Сохранение файлов на устройстве

Android использует файловую систему, которая похожа на дисковые файловые системы на других платформах. На этой странице описано, как работать с файловой системой Android для чтения и записи файлов с помощью [File](https://developer.android.com/reference/java/io/File.html) API.

[File](https://developer.android.com/reference/java/io/File.html)Объект хорошо работает для чтения или записи больших объемов данных в начале до конца заказа , не пропуская вокруг. Например, это хорошо для файлов изображений или чего-либо, обмениваемого по сети.

Точное местоположение файла, где могут быть сохранены ваши файлы, может различаться для разных устройств, поэтому вы должны использовать методы, описанные на этой странице, для доступа к внутренним и внешним путям хранения, а не к использованию абсолютных путей к файлам.

Чтобы просмотреть файлы на устройстве, вы можете зарегистрировать местоположение файла, предоставляемое такими методами, как [File.getAbsolutePath()](https://developer.android.com/reference/java/io/File.html" \l "getAbsolutePath()), а затем просмотреть файлы устройств с помощью Android- [обозревателя](https://developer.android.com/studio/debug/device-file-explorer.html) Android Studio .

Выберите внутреннее или внешнее хранилище

Все Android-устройства имеют две области хранения файлов: «внутреннее» и «внешнее» хранилище. Эти имена приходят с первых дней Android, когда большинство устройств предлагают встроенную энергонезависимую память (внутреннее хранилище), а также съемный носитель данных, такой как микро SD-карта (внешнее хранилище). Многие устройства теперь делят пространство постоянного хранения на отдельные «внутренние» и «внешние» разделы. Таким образом, даже без съемного носителя данных эти два пространства хранения всегда существуют, и поведение API одинаково независимо от того, удалено ли внешнее хранилище.

Поскольку внешнее хранилище может быть съемным, существуют некоторые различия между этими двумя параметрами следующим образом.

**Внутренняя память:**

* Это всегда доступно.
* Файлы, сохраненные здесь, доступны только вашему приложению.
* Когда пользователь удаляет ваше приложение, система удаляет все файлы вашего приложения из внутреннего хранилища.

Внутреннее хранилище лучше всего, когда вы хотите быть уверенным, что ни пользователь, ни другие приложения не могут получить доступ к вашим файлам.

**Внешнее хранилище:**

* Это не всегда доступно, поскольку пользователь может подключить внешнее хранилище к USB-накопителю и в некоторых случаях удалить его с устройства.
* Он читается в мире, поэтому файлы, сохраненные здесь, могут быть прочитаны вне вашего контроля.
* Когда пользователь удаляет ваше приложение, система удаляет файлы вашего приложения здесь, только если вы сохраните их в каталоге [getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html" \l "getExternalFilesDir(java.lang.String)).

Внешнее хранилище - лучшее место для файлов, для которых не требуются ограничения доступа, а также файлы, которые вы хотите предоставить другим приложениям, или разрешить пользователям доступ к компьютеру.

**Совет.** Несмотря на то, что приложения по умолчанию установлены во внутреннее хранилище, вы можете разрешить установку своего приложения на внешнем хранилище, указав атрибут в своем манифесте. Пользователи оценивают этот параметр, когда размер APK очень большой, и у них есть внешнее пространство для хранения, которое больше, чем внутреннее хранилище. Для получения дополнительной информации см. « [Место установки приложения»](https://developer.android.com/guide/topics/data/install-location.html) .[**android:installLocation**](https://developer.android.com/guide/topics/manifest/manifest-element.html#install)

Сохранение файла на внутренней памяти

Внутренний каталог хранения вашего приложения указывается именем пакета вашего приложения в специальном месте файловой системы Android, к которому можно получить доступ со следующими API.

**Примечание.** В отличие от [внешних каталогов хранилища](https://developer.android.com/training/data-storage/files#WriteExternalStorage) , ваше приложение не требует каких-либо системных разрешений для чтения и записи во внутренние каталоги, возвращаемые этими методами.

Создать файл

При сохранении файла во внутреннем хранилище вы можете приобрести соответствующий каталог как a [File](https://developer.android.com/reference/java/io/File.html), вызвав один из двух методов:

[getFilesDir()](https://developer.android.com/reference/android/content/Context.html#getFilesDir())

Возвращает [File](https://developer.android.com/reference/java/io/File.html)представление внутреннего каталога для вашего приложения.

[getCacheDir()](https://developer.android.com/reference/android/content/Context.html#getCacheDir())

Возвращает [File](https://developer.android.com/reference/java/io/File.html)представление внутреннего каталога для временных файлов кэша вашего приложения. Обязательно удаляйте каждый файл, если он больше не нужен, и используйте разумный размер для объема памяти, который вы используете в любой момент времени, например 1 МБ.

**Внимание:** если система работает с низким уровнем хранения, она может удалить файлы кеша без предупреждения.

Чтобы создать новый файл в одном из этих каталогов, вы можете использовать [File()](https://developer.android.com/reference/java/io/File.html" \l "File(java.io.File,%20java.lang.String))конструктор, передав [File](https://developer.android.com/reference/java/io/File.html)предоставленный одним из указанных выше методов, который указывает ваш внутренний каталог хранилища. Например:

File file = new File(context.getFilesDir(), filename);

Кроме того, вы можете позвонить, [openFileOutput()](https://developer.android.com/reference/android/content/Context.html" \l "openFileOutput(java.lang.String,%20int))чтобы получить запись [FileOutputStream](https://developer.android.com/reference/java/io/FileOutputStream.html) , записываемую в файл во внутреннем каталоге. Например, вот как написать текст в файл:KOTLIN

String filename = "myfile";  
String fileContents = "Hello world!";  
FileOutputStream outputStream;  
  
try {  
    outputStream = openFileOutput(filename, Context.MODE\_PRIVATE);  
    outputStream.write(fileContents.getBytes());  
    outputStream.close();  
} catch (Exception e) {  
    e.printStackTrace();  
}

Обратите внимание, что для этого [openFileOutput()](https://developer.android.com/reference/android/content/Context.html" \l "openFileOutput(java.lang.String,%20int))метода требуется параметр режима файла. Передача [MODE\_PRIVATE](https://developer.android.com/reference/android/content/Context.html" \l "MODE_PRIVATE)делает его конфиденциальным для вашего приложения. Другие параметры режима [MODE\_WORLD\_READABLE](https://developer.android.com/reference/android/content/Context.html" \l "MODE_WORLD_READABLE)и [MODE\_WORLD\_WRITEABLE](https://developer.android.com/reference/android/content/Context.html" \l "MODE_WORLD_WRITEABLE)были устарели с уровня API.1. Начиная с Android 7.0 (API-уровень 24), Android выдает, [SecurityException](https://developer.android.com/reference/java/lang/SecurityException.html)если вы их используете. Если вашему приложению нужно разделить личные файлы с другими приложениями, вместо этого вы должны использовать a [FileProvider](https://developer.android.com/reference/android/support/v4/content/FileProvider.html)с помощью [FLAG\_GRANT\_READ\_URI\_PERMISSION](https://developer.android.com/reference/android/content/Intent.html#FLAG_GRANT_READ_URI_PERMISSION). Дополнительные сведения см. В [разделе Совместное использование файлов](https://developer.android.com/training/secure-file-sharing/index.html) .

На Android 6.0 (уровень API 23) и ниже, другие приложения могут читать ваши внутренние файлы, если вы установите режим файла для чтения в мире. Однако другое приложение должно знать имя вашего приложения и имена файлов. Другие приложения не могут просматривать ваши внутренние каталоги и не имеют доступа к чтению или записи, если вы явно не устанавливаете файлы для чтения или записи. До тех пор, пока вы используете [MODE\_PRIVATE](https://developer.android.com/reference/android/content/Context.html" \l "MODE_PRIVATE)ваши файлы во внутреннем хранилище, они никогда не доступны для других приложений.

### Запись файла кеша

Если вам нужно кэшировать некоторые файлы, вы должны использовать [createTempFile()](https://developer.android.com/reference/java/io/File.html" \l "createTempFile(java.lang.String,%20java.lang.String)). Например, следующий метод извлекает имя файла из a [URL](https://developer.android.com/reference/java/net/URL.html)и создает файл с таким именем во внутреннем каталоге кэша вашего приложения:

private File getTempFile(Context context, String url) {  
    File file;  
    try {  
        String fileName = Uri.parse(url).getLastPathSegment();  
        file = File.createTempFile(fileName, null, context.getCacheDir());  
    } catch (IOException e) {  
        // Error while creating file  
    }  
    return file;  
}

Созданные [createTempFile()](https://developer.android.com/reference/java/io/File.html" \l "createTempFile(java.lang.String,%20java.lang.String))файлы помещаются в каталог кэша, который является приватным для вашего приложения. Вы должны регулярно [удалять файлы, которые](https://developer.android.com/training/data-storage/files#DeleteFile) вам больше не нужны.

**Внимание.** Если система работает с низким уровнем хранения, она может удалить файлы кэша без предупреждения, поэтому перед их чтением убедитесь, что вы проверяете наличие файлов кеша.

Открыть существующий файл

Чтобы прочитать существующий файл, вызовите [openFileInput(name)](https://developer.android.com/reference/android/content/Context.html" \l "openFileInput(java.lang.String)), передав имя файла.

Вы можете получить массив всех имен файлов вашего приложения, позвонив [fileList()](https://developer.android.com/reference/android/content/Context.html" \l "fileList()).

**Совет.** Если вам нужно упаковать файл в приложение, доступное во время установки, сохраните файл в каталоге вашего проекта **res/raw/**. Вы можете открыть эти файлы **[openRawResource()](https://developer.android.com/reference/android/content/res/Resources.html" \l "openRawResource(int))**, передав идентификатор ресурса. Этот метод возвращает то, что вы можете использовать для чтения файла. Вы не можете записать в исходный файл. **R.raw.*filename***[**InputStream**](https://developer.android.com/reference/java/io/InputStream.html)

Открыть каталог

Вы можете открыть каталог во внутренней файловой системе следующими способами:

[getFilesDir()](https://developer.android.com/reference/android/content/Context.html#getFilesDir())

Возвращает [File](https://developer.android.com/reference/java/io/File.html)представление каталога в файловой системе, уникально связанного с вашим приложением.

[getDir(name, mode)](https://developer.android.com/reference/android/content/Context.html#getDir(java.lang.String,%20int))

Создает новый каталог (или открывает существующий каталог) в каталоге уникальной файловой системы вашего приложения. Этот новый каталог появляется внутри каталога, предоставленного [getFilesDir()](https://developer.android.com/reference/android/content/Context.html" \l "getFilesDir()).

[getCacheDir()](https://developer.android.com/reference/android/content/Context.html#getCacheDir())

Возвращает [File](https://developer.android.com/reference/java/io/File.html)представление каталога кеша в файловой системе, которое уникально связано с вашим приложением. Этот каталог предназначен для временных файлов, и его следует регулярно очищать. Система может удалять файлы там, если на диске недостаточно места, поэтому перед их чтением убедитесь, что вы проверяете наличие файлов кеша.

Чтобы создать новый файл в одном из этих каталогов, вы можете использовать[File()](https://developer.android.com/reference/java/io/File.html#File(java.io.File,%20java.lang.String))конструктор, передавая [File](https://developer.android.com/reference/java/io/File.html)объект, предоставленный одним из указанных выше методов, который указывает ваш внутренний каталог хранилища. Например:

Сохранение файла на внешнем хранилище

Использование внешнего хранилища отлично подходит для файлов, которые вы хотите разделить с другими приложениями, или позволить пользователю получить доступ с помощью компьютера.

После того, как вы [запросите разрешения на доступ к хранилищу](https://developer.android.com/training/data-storage/files#ExternalStoragePermissions) и [убедитесь, что хранилище доступно](https://developer.android.com/training/data-storage/files#CheckExternalAvail) , вы можете сохранить два разных типа файлов:

* [Публичные файлы](https://developer.android.com/training/data-storage/files#PublicFiles) : файлы, которые должны быть свободно доступны другим приложениям и пользователю. Когда пользователь удаляет ваше приложение, эти файлы должны оставаться доступными для пользователя. Например, фотографии, снятые вашим приложением или другими загруженными файлами, должны быть сохранены как общедоступные файлы.
* [Частные файлы](https://developer.android.com/training/data-storage/files#PrivateFiles) : файлы, которые по праву принадлежат вашему приложению, и будут удалены, когда пользователь удалит ваше приложение. Хотя эти файлы технически доступны пользователю и другим приложениям, поскольку они находятся на внешнем хранилище, они не предоставляют ценность пользователю вне вашего приложения.

**Внимание** . Внешнее хранилище может стать недоступным, если пользователь удалит SD-карту или подключит устройство к компьютеру. И файлы все еще видны пользователю и другим приложениям, имеющим [**READ\_EXTERNAL\_STORAGE**](https://developer.android.com/reference/android/Manifest.permission.html#READ_EXTERNAL_STORAGE) разрешение. Поэтому, если функциональность вашего приложения зависит от этих файлов или вам необходимо полностью ограничить доступ, вы должны вместо этого записать свои файлы во [внутреннюю память](https://developer.android.com/training/data-storage/files#WriteInternalStorage) .

Запросить разрешения на внешнее хранилище

Для записи в общедоступное внешнее хранилище необходимо запросить[WRITE\_EXTERNAL\_STORAGE](https://developer.android.com/reference/android/Manifest.permission.html#WRITE_EXTERNAL_STORAGE)разрешение в [файле манифеста](https://developer.android.com/guide/topics/manifest/manifest-intro.html) :

<manifest ... > <uses-permission android: name = "android.permission.WRITE\_EXTERNAL\_STORAGE" />     ... </ manifest>  
   

**Примечание.** Если ваше приложение использует это [**WRITE\_EXTERNAL\_STORAGE**](https://developer.android.com/reference/android/Manifest.permission.html#WRITE_EXTERNAL_STORAGE) разрешение, оно также имеет разрешение на чтение внешнего хранилища.

Если вашему приложению нужно только прочитать внешнее хранилище (но не писать ему), вам необходимо объявить [READ\_EXTERNAL\_STORAGE](https://developer.android.com/reference/android/Manifest.permission.html" \l "READ_EXTERNAL_STORAGE)разрешение:

<manifest ... > <uses-permission android: name = "android.permission.READ\_EXTERNAL\_STORAGE" />     ... </ manifest>  
   

Начиная с Android 4.4 (API уровня 19), чтение или запись файлов в приватном внешнем хранилище вашего приложения, доступ к которому осуществляется с помощью [getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html#getExternalFilesDir(java.lang.String))-does, не требующих [READ\_EXTERNAL\_STORAGE](https://developer.android.com/reference/android/Manifest.permission.html#READ_EXTERNAL_STORAGE) или [WRITE\_EXTERNAL\_STORAGE](https://developer.android.com/reference/android/Manifest.permission.html#WRITE_EXTERNAL_STORAGE) разрешений. Поэтому, если ваше приложение поддерживает Android 4.3 (уровень API 18) и ниже, и вы хотите получить доступ только к частному внешнему хранилищу, вы должны заявить, что разрешение запрашивается только в более низких версиях Android, добавив [maxSdkVersion](https://developer.android.com/guide/topics/manifest/uses-permission-element.html" \l "maxSdk) атрибут:

<manifest ... > <uses-permission android: name = "android.permission.WRITE\_EXTERNAL\_STORAGE" android: maxSdkVersion = "18" />     ... </ manifest>  
      
                     

Убедитесь, что имеется внешнее хранилище

Поскольку внешнее хранилище может быть недоступно, например, когда пользователь установил хранилище на ПК или удалил SD-карту, которая обеспечивает внешнее хранилище, вы всегда должны убедиться, что этот том доступен перед его доступом. Вы можете запросить внешнее состояние хранилища, вызвав [getExternalStorageState()](https://developer.android.com/reference/android/os/Environment.html" \l "getExternalStorageState()). Если возвращается состояние [MEDIA\_MOUNTED](https://developer.android.com/reference/android/os/Environment.html#MEDIA_MOUNTED), то вы можете читать и писать свои файлы. Если это так [MEDIA\_MOUNTED\_READ\_ONLY](https://developer.android.com/reference/android/os/Environment.html#MEDIA_MOUNTED_READ_ONLY), вы можете читать только файлы.

Например, для определения доступности хранилища полезны следующие методы:

/\* Checks if external storage is available for read and write \*/  
public boolean isExternalStorageWritable() {  
    String state = Environment.getExternalStorageState();  
    if (Environment.MEDIA\_MOUNTED.equals(state)) {  
        return true;  
    }  
    return false;  
}  
  
/\* Checks if external storage is available to at least read \*/  
public boolean isExternalStorageReadable() {  
    String state = Environment.getExternalStorageState();  
    if (Environment.MEDIA\_MOUNTED.equals(state) ||  
        Environment.MEDIA\_MOUNTED\_READ\_ONLY.equals(state)) {  
        return true;  
    }  
    return false;  
}

### Сохранить в общедоступный каталог

Если вы хотите сохранить общедоступные файлы во внешнем хранилище, используйте этот[getExternalStoragePublicDirectory()](https://developer.android.com/reference/android/os/Environment.html#getExternalStoragePublicDirectory(java.lang.String))метод, чтобы получить [File](https://developer.android.com/reference/java/io/File.html)представление соответствующего каталога на внешнем хранилище. Метод принимает аргумент , указывающий тип файла , который вы хотите сохранить , так что они могут быть логически организованы с другими общественными файлами, такими как [DIRECTORY\_MUSIC](https://developer.android.com/reference/android/os/Environment.html" \l "DIRECTORY_MUSIC)или [DIRECTORY\_PICTURES](https://developer.android.com/reference/android/os/Environment.html#DIRECTORY_PICTURES). Например:

public File getPublicAlbumStorageDir(String albumName) {  
    // Get the directory for the user's public pictures directory.  
    File file = new File(Environment.getExternalStoragePublicDirectory(  
            Environment.DIRECTORY\_PICTURES), albumName);  
    if (!file.mkdirs()) {  
        Log.e(LOG\_TAG, "Directory not created");  
    }  
    return file;  
}

Если вы хотите скрыть свои файлы из Media Scanner, включите пустой файл, указанный .nomediaв вашем каталоге внешних файлов (обратите внимание на префикс точки в имени файла). Это предотвращает чтение медиа-сканера медиафайлами и предоставление их другим приложениям через [MediaStore](https://developer.android.com/reference/android/provider/MediaStore.html) поставщика контента.

### Сохранить в личный каталог

Если вы хотите сохранить файлы на внешнем хранилище, которые являются приватными для вашего приложения и недоступны [MediaStore](https://developer.android.com/reference/android/provider/MediaStore.html)поставщику контента, вы можете приобрести каталог, который используется только вашим приложением, вызывая [getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html" \l "getExternalFilesDir(java.lang.String))и передавая ему имя, указывающее тип нужной вам директории , Каждый созданный таким образом каталог добавляется в родительский каталог, который инкапсулирует все внешние файлы хранения вашего приложения, которые система удаляет, когда пользователь удаляет ваше приложение.

**Внимание: файлы на внешнем хранилище не всегда доступны** , поскольку пользователи могут подключать внешнее хранилище к компьютеру для использования в качестве устройства хранения. Поэтому, если вам нужно хранить файлы, которые имеют решающее значение для функциональности вашего приложения, вы должны хранить их на [внутренней памяти](https://developer.android.com/training/data-storage/files#WriteInternalStorage) .

Например, вот метод, который вы можете использовать для создания каталога для отдельного фотоальбома:

public File getPrivateAlbumStorageDir(Context context, String albumName) {  
    // Get the directory for the app's private pictures directory.  
    File file = new File(context.getExternalFilesDir(  
            Environment.DIRECTORY\_PICTURES), albumName);  
    if (!file.mkdirs()) {  
        Log.e(LOG\_TAG, "Directory not created");  
    }  
    return file;  
}

Если ни один из предопределенных имен подкаталогов не подходит для ваших файлов, вы можете вместо этого позвонить [getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html" \l "getExternalFilesDir(java.lang.String))и пройти null. Это возвращает корневой каталог для частного каталога вашего приложения на внешнем хранилище.

Помните, что [getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html" \l "getExternalFilesDir(java.lang.String)) создает каталог, который удаляется, когда пользователь удаляет ваше приложение. Если файлы, которые вы сохраняете, должны оставаться доступными после того, как пользователь удалит ваше приложение, например, когда ваше приложение захватывает фотографии, а пользователь должен сохранить эти фотографии - вместо этого вы должны [сохранить файлы в общем каталоге](https://developer.android.com/training/data-storage/files#PublicFiles) .

Независимо от того, используете ли вы [getExternalStoragePublicDirectory()](https://developer.android.com/reference/android/os/Environment.html" \l "getExternalStoragePublicDirectory(java.lang.String))файлы, которые являются общими, или [getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html" \l "getExternalFilesDir(java.lang.String))для файлов, которые являются приватными для вашего приложения, важно, чтобы вы использовали имена каталогов, предоставляемые константами API [DIRECTORY\_PICTURES](https://developer.android.com/reference/android/os/Environment.html#DIRECTORY_PICTURES). Эти имена каталогов гарантируют правильную обработку файлов системой. Например, файлы, сохраненные в [DIRECTORY\_RINGTONES](https://developer.android.com/reference/android/os/Environment.html" \l "DIRECTORY_RINGTONES)системе, классифицируются системным медиаплеером как мелодии, а не музыка.

#### Выбрать между несколькими местами хранения

Иногда устройство, которое выделяет раздел внутренней памяти для использования в качестве внешнего хранилища, также предоставляет слот для SD-карты. Это означает, что устройство имеет два разных внешних каталога хранения, поэтому вам нужно выбрать, какой из них использовать при записи «частных» файлов во внешнее хранилище.

Начиная с Android 4.4 (API уровня 19), вы можете получить доступ к обоим местоположениям по вызову [getExternalFilesDirs()](https://developer.android.com/reference/android/content/Context.html" \l "getExternalFilesDirs(java.lang.String)), который возвращает [File](https://developer.android.com/reference/java/io/File.html)массив с записями для каждого места хранения. Первая запись в массиве считается основным внешним хранилищем, и вы должны использовать это местоположение, если оно не будет полным или недоступным.

Если ваше приложение поддерживает Android 4.3 и ниже, вы должны использовать статический метод библиотеки поддержки [ContextCompat.getExternalFilesDirs()](https://developer.android.com/reference/android/support/v4/content/ContextCompat.html" \l "getExternalFilesDirs(android.content.Context,%20java.lang.String)). Это всегда возвращает [File](https://developer.android.com/reference/java/io/File.html)массив, но если устройство работает под управлением Android 4.3 и ниже, оно содержит только одну запись для основного внешнего хранилища (если есть второе место хранения, вы не можете получить к нему доступ на Android 4.3 и ниже).

## Свободное пространство запроса

Если вы заранее знаете, сколько данных вы сохраняете, вы можете узнать, доступно ли достаточное пространство, не вызывая [IOException](https://developer.android.com/reference/java/io/IOException.html)вызова [getFreeSpace()](https://developer.android.com/reference/java/io/File.html" \l "getFreeSpace())или [getTotalSpace()](https://developer.android.com/reference/java/io/File.html" \l "getTotalSpace()). Эти методы обеспечивают текущее доступное пространство и общее пространство в объеме хранилища, соответственно. Эта информация также полезна, чтобы избежать заполнения объема хранилища выше определенного порога.

Однако система не гарантирует, что вы можете записать столько байтов, сколько указано [getFreeSpace()](https://developer.android.com/reference/java/io/File.html" \l "getFreeSpace()). Если число возвращено на несколько Мбайт больше, чем размер данных, которые вы хотите сохранить, или если файловая система заполнена на 90%, тогда все в порядке. В противном случае вы, вероятно, не должны писать на хранение.

**Примечание.** Перед сохранением файла вам не требуется проверять объем доступного пространства. Вместо этого вы можете попытаться записать файл сразу, а затем поймать, **[IOException](https://developer.android.com/reference/java/io/IOException.html)**если это произойдет. Возможно, вам понадобится сделать это, если вы не знаете, сколько места вам нужно. Например, если вы изменили кодировку файла, прежде чем сохранить его, преобразов PNG-изображение в JPEG, вы не будете знать размер файла заранее.

## Удалить файл

Вы всегда должны удалять файлы, которые больше не нужны вашему приложению. Самый простой способ удалить файл, чтобы позвонить [delete()](https://developer.android.com/reference/java/io/File.html" \l "delete())на [File](https://developer.android.com/reference/java/io/File.html)объект.

myFile.delete();

Если файл сохранен во внутреннем хранилище, вы также можете попросить [Context](https://developer.android.com/reference/android/content/Context.html) его найти и удалить, вызвав [deleteFile()](https://developer.android.com/reference/android/content/Context.html" \l "deleteFile(java.lang.String)):

myContext.deleteFile(fileName);