Сохранение файлов

Содержание этого урока

1. [Выбор внутреннего или внешнего хранилища](https://developer.android.com/training/basics/data-storage/files.html?hl=ru#InternalVsExternalStorage)
2. [Получение разрешений на доступ к внешнему хранилищу](https://developer.android.com/training/basics/data-storage/files.html?hl=ru#GetWritePermission)
3. [Сохранение файла во внутреннем хранилище](https://developer.android.com/training/basics/data-storage/files.html?hl=ru#WriteInternalStorage)
4. [Сохранение файла во внешнем хранилище](https://developer.android.com/training/basics/data-storage/files.html?hl=ru#WriteExternalStorage)
5. [Запрос свободного пространства](https://developer.android.com/training/basics/data-storage/files.html?hl=ru#GetFreeSpace)
6. [Удаление файла](https://developer.android.com/training/basics/data-storage/files.html?hl=ru#DeleteFile)

См. также:

* [Использование внутреннего хранилища](https://developer.android.com/guide/topics/data/data-storage.html?hl=ru#filesInternal)
* [Использование внешнего хранилища](https://developer.android.com/guide/topics/data/data-storage.html?hl=ru#filesExternal)

Операционная система Android использует файловую систему, похожую на дисковые файловые системы других платформ. В этом уроке рассказывается, как работать с файловой системой Android для чтения и записи файлов в [File](https://developer.android.com/reference/java/io/File.html?hl=ru) API-интерфейсах.

Объект [File](https://developer.android.com/reference/java/io/File.html?hl=ru) подходит для чтения или записи больших объемов данных в порядке от начала к концу без пропусков. Например, этот способ оптимален для изображений или любых других файлов, передаваемых по сети.

В этом уроке вы узнаете, как выполнять в приложении базовые задачи, связанные с файлами. Для прохождения урока вы должны быть знакомы с основами файловой системы Linux и стандартными файловыми API-интерфейсами ввода/вывода в [java.io](https://developer.android.com/reference/java/io/package-summary.html?hl=ru).

Выбор внутреннего или внешнего хранилища

Все устройства Android имеют две области хранения файлов: внутренняя память и внешние хранилища. Эти области появились в первые годы существования Android, когда на большинстве устройств имелись встроенная память (внутреннее хранилище) и карты памяти (например micro SD, внешнее хранилище). Некоторые устройства делят встроенную память на внутренний и внешний разделы, так что даже без съемных носителей в системе две области хранения файлов, и API-интерфейс работает одинаково вне зависимости от типа внешнего хранилища. Ниже подробно описаны обе области хранения файлов.

**Внутренняя память**

* Всегда доступна.
* Сохраненные здесь файлы по умолчанию доступны только вашему приложению.
* При удалении пользователем вашего приложения система Android удаляет из внутренней памяти все файлы этого приложения.

Внутренняя память лучше всего подходит для ситуаций, когда вы хотите быть уверены, что ни пользователь, ни другие приложения не смогут получить доступ к вашим файлам.

**Внешнее хранилище**

* Доступно не всегда, потому что пользователь может в любое время подключать и отключать такие хранилища, например, USB-накопители.
* Такие хранилища доступны для чтения везде, поэтому вы не контролируете чтение сохраненных в них данных.
* При удалении пользователем вашего приложения система Android удаляет из внешних хранилищ файлы этого приложения, только если они сохраняются в директории из [getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html?hl=ru" \l "getExternalFilesDir(java.lang.String)).

Внешнее хранилище лучше всего подходит для файлов без ограничений доступа и для файлов, которые вы хотите сделать доступными другим приложениям или пользователю через компьютер.

**Совет.** Хотя по умолчанию приложения устанавливаются во внутреннюю память, вы можете указать в манифесте атрибут[android:installLocation](https://developer.android.com/guide/topics/manifest/manifest-element.html?hl=ru#install), чтобы приложение можно было установить во внешнее хранилище. Этот вариант будет полезен пользователям при большом размере файла APK, либо если у них есть доступ к внешнему хранилищу, объем которого превышает внутреннюю память. Дополнительную информацию см. в разделе [Место установки приложения](https://developer.android.com/guide/topics/data/install-location.html?hl=ru).

Получение разрешений доступа к внешнему хранилищу

Для записи во внешнее хранилище следует указать запрос разрешения [WRITE\_EXTERNAL\_STORAGE](https://developer.android.com/reference/android/Manifest.permission.html?hl=ru#WRITE_EXTERNAL_STORAGE) в [файле манифеста](https://developer.android.com/guide/topics/manifest/manifest-intro.html?hl=ru):

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

**Внимание!** В настоящее время все приложения могут считывать данные из внешних хранилищ без специального разрешения. Однако в новой версии эта ситуация будет изменена. Если вашему приложению потребуется считать данные из внешнего хранилища (но не записать туда данные), вам необходимо будет декларировать разрешение [READ\_EXTERNAL\_STORAGE](https://developer.android.com/reference/android/Manifest.permission.html?hl=ru#READ_EXTERNAL_STORAGE). Чтобы обеспечить дальнейшую работу вашего приложения ожидаемым образом, вы должны сразу декларировать это разрешение до того, как изменения вступят в силу.

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

Однако если ваше приложение использует разрешение [WRITE\_EXTERNAL\_STORAGE](https://developer.android.com/reference/android/Manifest.permission.html?hl=ru#WRITE_EXTERNAL_STORAGE), оно косвенно получает разрешение на чтение данных из внешнего хранилища.

Для сохранения файлов во внутреннем хранилище не требуется никаких разрешений. У вашего приложения всегда будет разрешение на чтение и запись файлов в его каталог внутренней памяти.

Сохранение файла во внутренней памяти

При сохранении файла во внутреннюю память вы можете получить соответствующую директорию в виде [File](https://developer.android.com/reference/java/io/File.html?hl=ru), вызвав один из двух методов:

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

Возвращает [File](https://developer.android.com/reference/java/io/File.html?hl=ru), соответствующий внутренней директории приложения.

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

Возвращает [File](https://developer.android.com/reference/java/io/File.html?hl=ru), соответствующий внутренней директории файлов временной кэш-памяти приложения. Обязательно удаляйте каждый файл, когда он перестанет быть нужным, и устанавливайте разумное ограничение размера памяти, используемой в каждый момент времени, например, 1 МБ. Если системе будет не хватать места в хранилище, она может удалять файлы из кэш-памяти без уведомления.

Для создания файла в одной из этих директорий можно использовать конструктор [File()](https://developer.android.com/reference/java/io/File.html?hl=ru" \l "File(java.io.File, java.lang.String)), который передает элемент [File](https://developer.android.com/reference/java/io/File.html?hl=ru), предоставляемый одним из вышеприведенных методов, с помощью которого указывается директория во внутренней памяти. Например:

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

Кроме того, можно вызвать метод [openFileOutput()](https://developer.android.com/reference/android/content/Context.html?hl=ru" \l "openFileOutput(java.lang.String, int)) для получения объекта [FileOutputStream](https://developer.android.com/reference/java/io/FileOutputStream.html?hl=ru) , производящего запись в файл во внутренней памяти. Вот пример записи текста в файл:

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

Если вам потребуется кэшировать какие-то файлы, используйте [createTempFile()](https://developer.android.com/reference/java/io/File.html?hl=ru" \l "createTempFile(java.lang.String, java.lang.String)). Например, следующий метод извлекает имя файла из[URL](https://developer.android.com/reference/java/net/URL.html?hl=ru) и создает файл с этим именем в каталоге внутренней кэш-памяти вашего приложения:

public 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;  
}

**Примечание.** Каталог вашего приложения во внутренней памяти указывается с использованием имени пакета приложения в определенном месте файловой системы Android. Технически другое приложение может прочитать ваши файлы во внутренней памяти, если вы установите для файлов режим Readable (доступно для чтения). Однако для этого другому приложению должны быть известны имя пакета вашего приложения и имена файлов. Другие приложения не могут просматривать внутренние каталоги вашего приложения и не имеют разрешений на чтение или запись, если вы специально не установите для своих файлов режим Readable (доступно для чтения) или Writable (доступно для записи). Следовательно, пока вы будете использовать режим [MODE\_PRIVATE](https://developer.android.com/reference/android/content/Context.html?hl=ru#MODE_PRIVATE) для своих файлов во внутренней памяти, они будут недоступны другим приложениям.

Сохранение файла во внешнем хранилище

Поскольку внешнее хранилище может быть недоступно— например, если пользователь установил хранилище в гнездо на компьютере или извлек из устройства SD-карту, — перед доступом к тому всегда следует проверять его доступность. Состояние внешнего хранилища можно узнать, если вызвать [getExternalStorageState()](https://developer.android.com/reference/android/os/Environment.html?hl=ru" \l "getExternalStorageState()). Если возвращаемое состояние равнозначно [MEDIA\_MOUNTED](https://developer.android.com/reference/android/os/Environment.html?hl=ru#MEDIA_MOUNTED), вы сможете выполнять с файлами операции чтения и записи. Например, следующие методы будут полезными для определения доступности хранилища данных:

/\* 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;  
}

Хотя внешнее хранилище может быть изменено пользователем и другими приложениями, существует две категории файлов, которые в нем можно сохранять:

Общедоступные файлы

Файлы, которые должны быть доступны другим приложениям и пользователю. Когда пользователь удаляет ваше приложение, эти файлы должны оставаться доступны пользователю.

Например, в эту категорию входят снимки, сделанные с помощью вашего приложения, а также другие загружаемые файлы.

Личные файлы

Это файлы, принадлежащие вашему приложению. Они должны удаляться при удалении вашего приложения пользователем. Хотя технически эти файлы доступны для пользователя и других приложений, поскольку находятся во внешнем хранилище, они не имеют никакой ценности для пользователей вне вашего приложения. Когда пользователь удаляет ваше приложение, система удаляет все файлы из каталога закрытых файлов вашего приложения во внешнем хранилище.

Например, к этой категории относятся дополнительные ресурсы, загруженные приложением, и временные мультимедийные файлы.

Если вы хотите сохранить публичные файлы во внешнем хранилище, используйте методы [getExternalStoragePublicDirectory()](https://developer.android.com/reference/android/os/Environment.html?hl=ru" \l "getExternalStoragePublicDirectory(java.lang.String)) для получения [File](https://developer.android.com/reference/java/io/File.html?hl=ru), отражающего соответствующий каталог во внешнем хранилище. Этот метод принимает аргумент, указывающий тип сохраняемого файла и позволяющий включить его в логическую структуру с другими публичными файлами, например, [DIRECTORY\_MUSIC](https://developer.android.com/reference/android/os/Environment.html?hl=ru#DIRECTORY_MUSIC) или[DIRECTORY\_PICTURES](https://developer.android.com/reference/android/os/Environment.html?hl=ru#DIRECTORY_PICTURES). Например:

public File getAlbumStorageDir(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 сохранить личные файлы вашего приложения, вы можете получить соответствующий каталог посредством, вызвав метод[getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html?hl=ru#getExternalFilesDir(java.lang.String)) и предоставив ему имя с указанием желаемого типа каталога. Каждый каталог, создаваемый таким способом, добавляется в родительский каталог, в котором объединены все файлы вашего приложения во внешнем хранилище. При удалении вашего приложения пользователем система удаляет этот каталог.

Например, следующий метод позволит вам создать каталог для персонального фотоальбома:

public File getAlbumStorageDir(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 одно из готовых имен подкаталогов не подходит для ваших файлов, вы можете вызвать [getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html?hl=ru" \l "getExternalFilesDir(java.lang.String)) и передать аргумент null. При этом будет возвращен корневой каталог закрытого каталога вашего приложения во внешнем хранилище.

Следует помнить, что [getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html?hl=ru" \l "getExternalFilesDir(java.lang.String)) создает каталог внутри каталога, который удаляется при удалении вашего приложения пользователем. Если сохраняемые вами файлы должны оставаться доступными после удаления вашего приложения пользователем — например, если ваше приложение работает с камерой, а пользователь хочет сохранить снимки, — вам следует использовать[getExternalStoragePublicDirectory()](https://developer.android.com/reference/android/os/Environment.html?hl=ru#getExternalStoragePublicDirectory(java.lang.String)).

Вне зависимости от того, используете ли вы [getExternalStoragePublicDirectory()](https://developer.android.com/reference/android/os/Environment.html?hl=ru" \l "getExternalStoragePublicDirectory(java.lang.String)) для общих файлов или link android.content.Context#getExternalFilesDir getExternalFilesDir()} для собственных файлов приложения, вы должны использовать имена каталогов, предоставляемые постоянными значениями API-интерфейсов, например, [DIRECTORY\_PICTURES](https://developer.android.com/reference/android/os/Environment.html?hl=ru#DIRECTORY_PICTURES). Эти имена каталогов обеспечивают правильную обработку файлов системой. Например, сохраненные в [DIRECTORY\_RINGTONES](https://developer.android.com/reference/android/os/Environment.html?hl=ru#DIRECTORY_RINGTONES) файлы относятся медиа-сканером системы в категорию рингтонов, а не музыки.

Запрос доступного пространства

Если вам заранее известен объем сохраняемых данных, вы можете определить наличие достаточного пространства без исключения[IOException](https://developer.android.com/reference/java/io/IOException.html?hl=ru), вызвав метод [getFreeSpace()](https://developer.android.com/reference/java/io/File.html?hl=ru" \l "getFreeSpace()) или [getTotalSpace()](https://developer.android.com/reference/java/io/File.html?hl=ru" \l "getTotalSpace()). Эти методы позволяют узнать текущее доступное пространство и общее пространство в хранилище. Эта информация также позволять избежать переполнения объема хранилища сверх определенного уровня.

Однако система не гарантирует возможность записи такого же количества байт, как указано в [getFreeSpace()](https://developer.android.com/reference/java/io/File.html?hl=ru" \l "getFreeSpace()). Если выводимое число на несколько мегабайт превышает размер данных, которые вы хотите сохранить, или если файловая система заполнена менее, чем на 90%, дальше можно действовать спокойно. В противном случае запись в хранилище осуществлять нежелательно.

**Примечание.** Вам необязательно проверять объем доступного места перед сохранением файла. Вы можете попробовать сразу записать файл, а затем определить событие [IOException](https://developer.android.com/reference/java/io/IOException.html?hl=ru), если оно возникнет. Это может потребоваться, если вы точно не знаете, сколько нужно свободного места. Например, если вы меняете кодировку файла перед сохранением при конвертации изображения PNG в формат JPEG, вы не будете знать размер файла заранее.

Удаление файла

Ненужные файлы всегда следует удалять. Наиболее простой способ удалить файл – вызвать в открытом файле ссылку [delete()](https://developer.android.com/reference/java/io/File.html?hl=ru" \l "delete()) на сам этот файл.

myFile.delete();

Если файл сохранен во внутреннем хранилище, вы также можете запросить [Context](https://developer.android.com/reference/android/content/Context.html?hl=ru), чтобы найти и удалить файл посредством вызова[deleteFile()](https://developer.android.com/reference/android/content/Context.html?hl=ru#deleteFile(java.lang.String)):

myContext.deleteFile(fileName);

**Примечание.** При удалении пользователем вашего приложения система Android удаляет следующие элементы:

* Все файлы, сохраненные во внутреннем хранилище
* Все файлы, сохраненные во внешнем хранилище с использованием [getExternalFilesDir()](https://developer.android.com/reference/android/content/Context.html?hl=ru" \l "getExternalFilesDir(java.lang.String)).

Однако вам следует регулярно вручную очищать кэш-память, чтобы удалить файлы, созданные с помощью [getCacheDir()](https://developer.android.com/reference/android/content/Context.html?hl=ru" \l "getCacheDir()), а также удалять любые другие ненужные файлы.