路径列表判空，不为空继续，为空则结束Overlay操作
ifneq ($(frameworks_ext_overlay_aidl_to_orign_list), $(frameworks_overlay_empty))
$(warning frameworks_ext_overlay_aidl_to_orign_list not empty)

# 利用转换后的原生路径列表从LOCAL_SRC_FILES中筛选出需要Overlay的原生aidl文件路径
frameworks_orign_overlayed_aidl := $(filter $(frameworks_ext_overlay_aidl_to_orign_list), $
```

```
# 需要Overlay的原生aidl文件路径列表判空，不为空继续，为空则结束Overlay操作
ifneq ($(frameworks_orign_overlayed_aidl), $(frameworks_overlay_empty))
$(warning frameworks_orign_overlayed_aidl not empty)
# $(warning $(frameworks_orign_overlayed_aidl))

# 将LOCAL_SRC_FILES中符合需要Overlay的原生aidl文件路径去除
LOCAL_SRC_FILES := $(filter-out $(frameworks_orign_overlayed_aidl), $(LOCAL_SRC_FILES))

# 利用Overlay文件目录路径前缀和需要Overlay的原生aidl文件路径组合成Overlay aidl文件路径，并将其添加进参与
LOCAL_SRC_FILES += $(addprefix $(frameworks_overlay_prefix), $(frameworks_orign_overlayed_a

$(warning frameworks aidl overlay ok)

endif

endif

endif

$(warning frameworks aidl overlay end)

......
```

以上，即是Android Java代码Overlay机制相关的实现方法。

Java Overlay机制的文件Overlay操作方法

已实施Java Overlay机制模块的Overlay操作方法

java文件Overlay操作方法

- 修改的原生java文件Overlay方法
  将修改后的java文件按其原生代码路径放到vendor/iauto/目录下对应路径，原生路径下保留未修改的原生java文件。
  例如
  原生java文件路径为：frameworks/base/core/java/android/content/Context.java
  修改后的Overlay文件路径为：vendor/iauto/frameworks/base/core/java/android/content/Context.java

- 在原生代码中添加的java文件Overlay方法
  将添加的java文件按其在原生代码中路径放到vendor/iauto/目录下对应路径，原生路径下文件删除。

aidl文件Overlay操作方法

- 修改的原生aidl文件Overlay方法
  将修改后的aidl文件按其原生代码路径放到vendor/iauto/目录下对应路径，原生路径下保留未修改的原生aidl文件。

- 在原生代码中添加的aidl文件Overlay方法
  将添加的aidl文件按其在原生代码中路径放到vendor/iauto/目录下对应路径，原生路径下文件删除；
  在对应的Android.mk中的LOCAL_SRC_FILES中添加此aidl文件Overlay目录路径（基于Android.mk文件位置的相对路径）；
  若已添加过此aidl的原生目录相对路径，则直接修改为Overlay目录相对路径。

未实施Java Overlay机制模块的Overlay操作方法

修改Android.mk文件

参照本igitlab中的“实现方法”部分进行模块对应的Overlay仓库目录的创建和模块对应的Android.mk的修改，注意参照样例代码中的注释说明；
一般apk以及单个java代码目录模块通常参照“实现方法\2.1.1 一般apk及单个子目录下的java文件Overlay及aidl文件Overlay”部分。

java文件Overlay操作方法

- 修改的原生java文件Overlay方法
  将修改后的java文件按其原生代码路径放到vendor/iauto/目录下对应路径，原生路径下保留未修改的原生java文件。
  例如
  原生java文件路径为：frameworks/base/core/java/android/content/Context.java
  修改后的Overlay文件路径为：vendor/iauto/frameworks/base/core/java/android/content/Context.java

- 在原生代码中添加的java文件Overlay方法
  将添加的java文件按其在原生代码中路径放到vendor/iauto/目录下对应路径，原生路径下文件删除。

aidl文件Overlay操作方法

- 修改的原生aidl文件Overlay方法
  将修改后的aidl文件按其原生代码路径放到vendor/iauto/目录下对应路径，原生路径下保留未修改的原生aidl文件。

- 在原生代码中添加的aidl文件Overlay方法
  将添加的aidl文件按其在原生代码中路径放到vendor/iauto/目录下对应路径，原生路径下文件删除；
  在对应的Android.mk中的LOCAL_SRC_FILES中添加此aidl文件Overlay目录路径（基于Android.mk文件位置的相对路径）；
  若已添加过此aidl的原生目录相对路径，则直接修改为Overlay目录相对路径。