Получение пути к карте памяти SD Card на Android

Разрабатывая приложение для проведения соревнований, я столкнулся с проблемой хранения базы данных. Проблема состояла в том, как мне определить внешнюю карту памяти. В целом поиск в сети точного ответа не дал. Поэтому, объединив все найденные результаты, я собрал свой класс. Если кому интересно, смотрим под катом.  
  
Итак, начнем с теории.

Терминология

Гугл нам говорит, что есть следующие понятия:

1. Внутренняя (**internal**) память — это часть встроенной в телефон карты памяти. При ее использовании по умолчанию папка приложения защищена от доступа других приложений ([Using the Internal Storage](http://developer.android.com/guide/topics/data/data-storage.html" \l "filesInternal)).
2. Внешняя (**external**) память — это общее «внешнее хранилище», т.е. это может быть как часть встроенной памяти, так и удаляемое устройство. Обычно это часть встроенной памяти, как удаляемое устройство я видел в последний раз на андройде 2.2, где встроенная память была около 2Гб, и подключаемая память становилась внешней ([Using the External Storage](http://developer.android.com/guide/topics/data/data-storage.html" \l "filesExternal)).
3. Удаляемая (**removable**) память — все хранилища, которые могут быть удалены из устройства без «хирургических» вмешательств.

До версии KitKat 4.4 API не предоставляло функционала для получения путей к внешней памяти. Начиная с этой версии (API 19) появилась функция public abstract File[] **getExternalFilesDirs** (String type), которая возвращает массив строк с путями к внутренней и внешней памяти. Но как же быть с нашей SD Card, которая вставлена в слот? Путь к ней мы опять не можем получить.

Результаты поиска

Чтобы ответить на поставленный вопрос я обратился к всезнающему гуглу. Но и он мне не дал четкого ответа. Было рассмотрено множество вариантов определения от использования стандартных функций, которые ведут к внешней памяти, но ничего общего с удаляемыми устройствами хранения данных они не имеют, до обработки правил монтирования устройств (Android же на ядре Linux работает). В последних случаях были использованы «зашитые» пути к папке с примонтироваными устройствами (в различных версиях эта директория разная). Не стоит забывать, что от версии к версии правила монтирования меняются.  
  
В конечном итоге я решил объединить все полученные знания и написал свой класс, который может нам вернуть пути к внешним и удаляемым устройствам.

Описание кода

Был создан класс **MountDevice**, который содержит в себе путь к устройству, тип устройства и некий хэш.  
Типов устройств выделено два (внутреннюю память я не стал трогать, так как к ней доступ можно получить через API системы).

**public** **enum** MountDeviceType {

EXTERNAL\_SD\_CARD, REMOVABLE\_SD\_CARD

}

И был создан класс **StorageHelper**, который и осуществляет поиск доступных карт памяти.  
  
В классе StorageHelper реализовано два способа поиска — через системное окружение (**Environment**) и с использованием утилиты Linux **mount**, а точнее результата ее выполнения.

Способ первый — Environment

При работе с окружением я использую стандартную функцию getExternalStorageDirectory() для получения информации о внешней памяти. Чтобы получить информацию о удаляемой памяти, я использую переменную окружения "*SECONDARY\_STORAGE*".   
  
Внешняя память всегда одна и обычно всегда есть, поэтому проверяем ее на читаемость, вычисляем хэш и запоминаем. Удаляемой памяти может быть много, поэтому необходимо полученную строку разбить по разделителю и проверять каждое значение.

Функция fillDevicesEnvirement

String path = android.os.Environment.getExternalStorageDirectory()

.getAbsolutePath();

**if** (!path.trim().isEmpty()

&& android.os.Environment.getExternalStorageState().equals(

android.os.Environment.MEDIA\_MOUNTED)) {

testAndAdd(path, MountDeviceType.EXTERNAL\_SD\_CARD);

}

*// Получаем ремувабл*

String rawSecondaryStoragesStr = System.getenv("SECONDARY\_STORAGE");

**if** (rawSecondaryStoragesStr != **null**

&& !rawSecondaryStoragesStr.isEmpty()) {

*// All Secondary SD-CARDs splited into array*

**final** String[] rawSecondaryStorages = rawSecondaryStoragesStr

.split(File.pathSeparator);

**for** (String rawSecondaryStorage : rawSecondaryStorages) {

testAndAdd(rawSecondaryStorage,

MountDeviceType.REMOVABLE\_SD\_CARD);

}

}

Вариант решения взят со [stackoverflow](http://stackoverflow.com/questions/5694933/find-an-external-sd-card-location" \l "19831753). Ответ где-то там внизу.

Способ второй — mount

Так как у меня долго не получалось заставить систему мне сказать путь к удаляемой памяти, я решил искать в сторону примонтированных устройств. В системе есть файлы конфигурации, в которых описаны правила монтирования внешних устройств. Все бы хорошо, но на Android версии 4.\* к этому файлу простым смертным доступа нет, поэтому рассматривать этот способ не буду.  
  
Вернемся к утилите mount. При запуске без параметров команда возвращает список смонтированных файловых систем. Удаляемые устройства имеют обычно формат файловой системы FAT, то будем выделять строки, в которых есть характеристика "**fat**". Внешняя память будет характеризоваться параметром "**fuse**".   
  
Примечание: при использовании такого способа не всегда корректно (скорее всего я что-то не учел) определяются типы смотнтированных устройств. Разницу замечал на разных версиях Android. Поэтому этот способ можно использовать как дополнительный.

Функция fillDevicesProcess

**try** {

Runtime runtime = Runtime.getRuntime();

proc = runtime.exec("mount");

**try** {

is = proc.getInputStream();

isr = **new** InputStreamReader(is);

br = **new** BufferedReader(isr);

**while** ((line = br.readLine()) != **null**) {

**if** (line.contains("secure"))

**continue**;

**if** (line.contains("asec"))

**continue**;

**if** (line.contains("fat")) {*// TF card*

String columns[] = line.split(" ");

**if** (columns != **null** && columns.length > 1) {

testAndAdd(columns[1],

MountDeviceType.REMOVABLE\_SD\_CARD);

}

} **else** **if** (line.contains("fuse")) {*// internal(External)*

*// storage*

String columns[] = line.split(" ");

**if** (columns != **null** && columns.length > 1) {

*// mount = mount.concat(columns[1] + "\n");*

testAndAdd(columns[1],

MountDeviceType.EXTERNAL\_SD\_CARD);

}

}

}

} **finally** {

...

}

} **catch** (Exception e) {

...

}

Вариант решения взят со [stackoverflow](http://stackoverflow.com/questions/5694933/find-an-external-sd-card-location" \l "15612964). Ответов там несколько примерно одинаковых.

Про дублирование

Многие замечали в директории монтирования устройств такую картину:

/storage/sdcard0/

/storage/emulated/0/

/storage/emulated/legacy/

И что самое интересно, все это одна и та же внешняя карта памяти. Такое дробление начинается с версии Jelly Bean и сделано это для поддержки многопользовательского режима работы системы. Более подробно [тут](http://android.stackexchange.com/questions/39542/confused-by-the-many-locations-of-the-virtual-sdcard#39546). И вот, чтобы не получать одну и туже карту памяти как различные устройства, необходим способ определения идентичности. Если бы был доступ к конфигурации монтирования, то и вопросов не было. Но доступа нет. Поэтому я [тут](http://stackoverflow.com/questions/5694933/find-an-external-sd-card-location#15612964) подсмотрел решение с расчетом хэша для каждого устройства: 

1. создаем StringBuilder
2. записываем в него общий размер устройства и размер используемого пространства устройства
3. обходим содержимое корня устройства
4. записываем имя каталога
5. записываем имя файла и размер
6. вычисляем hash

Своя функция расчета хэша calcHash

**private** **int** **calcHash**(File dir) {

StringBuilder tmpHash = **new** StringBuilder();

tmpHash.append(dir.getTotalSpace());

tmpHash.append(dir.getUsableSpace());

File[] list = dir.listFiles();

**for** (File file : list) {

tmpHash.append(file.getName());

**if** (file.isFile()) {

tmpHash.append(file.length());

}

}

**return** tmpHash.toString().hashCode();

}

Пример использования

*/\* Получаем базовый путь \*/*

**if** (!mPreferences.contains(PREFS\_BASEBATH)) {

*// Если еще не сохранялся в настройках, то пытаемся найти карты*

*// памяти*

ArrayList<MountDevice> storages = StorageHelper.getInstance()

.getRemovableMountedDevices();

*// проверяем съемные карты памяти*

**if** (storages.size() != 0) {

setBasePath(storages.get(0).getPath() + mAppPath);

} **else** **if** ((storages = StorageHelper.getInstance() *// Проверяем*

*// внутреннюю*

*// память*

.getExternalMountedDevices()).size() != 0) {

setBasePath(storages.get(0).getPath() + mAppPath);

}

} **else** {

*// Вытаскиваем из сохранненых настроек*

mBasePath = mPreferences.getString(PREFS\_BASEBATH, context

.getFilesDir().getParent());

}

Заключение

Подробные рассуждения по этому вопросу понимания памяти в Android, некоторые советы можно прочитать [тут](http://commonsware.com/blog/2014/04/09/storage-situation-removable-storage.html).  
  
Исходный код всего класса ~~расположен~~ еще нигде не расположен. На днях постараюсь разместить на gitHub.  
  
Кто еще какими способами пользуется?

UPD1: Исходный код класса на [bitbucket](https://bitbucket.org/vait/android-helpers/src/4f0abd990ae43aa75fd6eabf377eb4cc5fda2fab/StorageHelper.java?at=master)

*/\*\**

*\* Created 16.03.2015*

*\* CyclingReferee*

*\*/*

**package** name.victorgrek.bike.cyclingreferee.helpers**;**

**import** java.io.BufferedReader**;**

**import** java.io.File**;**

**import** java.io.InputStream**;**

**import** java.io.InputStreamReader**;**

**import** java.util.ArrayList**;**

*/\*\**

*\* @author Victor Grekov victor.grek@gmail.com*

*\**

*\*/*

**public** **class** **StorageHelper** **{**

**private** **static** StorageHelper sStorage**;**

**private** MountDeviceGetter mGetter**;**

**private** **StorageHelper()** **{**

mGetter **=** **this.**new MountDeviceGetter**();**

mGetter**.**fillDevicesEnvirement**();**

**}**

**public** ArrayList**<**MountDevice**>** **getAllMountedDevices()** **{**

ArrayList**<**MountDevice**>** mountedDevice **=** **new** ArrayList**<**StorageHelper**.**MountDevice**>(**

mGetter**.**getMountedExternalDevices**());**

mountedDevice**.**addAll**(**mGetter**.**getMountedRemovableDevices**());**

**return** mountedDevice**;**

**}**

**public** ArrayList**<**MountDevice**>** **getExternalMountedDevices()** **{**

**return** mGetter**.**getMountedExternalDevices**();**

**}**

**public** ArrayList**<**MountDevice**>** **getRemovableMountedDevices()** **{**

**return** mGetter**.**getMountedRemovableDevices**();**

**}**

**public** **static** StorageHelper **getInstance()** **{**

**if** **(**sStorage **==** **null)** **{**

sStorage **=** **new** StorageHelper**();**

**}**

**return** sStorage**;**

**}**

**public** **enum** MountDeviceType **{**

EXTERNAL\_SD\_CARD**,** REMOVABLE\_SD\_CARD

**}**

**public** **class** **MountDevice** **{**

**private** MountDeviceType mType**;**

**private** String mPath**;**

**private** Integer mHash**;**

*/\*\**

*\* @return the type*

*\*/*

**public** **final** MountDeviceType **getType()** **{**

**return** mType**;**

**}**

*/\*\**

*\* @return the path*

*\*/*

**public** **final** String **getPath()** **{**

**return** mPath**;**

**}**

*/\*\**

*\* @return the hash*

*\*/*

**public** **final** Integer **getHash()** **{**

**return** mHash**;**

**}**

**public** **MountDevice(**MountDeviceType type**,** String path**,** Integer hash**)** **{**

**super();**

mType **=** type**;**

mPath **=** path**;**

mHash **=** hash**;**

**}**

*/\**

*\* (non-Javadoc)*

*\**

*\* @see java.lang.Object#equals(java.lang.Object)*

*\*/*

@Override

**public** **boolean** **equals(**Object o**)** **{**

**if** **(!**mPath**.**equals**(((**MountDevice**)** o**).**getPath**()))** **{**

**return** mHash**.**equals**(((**MountDevice**)** o**).**getHash**());**

**}**

**return** **true;**

**}**

*/\**

*\* (non-Javadoc)*

*\**

*\* @see java.lang.Object#hashCode()*

*\*/*

@Override

**public** **int** **hashCode()** **{**

**return** mHash**;**

**}**

**}**

**private** **class** **MountDeviceGetter** **{**

**private** ArrayList**<**MountDevice**>** mMountedExternalDevices **=** **null;**

**private** ArrayList**<**MountDevice**>** mMountedRemovableDevices **=** **null;**

**private** **int** **calcHash(**File dir**)** **{**

StringBuilder tmpHash **=** **new** StringBuilder**();**

tmpHash**.**append**(**dir**.**getTotalSpace**());**

tmpHash**.**append**(**dir**.**getUsableSpace**());**

File**[]** list **=** dir**.**listFiles**();**

**for** **(**File file **:** list**)** **{**

tmpHash**.**append**(**file**.**getName**());**

**if** **(**file**.**isFile**())** **{**

tmpHash**.**append**(**file**.**length**());**

**}**

**}**

**return** tmpHash**.**toString**().**hashCode**();**

**}**

**private** **void** **testAndAdd(**String path**,** MountDeviceType type**)** **{**

File root **=** **new** File**(**path**);**

**if** **(**root**.**exists**()** **&&** root**.**isDirectory**()** **&&** root**.**canWrite**())** **{**

MountDevice device **=** **new** MountDevice**(**type**,** path**,** calcHash**(**root**));**

**switch** **(**type**)** **{**

**case** EXTERNAL\_SD\_CARD**:**

**if** **(!**mMountedExternalDevices**.**contains**(**device**))** **{**

mMountedExternalDevices**.**add**(**device**);**

**}**

**break;**

**case** REMOVABLE\_SD\_CARD**:**

**if** **(!**mMountedRemovableDevices**.**contains**(**device**))** **{**

mMountedRemovableDevices**.**add**(**device**);**

**}**

**break;**

**}**

**}**

root **=** **null;**

**}**

*/\*\**

*\* @return the mountedExternalDevices*

*\*/*

**public** ArrayList**<**MountDevice**>** **getMountedExternalDevices()** **{**

**return** mMountedExternalDevices**;**

**}**

*/\*\**

*\* @return the mountedDevices*

*\*/*

**public** ArrayList**<**MountDevice**>** **getMountedRemovableDevices()** **{**

**return** mMountedRemovableDevices**;**

**}**

**public** **void** **fillDevicesEnvirement()** **{**

mMountedExternalDevices **=** **new** ArrayList**<**StorageHelper**.**MountDevice**>(**3**);**

mMountedRemovableDevices **=** **new** ArrayList**<**StorageHelper**.**MountDevice**>(**3**);**

*// получить экстернал*

String path **=** android**.**os**.**Environment**.**getExternalStorageDirectory**()**

**.**getAbsolutePath**();**

**if** **(!**path**.**trim**().**isEmpty**()**

**&&** android**.**os**.**Environment**.**getExternalStorageState**().**equals**(**

android**.**os**.**Environment**.**MEDIA\_MOUNTED**))** **{**

testAndAdd**(**path**,** MountDeviceType**.**EXTERNAL\_SD\_CARD**);**

**}**

*// Получаем ремувабл*

String rawSecondaryStoragesStr **=** System**.**getenv**(**"SECONDARY\_STORAGE"**);**

**if** **(**rawSecondaryStoragesStr **!=** **null**

**&&** **!**rawSecondaryStoragesStr**.**isEmpty**())** **{**

*// All Secondary SD-CARDs splited into array*

**final** String**[]** rawSecondaryStorages **=** rawSecondaryStoragesStr

**.**split**(**File**.**pathSeparator**);**

**for** **(**String rawSecondaryStorage **:** rawSecondaryStorages**)** **{**

testAndAdd**(**rawSecondaryStorage**,**

MountDeviceType**.**REMOVABLE\_SD\_CARD**);**

**}**

**}**

**}**

**public** **void** **fillDevicesProcess()** **{**

mMountedExternalDevices **=** **new** ArrayList**<**StorageHelper**.**MountDevice**>(**3**);**

mMountedRemovableDevices **=** **new** ArrayList**<**StorageHelper**.**MountDevice**>(**3**);**

InputStream is **=** **null;**

InputStreamReader isr **=** **null;**

BufferedReader br **=** **null;**

Process proc **=** **null;**

String line**;**

**try** **{**

Runtime runtime **=** Runtime**.**getRuntime**();**

proc **=** runtime**.**exec**(**"mount"**);**

**try** **{**

is **=** proc**.**getInputStream**();**

isr **=** **new** InputStreamReader**(**is**);**

br **=** **new** BufferedReader**(**isr**);**

**while** **((**line **=** br**.**readLine**())** **!=** **null)** **{**

**if** **(**line**.**contains**(**"secure"**))**

**continue;**

**if** **(**line**.**contains**(**"asec"**))**

**continue;**

**if** **(**line**.**contains**(**"fat"**))** **{***// TF card*

String columns**[]** **=** line**.**split**(**" "**);**

**if** **(**columns **!=** **null** **&&** columns**.**length **>** 1**)** **{**

testAndAdd**(**columns**[**1**],**

MountDeviceType**.**REMOVABLE\_SD\_CARD**);**

**}**

**}** **else** **if** **(**line**.**contains**(**"fuse"**))** **{***// internal(External)*

*// storage*

String columns**[]** **=** line**.**split**(**" "**);**

**if** **(**columns **!=** **null** **&&** columns**.**length **>** 1**)** **{**

*// mount = mount.concat(columns[1] + "\n");*

testAndAdd**(**columns**[**1**],**

MountDeviceType**.**EXTERNAL\_SD\_CARD**);**

**}**

**}**

**}**

**}** **finally** **{**

**if** **(**br **!=** **null)** **{**

br**.**close**();**

**}**

**if** **(**isr **!=** **null)** **{**

isr**.**close**();**

**}**

**if** **(**proc **!=** **null)** **{**

proc**.**destroy**();**

**}**

**if** **(**is **!=** **null)** **{**

is**.**close**();**

**}**

**}**

**}** **catch** **(**Exception e**)** **{**

e**.**printStackTrace**();**

**}**

**}**

**}**

**}**

*// /storage/emulated/0*

*// /storage/emulated/legacy*

*// https://source.android.com/devices/storage/config.html*

*// http://stackoverflow.com/questions/22219312/android-open-external-storage-directorysdcard-for-storing-file*

*// http://stackoverflow.com/questions/11281010/how-can-i-get-external-sd-card-path-for-android-4-0*

*// http://stackoverflow.com/questions/5694933/find-an-external-sd-card-location*