File Access

Reference: https://developer.android.com/guide/topics/data
https://developer.android.com/training/data-storage
https://developer.android.com/training/data-storage/app-specific
https://developer.android.com/training/data-storage/shared/documents-files
https://developer.android.com/reference/android/provider/MediaStore
https://developer.android.com/reference/java/io/File
https://developer.android.com/reference/androidx/core/content/FileProvider
https://gist.github.com/granoeste/5574148

Contents:

- View files on a device
- Storage location paths:
  - App's private internal storage
  - App's private external storage (this is NOT on the removable SD card)
  - App's private external storage (this is on the removable SD card)
  - Public storage folder Downloads/Documents (this is NOT on the removable SD card)
- Permission request
- Get file path and usage
- Write file
- Write a bitmap picture file
- Read file
- Read and load a bitmap picture file
- File or Directory exists
- File delete
- File copy
- Rename file
- List files in directory
- Create directory
- Copy Directory
- Delete Directory Recursively
- Intents
- URI parts
- AssetManager

View files on a device

To view the files stored on a device, use Android Studio's **Device File Explorer**
https://developer.android.com/studio/debug/device-file-explorer

To refresh a folder's contents, right-click on the folder and select **Synchronize** from the drop-down menu. Three folders that are of interest are:

- **data/data/<app's package name>/** – files for your app stored on internal storage

- **sdcard/** – files stored on external user storage
- **storage/** – files stored on removable SD card

App's private internal storage

Permanent (deleted when app is uninstalled)

- Method to get the path: **File path = getFilesDir();**
- Path returned by method: **/data/user/0/<app's package name>/files**
- Path using the Device File Explorer: **/data/data/<app's package name>/files**
- https://developer.android.com/reference/android/content/Context#getFilesDir()

Cache (deleted anytime by system)

- Method to get the path: **File path = getCacheDir()**;
- Path returned by method: **/data/user/0/<app's package name>/cache**
- Path using the Device File Explorer: **/data/data/<app's package name>/cache**
- https://developer.android.com/reference/android/content/Context#getCacheDir()

App's private external storage (this is NOT on the removable SD card)

Permanent (deleted when app is uninstalled)

- Method to get the path: **File path = getExternalFilesDir(null)**;
- Path returned by method: **/storage/emulated/0/Android/data/<app's package name>/files**
- Path using the Device File Explorer: **/sdcard/Android/data/<app's package name>/files**
- https://developer.android.com/reference/android/content/Context#getExternalFilesDir(java.lang.String)

Cache (deleted anytime by system)

- Method to get the path: **File path = getExternalCacheDir()**;
- Path returned by method: **/storage/emulated/0/Android/data/<app's package name>/cache**
- Path using the Device File Explorer: **/sdcard/Android/data/<app's package name>/cache**

???

- Method to get the path: **File path = Environment.getExternalStorageDirectory();**
- Path returned by method: **/storage/emulated/0**

Media

- Method to get the path: **File[] path = getExternalMediaDirs();**
- Path returned by method: **/storage/emulated/0/Android/media/<app's package name>/**
- Path using the Device File Explorer: **/sdcard/Android/media/<app's package name>/**

App's private external storage (this is on the removable SD card)

Permanent (deleted when app is uninstalled)

- Method to get the path: **File paths[] = getExternalFilesDirs("");** (note the plural in **Dirs**)
  If paths.length() > 1 then the second (path[1]) and subsequent entries are locations on the removable SD card. path[0] is the same as getExternalFilesDir(). (no plural)
- Path returned by method: **/storage/AF76-101D/Android/data/<app's package name>/files**

- Path using the Device File Explorer: **/storage/AF76-101D/Android/data/<app's package name>/files**
- Note that the ID AF76-101D will be different
- These files can be seen in Windows' File Manager under **Android/data/<app's package name>/files**
- https://developer.android.com/reference/android/content/Context#getExternalFilesDirs(java.lang.String)

Cache (deleted anytime by system)

- Method to get the path: **File paths[] = getExternalCacheDirs("");** (note the plural in **Dirs**)
  If paths.length() > 1 then the second (path[1]) and subsequent ones are locations on the removable SD card. path[0] is still the same as getExternalFilesDir(). (no plural)
- Path returned by method: **/storage/AF76-101D/Android/data/<app's package name>/cache**
- Path using the Device File Explorer: **/storage/AF76-101D/Android/data/<app's package name>/cache**
- Note that the ID AF76-101D will be different
- These files can be seen in Windows' File Manager under **Android/data/<app's package name>/cache**
- https://developer.android.com/reference/android/content/Context#getExternalCacheDirs()

Public storage folder Downloads/Documents (this is NOT on the removable SD card)

Download folder

- Methods to get the path: **File path = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);**
- Path returned by method: **/storage/emulated/0/Download**
- Path using the Device File Explorer: **/sdcard/Download/**
  or **/storage/self/primary/Download**

Documents folder

- Methods to get the path: **File path = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOCUMENTS);**
- Path returned by method: **/storage/emulated/0/Documents**
- Path using the Device File Explorer: **/sdcard/ Documents /**
  or **/storage/self/primary/ Documents**

DCIM folder

- Methods to get the path: **File path = Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_ DCIM);**

Root folder

- Methods to get the path: **File path = Environment.getExternalStorageDirectory();**
- Methods to get the path: **File path = Environment.getExternalStorageDirectory().getAbsolutePath();**
- Path returned by method: **/storage/emulated/0**
- Cannot access???

App's directory structure

https://medium.com/mobile-app-development-publication/assets-or-resource-raw-folder-of-android-5bdc042570e0

/res and /assets are read-only

The folders in **/res/** have a predefined structure, and you can't add subfolders. File names can only use a-z, 0-9, and underscore.
The app's **assets** folder is more like a file system where you can have subfolders. File names can be anything

https://www.google.com/search?q=android+subfolders+in+res%2Fraw&rlz=1C1CHBF_enCA950CA950&oq=android+subfolders+in+res%2Fraw&aqs=chrome..69i57j33i22i29i30.10404j1j4&sourceid=chrome&ie=UTF-8

Permission request

READ_EXTERNAL_STORAGE or WRITE_EXTERNAL_STORAGE permission requests are needed when accessing other apps' files.

1. Add the user-permission request in the **AndroidManifest.xml**

```
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>
```

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.example.fileaccess">

    <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE"/>

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="FileAccess"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.FileAccess">
        <activity android:name=".MainActivity">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>

</manifest>
```

2. Add the run-time user-permission request in the **onCreate** method
3. The onRequestPermissionResult callback is only needed if you need to check whether the user has granted the permission or not

```
private static final int MY_PERMISSION_REQUEST_STORAGE = 1;

@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);

    // request permission to access storage
    if (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.WRITE_EXTERNAL_STORAGE) != PackageManager.PERMISSION_GRANTED) {
        ActivityCompat.requestPermissions(MainActivity.this, new
String[]{Manifest.permission.WRITE_EXTERNAL_STORAGE}, MY_PERMISSION_REQUEST_STORAGE);
    }
}

@Override
public void onRequestPermissionsResult(int requestCode, @NonNull String[]
permissions, @NonNull int[] grantResults) {
    super.onRequestPermissionsResult(requestCode, permissions, grantResults);
    switch (requestCode) {
        case MY_PERMISSION_REQUEST_STORAGE: {
            if (grantResults.length > 0 && grantResults[0] ==
PackageManager.PERMISSION_GRANTED) {
                if (ContextCompat.checkSelfPermission(MainActivity.this,
Manifest.permission.WRITE_EXTERNAL_STORAGE) == PackageManager.PERMISSION_GRANTED) {
                    // do nothing
                }
            } else {
                Toast.makeText(this, "Permission not granted",
Toast.LENGTH_SHORT).show();
            }
        }
    }
}
```

Get file path and usage

4. Uncomment the line for the storage location path that you want in the code below

```
try {
    //File path = getFilesDir();
    //File path = getCacheDir();

    //File path = getExternalFilesDir(null);
    //File path = getExternalCacheDir();

    //File path =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
```

```
    //File path =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOCUMENTS);

    //File paths[] = getExternalFilesDirs(null);
    File paths[] = getExternalCacheDirs();
    File path;
    if (paths.length > 1) {
        path = paths[1];
    } else {
        path = paths[0];
    }

    String filename = "Testfile.txt";
    String filecontent = "Hello world " + path;
    writeFile(path, filename, filecontent);
    String s = readFile(path, filename);
    textView.setText(s);
} catch (IOException e) {
    e.printStackTrace();
}
```

Write file

Method 1

```
private void writeFile(File path, String filename, String fileContents) throws
IOException {
    File file = new File(path, filename);
    OutputStream outputStream = new FileOutputStream(file);
    try {
        outputStream.write(fileContents.getBytes());
    } finally {
        outputStream.flush();
        outputStream.close();
    }
}
```

Method 2

```
private void writeFileInternalStorage(String filename, String fileContents) throws
IOException {
    FileOutputStream fos;
    try {
        fos = openFileOutput(filename, Context.MODE_PRIVATE);
        fos.write(fileContents.getBytes());
        fos.flush();
        fos.close();
    } catch (FileNotFoundException e) {
        e.printStackTrace();
    }
}
```

Method 3. Writing a string array to file

```
private void writeFile(File path, String filename, String[] allItems) {
    try {
        File file = new File (path, filename);
        FileWriter fileWriter = new FileWriter(file);
        PrintWriter writer = new PrintWriter(fileWriter);
        for (String allItem : allItems) {
            writer.println(allItem);
        }
        writer.flush();
        writer.close();
    } catch (Exception e) {
        e.printStackTrace();
    }
}
```

5. Usage.

```
try {
    writeFileInternalStorage("myfile.txt", "Hello world internal storage");
} catch (IOException e) {
    e.printStackTrace();
}
```

Write a bitmap picture file

```
private void writeFile(File path, String filename, Bitmap bitmap) {
    File file = new File(path, filename);
    ByteArrayOutputStream byteArrayOutputStream = new ByteArrayOutputStream();
    // the second parameter is the compression quality: 0=worse to 100=best
    bitmap.compress(Bitmap.CompressFormat.JPEG, 0, byteArrayOutputStream);
    try {
        FileOutputStream fileOutputStream = new FileOutputStream(file);
        fileOutputStream.write(byteArrayOutputStream.toByteArray());
        fileOutputStream.flush();
        fileOutputStream.close();
    } catch (IOException e) {
        e.printStackTrace();
    }
}
```

Read file

Method 1

```
private String readFile(File path, String filename) throws IOException {
    String s = "";
    try {
        File file = new File(path, filename);
        InputStream inputStream = new FileInputStream(file);
```

```
        InputStreamReader inputStreamReader = new InputStreamReader(inputStream);
        int size = inputStream.available();
        if (size > 0) {
            char[] buffer = new char[size];
            inputStreamReader.read(buffer);
            inputStream.close();
            //s = buffer.toString();    // wrong
            s = new String(buffer);
        } else {
            s = "File is empty "+filename;
        }
    } catch (Exception e) {
        e.printStackTrace();
        s = "File not found "+filename;
    }
    return s;
}
```

Method 2

```
private String readFile2(File path, String filename) throws IOException {
    String s = "";
    try {
        File file = new File(path, filename);
        InputStream inputStream = new FileInputStream(file);
        BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream));
        StringBuffer stringBuffer = new StringBuffer();
        String line = bufferedReader.readLine();
        while (line != null) {
            stringBuffer.append(line);
            line = bufferedReader.readLine();
        }
        if (stringBuffer.length() > 0) {
            s = stringBuffer.toString();
        } else {
            s = "File is empty " + filename;
        }
    } catch (Exception e) {
        e.printStackTrace();
        s = "File not found "+filename;
    }
    return s;
}
```

Method 3. Reading a file into a string array

```
private String[] readFile(File path, String filename) {
    try {
        File file = new File (path, filename);
        FileReader fileReader = new FileReader(file);
        BufferedReader reader = new BufferedReader(fileReader);
        ArrayList<String> tempList = new ArrayList<>();
        String line = reader.readLine();
```

```
        while (line != null) {
            tempList.add(line);
            line = reader.readLine();
        }
        reader.close();
        // convert from ArrayList<String> to String[]
        Object[] objArr = tempList.toArray();    // Convert ArrayList to object array
        String[] allStrings = Arrays.copyOf(objArr, objArr.length, String[].class);
// convert object array to String array
        return allStrings;
    } catch (IOException e) {
        e.printStackTrace();
        return null;
    }
}
```

Read and load a bitmap picture file

```
// this version is when given the URI to the picture file
private void loadPicture(Uri uri) {
    ImageView imageView = findViewById(R.id.imageView);
    imageView.setImageURI(uri);
}



// this version is when given the file path and name
// the picture file is first loaded into a bitmap variable
// returns the bitmap file of the picture read
private Bitmap loadPicture(File path, String filename) {
    File file = new File(path, filename);
    Bitmap bitmap = null;
    if (file.exists()) {
        ImageView imageView = findViewById(R.id.imageView);
        bitmap = BitmapFactory.decodeFile(String.valueOf(file));
        imageView.setImageBitmap(bitmap);
    } else {
        Log.i("myTag", "File not found "+file);
    }
    return bitmap;
}
```

File or Directory exists

6. To check if a file or a directory exists or not.

```
private boolean fileExists(File path, String filename) {
    File file = new File(path, filename);
    if (file.exists()) {
        Toast.makeText(this, "File exists: "+pathExternal+"/"+filename,
Toast.LENGTH_SHORT).show();
```

```
        return true;
    } else {
        Toast.makeText(this, "File not exists: "+pathExternal+"/"+filename,
Toast.LENGTH_SHORT).show();
        return false;
    }
}
```

File delete

```
private boolean deleteFile(File path, String filename) {
    File file = new File(path, filename);
    return file.delete();
}
```

File copy

Method 1

This version takes a File fromPath, File toPath, and String filename.

```
private void copyFile(File fromPath, File toPath, String filename) throws IOException
{
    try {
        File fromFile = new File(fromPath, filename);
        File toFile = new File(toPath, filename);
        InputStream inputStream = new FileInputStream(fromFile);
        OutputStream outputStream = new FileOutputStream(toFile);
        byte[] buffer = new byte[1024];
        int read;
        while ((read = inputStream.read(buffer)) != -1) {
            outputStream.write(buffer, 0, read);
        }
        inputStream.close();
        outputStream.close();
    } catch (IOException e) {
        e.printStackTrace();
        Log.i("myTag", "copyFile error "+e);
    }
}
```

Method 2

This version takes a Uri fromUri, and File toPath. The filename is extracted from the fromUri.

```
private void copyFile(Uri fromUri, File toPath) throws IOException {
    try {
        InputStream inputStream = getContentResolver().openInputStream(fromUri);
        //String filename = new File(fromUri.getPath()).getName();  // this method
might get the file ID instead
        // get the filename
        Cursor cursor = getContentResolver().query(fromUri, null, null, null, null);
```

```
        int nameIndex = cursor.getColumnIndex(OpenableColumns.DISPLAY_NAME);
        cursor.moveToFirst();
        String filename = cursor.getString(nameIndex);
        File toFile = new File(toPath, filename);
        OutputStream outputStream = new FileOutputStream(toFile);
        byte[] buffer = new byte[1024];
        int read;
        while ((read = inputStream.read(buffer)) != -1) {
            outputStream.write(buffer, 0, read);
        }
        inputStream.close();
        outputStream.close();
        Log.i("myTag", "File copied to "+toFile);
    } catch (Exception e) {
        e.printStackTrace();
        Log.i("myTag", "File copy error "+e);
    }
}
```

Method 3

The source and destination contain the full path including the filename

```
private void copyFile(File source, File destination) throws IOException {
    FileChannel in = new FileInputStream(source).getChannel();
    FileChannel out = new FileOutputStream(destination).getChannel();
    try {
        in.transferTo(0, in.size(), out);
    } catch (Exception e) {
        // post to log
    } finally {
        in.close();
        out.close();
    }
}
```

7. Example to copy a file from the DOWNLOAD directory to internal private files directory.

```
File fromPath =
Environment.getExternalStoragePublicDirectory(Environment.DIRECTORY_DOWNLOADS);
File toPath = getFilesDir();
String filename = "Testfile.txt";
try {
    copyFile(fromPath, toPath, filename);
} catch (IOException e) {
    e.printStackTrace();
}
```

Rename file

https://developer.android.com/reference/java/io/File#renameTo(java.io.File)

```
private Boolean renameFile(File path, String fromName, String toName) {
    File fromFile = new File(path, fromName);
    File toFile = new File(path, toName);
    return fromFile.renameTo(toFile);
}
```

Call

```
renameFile(getFilesDir(), "msf:1930", "image.png");
```

List files in directory

This will list all the files under the given directory path.
https://developer.android.com/reference/java/io/File#listFiles()

```
private void listFiles(File path) {
    File[] files = path.listFiles();
    for (int i=0; i<files.length; i++) {
        Log.i("myTag", "files="+files[i]);
    }
}
```

Call

```
listFiles(getFilesDir());
```

Create directory

https://developer.android.com/reference/java/io/File#mkdir()
https://developer.android.com/reference/java/io/File#mkdirs()

```
// directory can be a single directory name or a path
// Usage: createDirectory("MyData");
//        createDirectory("MyData/pictures");
private boolean createDirectory(File path String directory) {
    File file = new File(path, directory);
    return file.mkdirs();
}
```

8. To use a path with a subdirectory

```
File path = getFilesDir();
path = new File(path + "/subdirectory");
```

Copy Directory

```
// copy all files from one directory to another directory
// sub-directories inside are not copied
private void copyDirectory(File source, File destination) throws IOException {
    if (!destination.exists()) {
        destination.mkdir();
    }
```

```
    if (source.isDirectory()) {
        String [] children = source.list();
        for (int i=0; i < source.listFiles().length; i++) {
            copyFile(new File(source, children[i]), new File(destination,
children[i]));
        }
    } else {
        copyFile(source, destination);
    }
}
```

```
// recursively copy all files including sub-directories from one directory to another
directory
private void copyDirectoryAll(File source, File destination) throws IOException {
    if (source.isDirectory()) {
        if (!destination.exists()) {
            destination.mkdir();
        }
        String [] children = source.list();
        for (int i=0; i < source.listFiles().length; i++) {
            copyDirectoryAll(new File(source, children[i]), new File(destination,
children[i]));    // recursive call
        }
    } else {
        FileChannel inChannel = new FileInputStream(source).getChannel();
        FileChannel outChannel = new FileInputStream(destination).getChannel();
        try {
            inChannel.transferTo(0, inChannel.size(), outChannel);
        } finally {
            if (inChannel != null) inChannel.close();
            if (outChannel != null) outChannel.close();
        }
    }
}
```

Delete Directory Recursively

```
void deleteRecursive(File fileOrDirectory) {
    if (fileOrDirectory.isDirectory())
        for (File child : fileOrDirectory.listFiles())
            deleteRecursive(child);

    fileOrDirectory.delete();
}
```

Intents

Open file picker using intent for a text file

- https://developer.android.com/training/data-storage/shared/documents-files
- https://developer.android.com/guide/components/intents-common#Storage

```
private static final int REQUEST_CODE = 2;
private void filePicker() {
    Intent intent = new Intent();
    //intent.setType("image/*");
    intent.setType("text/*");
    intent.setAction(Intent.ACTION_GET_CONTENT);
    // Optionally, specify a URI for the file that should appear in the
    // system file picker when it loads.
    //intent.putExtra(DocumentsContract.EXTRA_INITIAL_URI, pickerInitialUri);
    //startActivityForResult(Intent.createChooser(intent, "Select File"),
REQUEST_CODE);
    startActivityForResult(intent, REQUEST_CODE);
}

// onActivityResult is called after the user has selected a file from the file picker
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent
data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_CODE && resultCode == Activity.RESULT_OK) {
        // The intent data contains a URI for the document or directory that the user
selected.
        Uri uri = data.getData();
        //String path = data.getData().getPath();
        //Log.i("myTag", "uri=" + uri);
        //Log.i("myTag", "path=" + path);
        // Perform operations on the document using its URI.
        try {
            String s = readFileUri(uri);
            textView.setText(s);
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}

private String readFileUri(Uri uri) throws IOException {
    try {
        InputStream inputStream = getContentResolver().openInputStream(uri);
        BufferedReader bufferedReader = new BufferedReader(new
InputStreamReader(inputStream));
        StringBuffer stringBuffer = new StringBuffer();
        String line = bufferedReader.readLine();
        while (line != null) {
            stringBuffer.append(line);
            line = bufferedReader.readLine();
        }
        if (stringBuffer.length() > 0) {
            return stringBuffer.toString();
        } else {
            return "File is empty ";
        }
    } catch (Exception e) {
        return "File not found ";
```

```
    }
}
```

Open file picker using intent for an image file

Document: https://developer.android.com/guide/components/intents-common#ImageCapture
Example: https://developer.android.com/training/camera/photobasics

```
private static final int REQUEST_CODE = 2;
private void filePicker() {
    Intent intent = new Intent();
    intent.setType("image/*");
    //intent.setType("text/*");
    intent.setAction(Intent.ACTION_GET_CONTENT);
    // Optionally, specify a URI for the file that should appear in the
    // system file picker when it loads.
    //intent.putExtra(DocumentsContract.EXTRA_INITIAL_URI, pickerInitialUri);
    //startActivityForResult(Intent.createChooser(intent, "Select File"),
REQUEST_CODE);
    startActivityForResult(intent, REQUEST_CODE);
}

@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent
data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_CODE && resultCode == Activity.RESULT_OK) {
        // The intent data contains a URI for the document or directory that the user
selected.
        Uri uri = data.getData();
        String path = data.getData().getPath();
        //Log.i("myTag", "uri=" + uri);
        //Log.i("myTag", "path=" + path);
        // Perform operations on the document using its URI.
        //String s = readFileUri(uri);
        textView.setText(uri.toString());
        loadPicture(uri);
    }
}

private void loadPicture(Uri uri) {
    ImageView imageView = findViewById(R.id.imageView);
    imageView.setImageURI(uri);
}


private void loadPicture(File path, String filename) {
    File file = new File(path, filename);
    if (file.exists()) {
        imageView = findViewById(R.id.imageView);
        Bitmap bitmap = BitmapFactory.decodeFile(String.valueOf(file));
        imageView.setImageBitmap(bitmap);
    } else {
        Log.i("myTag", "File not found "+file);
```

```
    }
}
```

Selecting multiple files

Allow the user to select and return multiple items
Intent.EXTRA_ALLOW_MULTIPLE
Document:
https://developer.android.com/reference/android/content/Intent#EXTRA_ALLOW_MULTIPLE
Example: https://newbedev.com/select-multiple-images-from-android-gallery

```
private static final int FILE_REQUEST_CODE = 2;
private void FilePicker() {
    Intent intent = new Intent();
    intent.setType("image/*");
    intent.setAction(Intent.ACTION_GET_CONTENT);
    intent.putExtra(Intent.EXTRA_ALLOW_MULTIPLE, true);
    startActivityForResult(intent, FILE_REQUEST_CODE);
}
```

```
@Override
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent
data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == FILE_REQUEST_CODE && resultCode == RESULT_OK) {
        File toPath = getFilesDir();
        Uri uri;
        // check if single picture or multiple pictures selected
        // https://newbedev.com/select-multiple-images-from-android-gallery
        if (data.getClipData() != null) {
            // multiple pictures selected
            int count = data.getClipData().getItemCount();
            for(int i = 0; i < count; i++) {
                uri = data.getClipData().getItemAt(i).getUri();
                Log.i("myTag", "copy multiple file "+uri);
                try {
                    copyFile(uri, toPath);
                } catch (IOException e) {
                    e.printStackTrace();
                }
            }
        } else if (data.getData() != null) {
            // single picture selected
            uri = data.getData();
            try {
                copyFile(uri, toPath);
            } catch (IOException e) {
                e.printStackTrace();
            }
        }
    }
}
```

URI parts

https://developer.android.com/training/secure-file-sharing/retrieve-info#RetrieveFileInfo

9. To convert from a file path to URI

```
String filename = "image.png";
File directory = this.getFilesDir(); //storage path /data/user/0/<project name>/files
File filepath = new File(directory, filename);
Uri uri = Uri.fromFile(filepath);
```

Note: if uri is null then see
https://stackoverflow.com/questions/13080540/what-causes-androids-contentresolver-query-to-return-null

10. To extract a file's MIME type, filename and size given its URI. Using this method the filename is always the correct original filename including the type.

```
private void getFileInfo(Uri uri) {
    String mimeType = getContentResolver().getType(uri);
    Log.i("myTag", "mimeType = " + mimeType);

    Cursor cursor = getContentResolver().query(uri, null, null, null, null);
    /*
     * Get the column indexes of the data in the Cursor,
     * move to the first row in the Cursor, get the data,
     * and display it.
     */
    int nameIndex = cursor.getColumnIndex(OpenableColumns.DISPLAY_NAME);
    int sizeIndex = cursor.getColumnIndex(OpenableColumns.SIZE);
    cursor.moveToFirst();
    Log.i("myTag", "File name = " + cursor.getString(nameIndex));
    Log.i("myTag", "File size = " + cursor.getLong(sizeIndex));
}
```

11. To extract a file's path and filename given its URI. Using this method the filename might not be the original filename, but instead is an arbitrary ID. See note below.

```
protected void onActivityResult(int requestCode, int resultCode, @Nullable Intent
data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == REQUEST_CODE && resultCode == Activity.RESULT_OK) {
        // The intent data contains a URI for the document or directory that the user
selected.
        Uri uri = data.getData();
        String pathRaw = data.getData().getPath();
        String path = pathRaw.split(":")[1];    // split string into two parts
delimited at :
        File filename = new File(new File(data.getData().getPath()).getName());
        Log.i("myTag", "uri = " + uri);
        Log.i("myTag", "pathRaw = " + pathRaw);
```

```
        Log.i("myTag", "path = " + path);
        Log.i("myTag", "filename = " + filename);
```

Assuming that a picture file named img.png in the Documents directory is selected in the file picker then the following is printed:

uri = content://com.android.externalstorage.documents/document/primary%3ADocuments%2Fimg.png

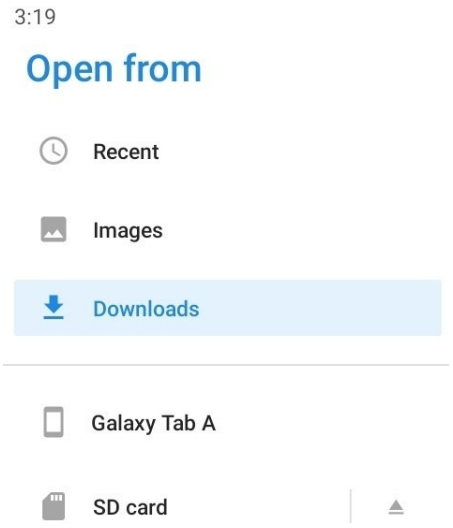
pathRaw = /document/primary:Documents/img.png

path = Documents/img.png

filename = img.png

Note: the filename might be something like msf:1930 instead of the actual original filename. See below.

- In the file picker if selecting a file from the Recent, Images, or Downloads folders then the filename extracted from the URI is (might be) an ID number such as 82 or msf:1930. However, if selecting a file from the device folder (e.g. Galaxy Tab A) or the SD card then the filename extracted from the URI is the original filename such as image.jpg. See file picker screenshot below



??????

```
private String getRealPathFromURI(Uri contentUri) {
    String[] proj = { MediaStore.Video.Media.DATA };
    Cursor cursor = managedQuery(contentUri, proj, null, null, null);
    int column_index = cursor.getColumnIndexOrThrow(MediaStore.Images.Media.DATA);
    cursor.moveToFirst();
    Log.i("myTag","realPathFromURI="+cursor.getString(column_index));
    return cursor.getString(column_index);
}
```

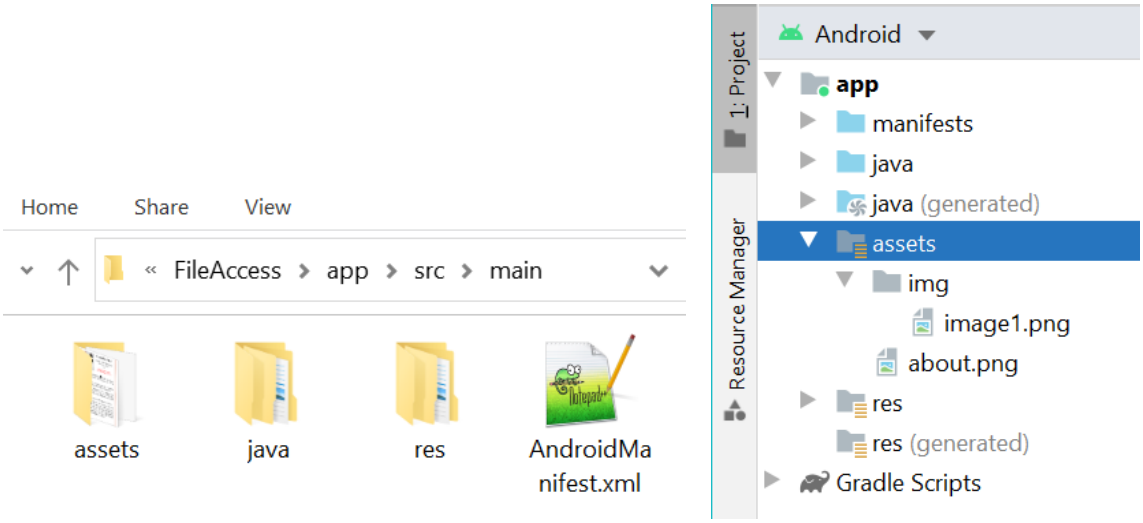
??????

```
//        imageView.setImageDrawable(Drawable.createFromPath(pathName));
        //imageView.setImageResource(imgResId);

//        Uri uri = Uri.parse(data.getData().getPath());
//        Log.i("myTag", "URI="+uri);

//        imageView.setImageURI(uri);
//        imageView.setImageURI(Uri.fromFile(new File("/sdcard/cats.jpg")));

        // the following works
//        final Integer imgResId = R.drawable.about;
//        Log.i("myTag", "imgResId="+imgResId);
//        imageView.setImageResource(imgResId);



        File file = new File(method3().toString() + "/Camera/Picture.jpg");
        if (file.exists()) {
            Log.i("myTag", "method1 file exists path="+file);
//            imageView.setImageDrawable(Drawable.createFromPath(file.toString()));

//            Bitmap bitmap = BitmapFactory.decodeFile(file.toString());
//            imageView.setImageBitmap(bitmap);
            Uri uri = Uri.parse(file.toString());
//            imageView.setImageURI(uri);
//            imageView.setImageURI(Uri.fromFile(new File("/sdcard/cats.jpg")));
```

AssetManager

https://developer.android.com/reference/android/content/res/AssetManager
https://www.concretepage.com/android/android-assetmanager-example-to-load-image-from-assets-folder

12. The AssetManager can be used to access files in the **assets** folder in your project.
    - Using Windows File Manager, create a folder named **assets** in **<project>/app/src/main**. You cannot create this folder from Android Studio. This folder will show up in your project



13. Copy your app's resources files into this folder. In the above figure, I have two png files, with one inside a subfolder.

14. To open the files in ACCESS_STREAMING mode use

    InputStream in = assetManager.open(about.png);

    or

    InputStream in = assetManager.open(img/image1.png).

15. To load and display the image

```
private void getAsset(String filename) throws IOException {
    AssetManager assetManager = getAssets();
    try {
        InputStream in = assetManager.open(filename);
        Bitmap bitmap = BitmapFactory.decodeStream(in);
        imageView = findViewById(R.id.imageView);
        imageView.setImageBitmap(bitmap);
    } catch (IOException e) {
        e.printStackTrace();
        Log.i("myTag", "File not found "+filename);
    }
}
```

16. To list all the files in the **assets** folder (including subdirectories)

```
private void listAllAssets() throws IOException {
    AssetManager assetManager = getAssets();
    String[] paths = assetManager.list(""); // specify subdirectory path. "" = root
    for (int i=0; i<paths.length; i++) {
        Log.i("myTag", "path["+i+"]="+paths[i]);
    }
}
```

raw folder

17. The raw folder path is in **/res/raw**

18. How to get files in the raw folder

```
imageView.setImageResource(context.getResources().getIdentifier(fileName, "raw",
context.getPackageName()));    // file from app's /res/raw folder
```

```
mediaPlayer = MediaPlayer.create(MainActivity.this,
MainActivity.this.getResources().getIdentifier(fileName, "raw",
MainActivity.this.getPackageName()));   // get file in the /res/raw folder
```

drawable folder

19. The drawable folder path is in **/res/drawable**

20. How to get files in the drawable folder

```
imageView.setImageResource(context.getResources().getIdentifier(fileName, "drawable",
context.getPackageName()));    // file from app's /res/raw folder
```

```
mediaPlayer = MediaPlayer.create(MainActivity.this,
MainActivity.this.getResources().getIdentifier(fileName, "drawable",
MainActivity.this.getPackageName()));   // get file in the /res/raw folder
```