15

Android Persistency: Files

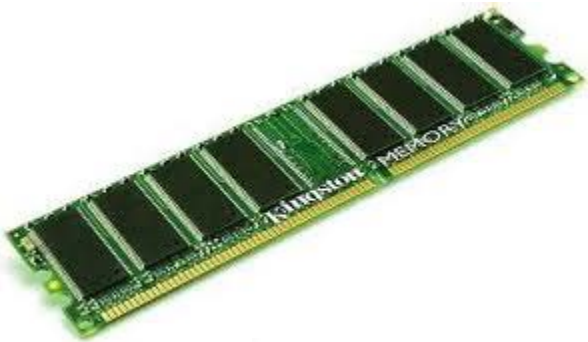
Victor Matos
Cleveland State University

Notes are based on:

The Busy Coder's Guide to Android Development
by Mark L. Murphy

ISBN: 978-0-9816780-0-9
&
Android Developers
http://developer.android.com/index.html

Android Files



Android uses the same *file* constructions found in a typical Java application.

Files can be stored in the device's (small) main memory or in the much larger SD card. They can also be obtained from the network (as we will see later).

Files stored in the device's memory, stay together with other application's resources (such as icons, pictures, music, …).
We will call this type: *Resource Files.*

Android Files

Your data storage options are the following:

1. **Shared Preferences** Store private primitive data in key-value pairs.
2. **Internal Storage** Store private data on the device's memory.
3. **External Storage** Store public data on the shared external storage.
4. **SQLite Databases** Store structured data in a private/public database.
5. **Network Connection** Store data on the web with your own network server.

Android Files

Shared Preferences. Good for a few items saved as <Name, Value>

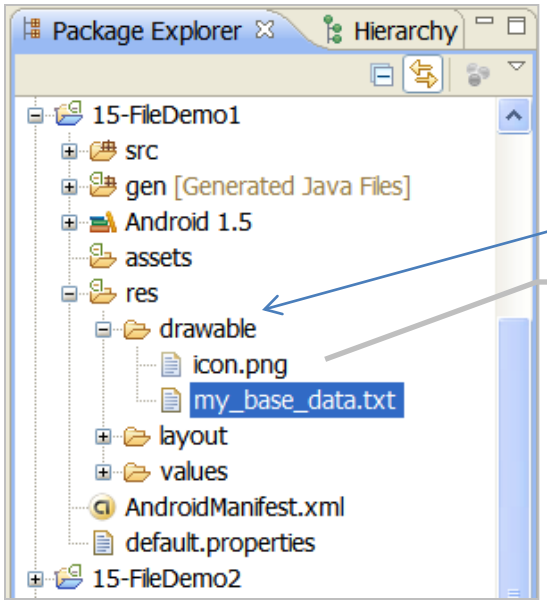
```
private void usingPreferences(){
  // Save data in a SharedPreferences container
  // We need an Editor object to make preference changes.

  SharedPreferences settings = getSharedPreferences("my_preferred_Choices",
                                                     Context.MODE_PRIVATE);
  SharedPreferences.Editor editor = settings.edit();
          editor.putString("favorite_color", "#ff0000ff");
          editor.putInt("favorite_number", 101);
  editor.commit();

  // retrieving data from SharedPreferences container
  String favColor = settings.getString("favorite_color", "default black");
  int favNumber = settings.getInt("favorite_number", 0);

  Toast.makeText(this, favColor + " " + favNumber, 1).show();

}
```

Android Files

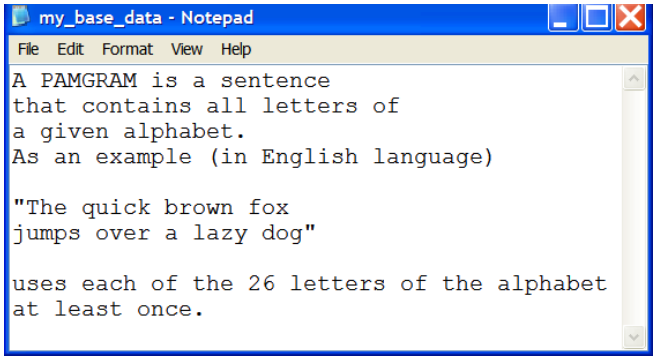
Internal Storage. Using Android Resource Files

When an application's **.apk** bytecode is deployed it may store in memory: *code*, *drawables*, and other *raw* resources (such as files). Acquiring those resources could be done using a statement such as:

```
InputStream is = this.getResources()
                    .openRawResource(R.drawable.my_base_data);
```



Use drag/drop to place file **my_base_data.txt** in **res** folder. It will be stored in the device's memory as part of the .apk

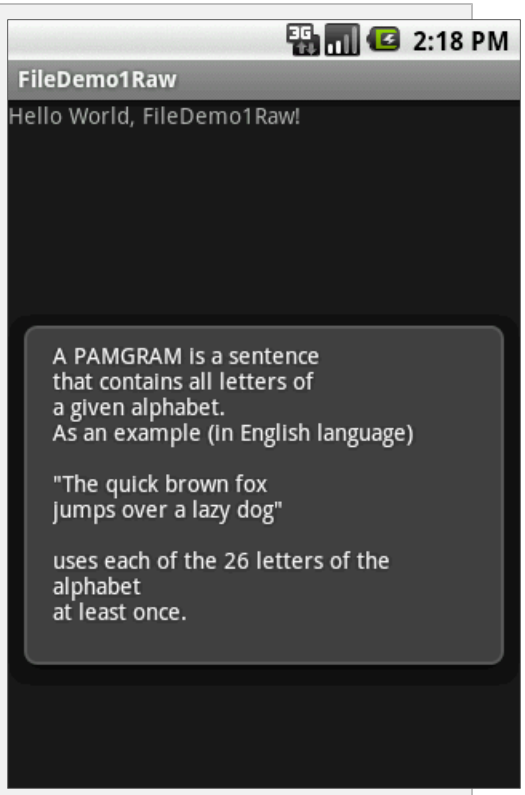


Spanish Pamgram
La cigüeña tocaba cada vez mejor el saxofón y el búho pedía kiwi y queso.

Android Files

Example 0: Reading a Resource File (see previous figure)

```
//reading an embedded RAW data file
package cis493.files;
import . . .
import java.io.*;


public class FileDemo1Raw extends Activity {

@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);

    try {
        PlayWithRawFiles();
    } catch (IOException e) {
        Toast.makeText(getApplicationContext(),
                       "Problems: " + e.getMessage(), 1).show();
    }
}// onCreate
```



Android Files

Example 1: Reading a Resource File (see previous figure)

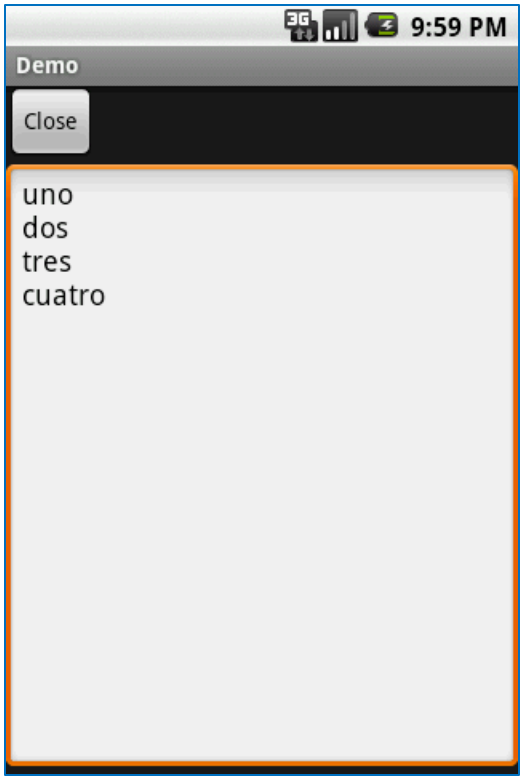
```
public void PlayWithRawFiles() throws IOException {
        String str="";
        StringBuffer buf = new StringBuffer();

        InputStream is = this.getResources()
                        .openRawResource(R.drawable.my_base_data);

        BufferedReader reader = new BufferedReader(
                                    new InputStreamReader(is));
        if (is!=null) {
            while ((str = reader.readLine()) != null) {
                buf.append(str + "\n" );
            }
        }
        is.close();
        Toast.makeText(getBaseContext(),
                        buf.toString(), Toast.LENGTH_LONG).show();
}// PlayWithRawFiles

} // FilesDemo1
```

Android Files

Example2: **(Internal Storage)** Read/Write an Internal File.

In this example an application collects data from the UI and saves it to a persistent data file into the (limited) internal Android System Space area.

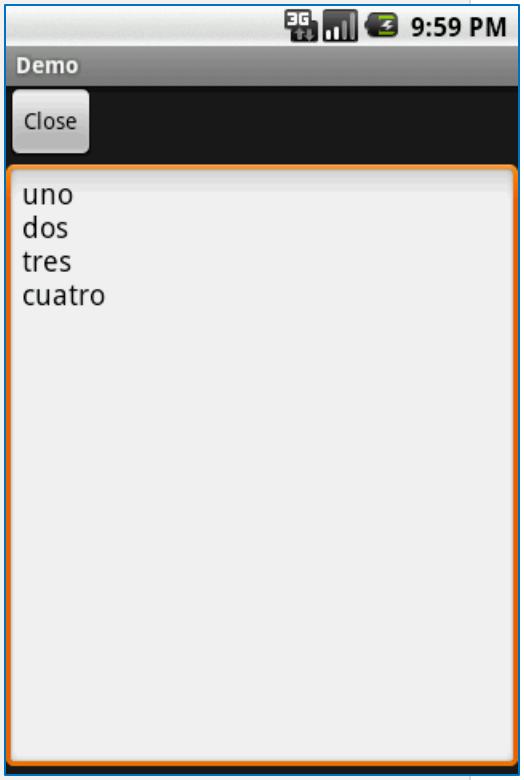
Next time the application is executed the Resource file is read and its data is shown in the UI



Android Files

Example2: Grab from screen, save to file, retrieve from file.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:orientation="vertical">

    <Button android:id="@+id/close"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="Close" />

    <EditText
        android:id="@+id/editor"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        android:singleLine="false"
        android:gravity="top"
        />

</LinearLayout>
```



Android Files

Example2: Grab from screen, save to file, retrieve from file.

```
package cis493.demo;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import java.io.BufferedReader;
import java.io.InputStream;
import java.io.InputStreamReader;
import java.io.OutputStreamWriter;

public class Demo extends Activity {

     private final static String NOTES="notes.txt";
     private EditText txtBox;
```

Android Files

Example2: Grab from screen, save to file, retrieve from file.

```
@Override
public void onCreate(Bundle icicle) {
     super.onCreate(icicle);
     setContentView(R.layout.main);

     txtBox=(EditText)findViewById(R.id.editor);

     Button btn=(Button)findViewById(R.id.close);

     btn.setOnClickListener(new Button.OnClickListener() {
          public void onClick(View v) {
            finish();
          }
     });

}//onCreate
```

Android Files

Example2: Grab from screen, save to file, retrieve from file.

```
public void onResume() {
     super.onResume();
     try {
          InputStream in = openFileInput(NOTES);
          if (in!=null) {
               BufferedReader reader = new BufferedReader(new InputStreamReader(in));

               String str = "";
               StringBuffer buf=new StringBuffer();

               while ((str = reader.readLine()) != null) {
                    buf.append(str+"\n");
               }
               in.close();
               editor.setText(buf.toString());
          }//if
     }
     catch (java.io.FileNotFoundException e) {
          // that's OK, we probably haven't created it yet
     }
     catch (Throwable t) {
          Toast.makeText(this, "Exception: "+ t.toString(), 2000).show();
     }
}
```

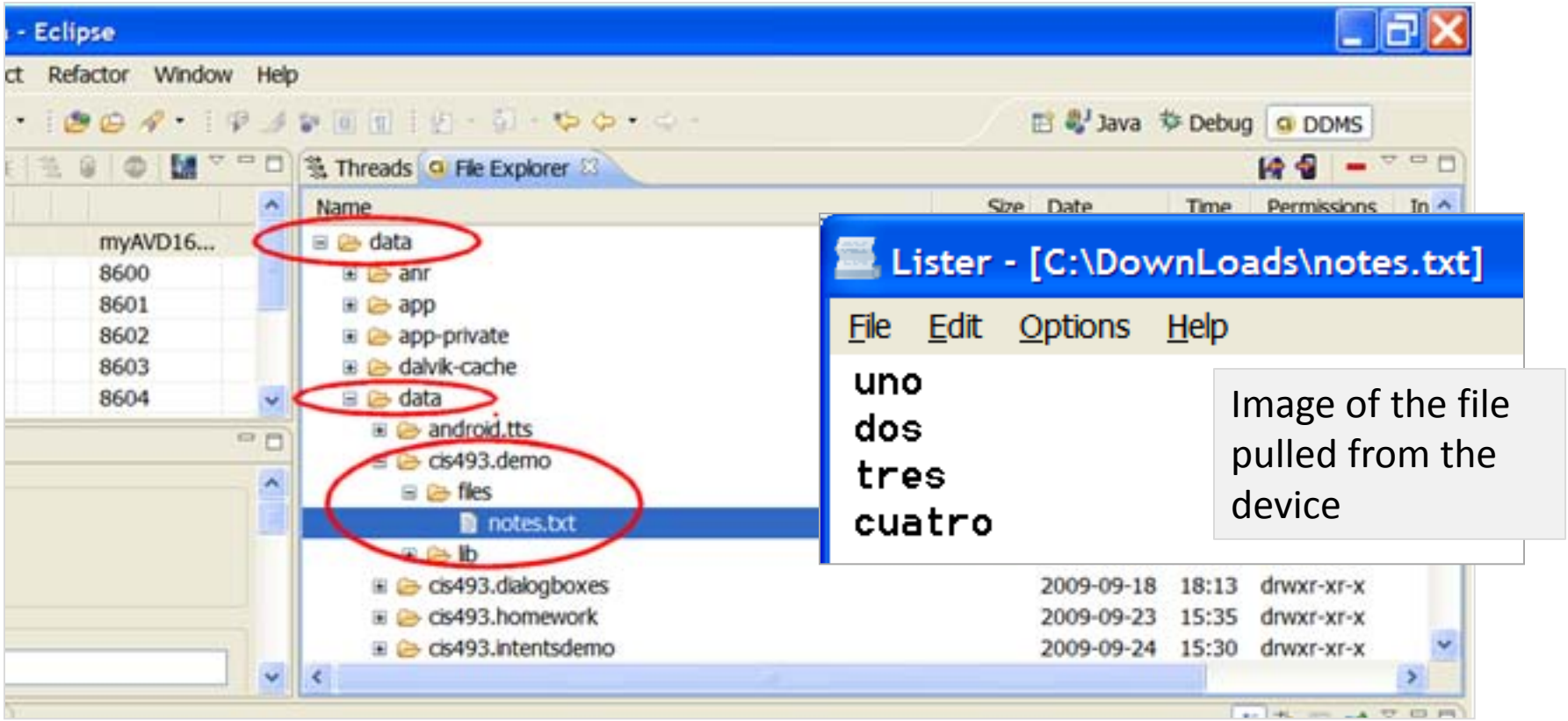
Android Files

Example2: Grab from screen, save to file, retrieve from file.

```
 public void onPause() {
     super.onPause();

     try {
          OutputStreamWriter out=
                                 new OutputStreamWriter(openFileOutput(NOTES, 0));

          // get your data from the screen's text box
          out.write(txtBox.getText().toString());
          out.close();
     }
     catch (Throwable t) {
          Toast.makeText(this, "Exception: "+ t.toString(), 2000).show();
     }
 }


}//class
```

Android Files

File is stored in the phone's memory under: /data/data/app/files

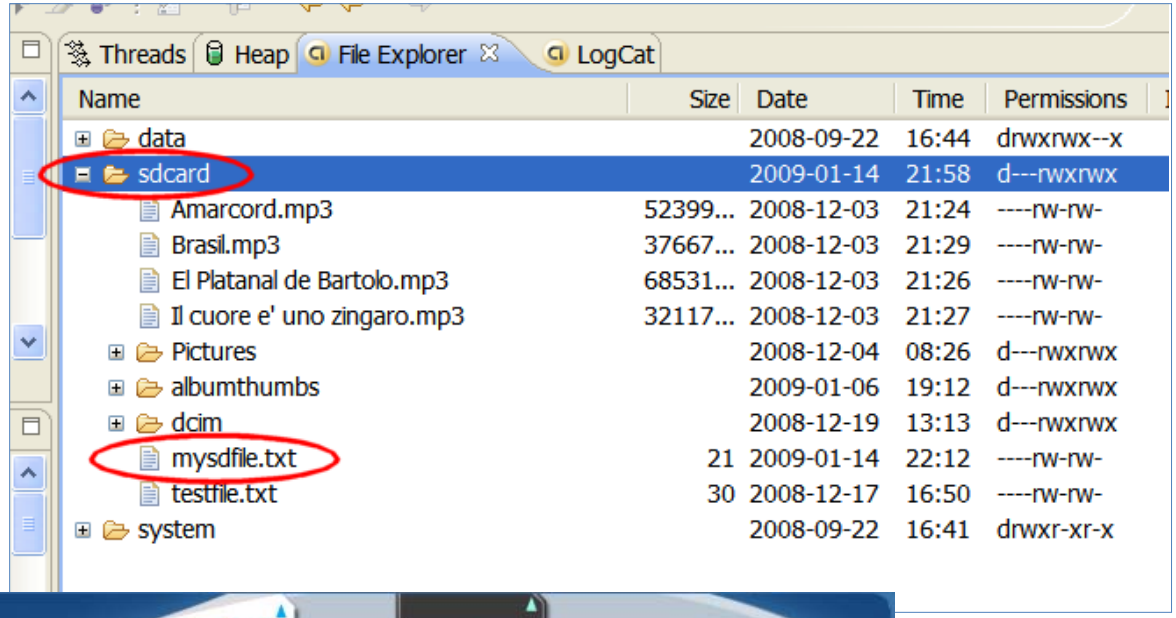


Android Files

Example 3: (External Storage)

Reading/Writing to the External Device's **SD card**.

Storing data into the SD card has the obvious advantage of a larger working space.



Android Files

WARNING: Writing to the Device's **SD card**.

Since SDK1.6 it is necessary to request permission to write to the SD card.
Add the following clause to your AndroidManifest.xml

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

Android Files

Example 3: Reading/Writing to the Device's SD card.

Assume the SD card in this example has been named *sdcard*. We will use the Java.io.File class to designate the file's path. The following fragment illustrates the code strategy for output files.

```
File myFile = new File("/sdcard/mysdfile.txt");
myFile.createNewFile();

OutputStreamWriter myOutWriter = new OutputStreamWriter(
                                        new FileOutputStream(myFile) );

myOutWriter.append(txtData.getText());

myOutWriter.close();
fOut.close();
```

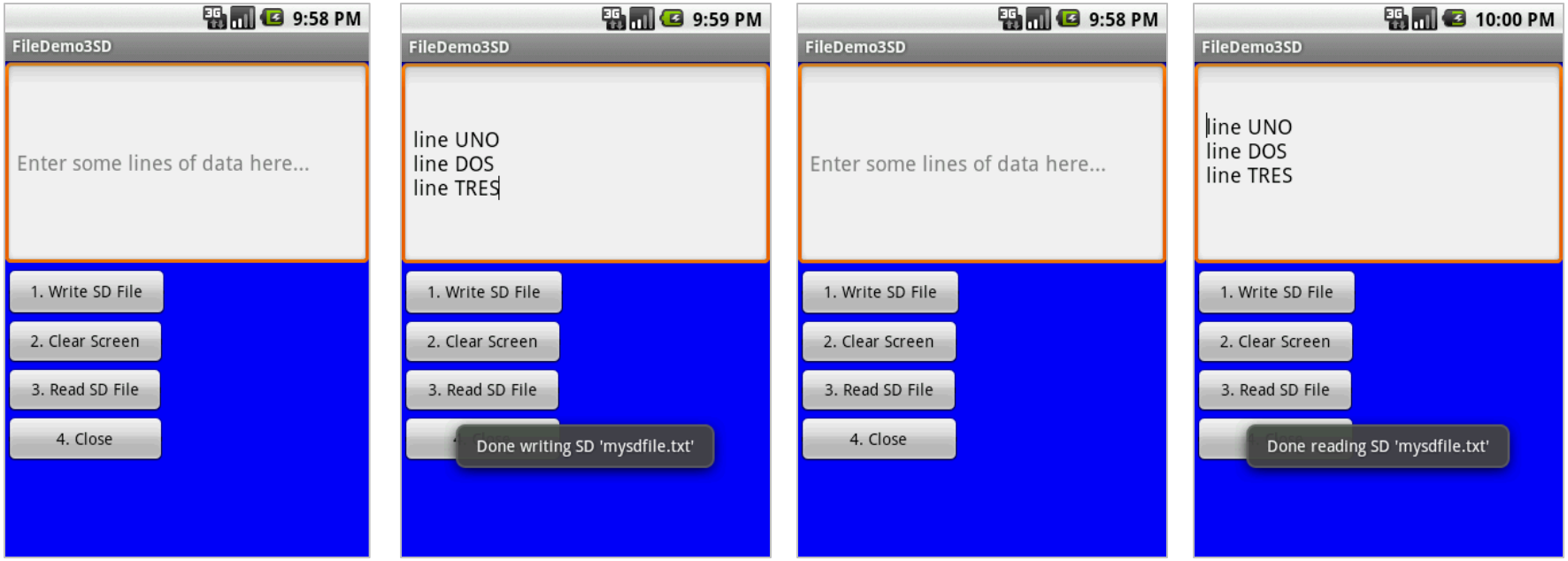
Android Files

Example 3: Reading/Writing to the Device's SD card.

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
xmlns:android=http://schemas.android.com/apk/res
/android
android:id="@+id/widget28"
android:layout_width="fill_parent"
android:layout_height="fill_parent"
android:background="#ff0000ff"
android:orientation="vertical"
>
<EditText
android:id="@+id/txtData"
android:layout_width="fill_parent"
android:layout_height="180px"
android:text="Enter some data here ..."
android:textSize="18sp"  />

<Button
android:id="@+id/btnWriteSDFile"
android:layout_width="143px"
android:layout_height="44px"
android:text="1. Write SD File"  />

<Button
android:id="@+id/btnClearScreen"
android:layout_width="141px"
android:layout_height="42px"
android:text="2. Clear Screen"  />

<Button
android:id="@+id/btnReadSDFile"
android:layout_width="140px"
android:layout_height="42px"
android:text="3. Read SD File"  />

<Button
android:id="@+id/btnClose"
android:layout_width="141px"
android:layout_height="43px"
android:text="4. Close" />

</LinearLayout>
```

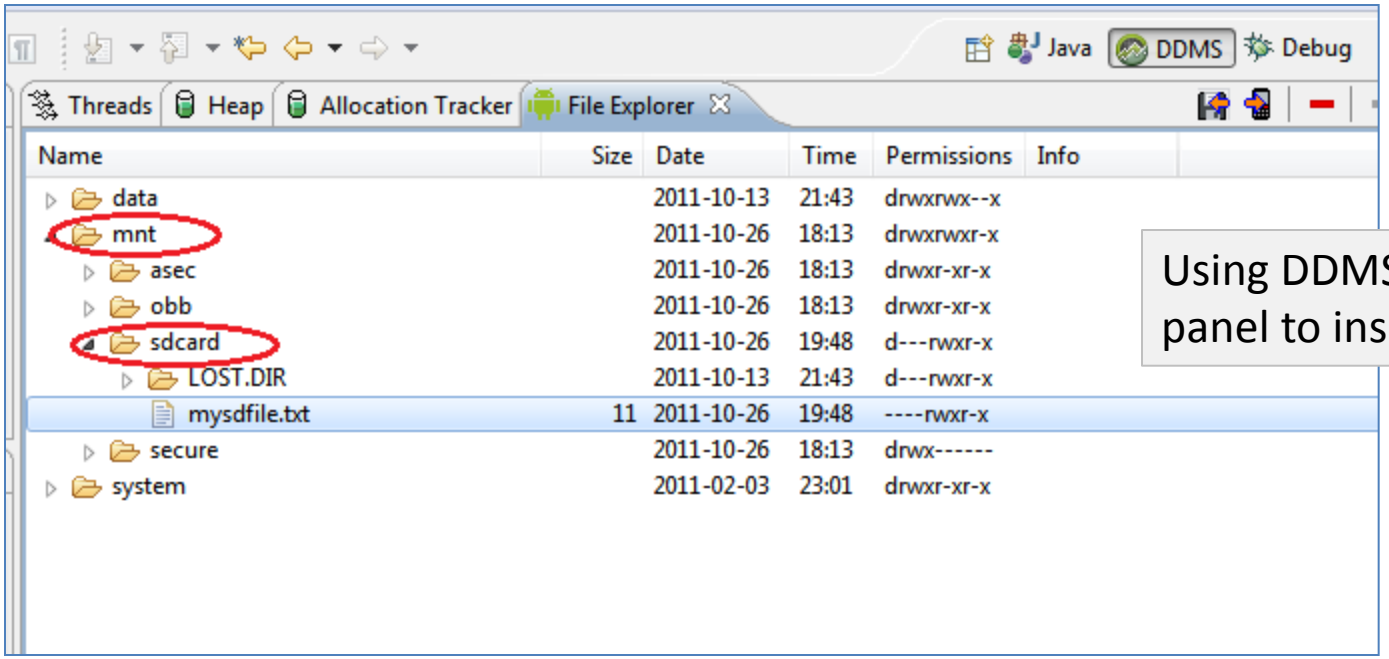
Android Files

Example 3: Reading/Writing to the Device's SD card.



Android Files

Example 3: Reading/Writing to the Device's SD card.



Using DDMS *File Explorer* panel to inspect the SD card.

Android Files

Example 3: Reading/Writing to the Device's SD card.

```
package cis493.filedemo;
import java.io.*;
import android.app.Activity;
import android.os.Bundle;
import android.view.*;
import android.view.View.OnClickListener;
import android.widget.*;

public class FileDemo3SD extends Activity {
// GUI controls
EditText txtData;
Button btnWriteSDFile;
Button btnReadSDFile;
Button btnClearScreen;
Button btnClose;
@Override
public void onCreate(Bundle savedInstanceState) {
     super.onCreate(savedInstanceState);
     setContentView(R.layout.main);
     // bind GUI elements with local controls
     txtData = (EditText) findViewById(R.id.txtData);
     txtData.setHint("Enter some lines of data here...");
```

Android Files

Example 3: Reading/Writing to the Device's SD card.

```
btnWriteSDFile = (Button) findViewById(R.id.btnWriteSDFile);
btnWriteSDFile.setOnClickListener(new OnClickListener() {
    @Override
    public void onClick(View v) {
    // write on SD card file data from the text box
    try {
        File myFile = new File("/sdcard/mysdfile.txt");
        myFile.createNewFile();
        FileOutputStream fOut = new FileOutputStream(myFile);
        OutputStreamWriter myOutWriter = new OutputStreamWriter(fOut);
        myOutWriter.append(txtData.getText());
        myOutWriter.close();
        fOut.close();
        Toast.makeText(getBaseContext(),
                "Done writing SD 'mysdfile.txt'",
        Toast.LENGTH_SHORT).show();
    } catch (Exception e) {
        } Toast.makeText(getBaseContext(),
                  e.getMessage(), Toast.LENGTH_SHORT).show();

    }// onClick
}); // btnWriteSDFile
```

Android Files

Example 3: Reading/Writing to the Device's SD card.

```
btnReadSDFile = (Button) findViewById(R.id.btnReadSDFile);
btnReadSDFile.setOnClickListener(new OnClickListener() {
 @Override
 public void onClick(View v) {
 // write on SD card file data from the text box
 try {
  File myFile = new File("/sdcard/mysdfile.txt");
  FileInputStream fIn = new FileInputStream(myFile);
  BufferedReader myReader = new BufferedReader(new InputStreamReader(fIn));
  String aDataRow = "";
  String aBuffer = "";
  while ((aDataRow = myReader.readLine()) != null) {
     aBuffer += aDataRow + "\n";
  }
  txtData.setText(aBuffer);
  myReader.close();
  Toast.makeText(getBaseContext(),
       "Done reading SD 'mysdfile.txt'", 1).show();
  } catch (Exception e) {
      Toast.makeText(getBaseContext(), e.getMessage(), 1).show();
  }
 }// onClick
}); // btnReadSDFile
```

Android Files

Example 3: Reading/Writing to the Device's SD card.

```
 btnClearScreen = (Button) findViewById(R.id.btnClearScreen);
 btnClearScreen.setOnClickListener(new OnClickListener() {
     @Override
     public void onClick(View v) {
          // clear text box
          txtData.setText("");
     }
 }); // btnClearScreen

 btnClose = (Button) findViewById(R.id.btnClose);
 btnClose.setOnClickListener(new OnClickListener() {
     @Override
     public void onClick(View v) {
          // clear text box
          finish();
     }
 }); // btnClose

 }// onCreate

}// class
```

Android Files

Example 3: Reading/Writing to the Device's SD card. You may also use the Scanner/PrintWriter classes, as suggested below:

```
private void testScannerFiles(){
// Add to manifest the following permission request
// <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
      try {
            String SDcardPath = Environment.getExternalStorageDirectory().getPath();
            String mySDFileName = SDcardPath + "/" + "mysdfiletest.txt";
            tvMessage.setText("Writing to: " + mySDFileName);
            PrintWriter outfile= new PrintWriter( new FileWriter(mySDFileName) );
                  outfile.println("Hola Android");
                  outfile.println("Adios Android");
                  outfile.println(new Date().toString());
            outfile.close();
            // read SD-file,show records.
            Scanner infile= new Scanner(new FileReader(mySDFileName));
            String inString= "\n\nReading from: " + mySDFileName + "\n";
            while(infile.hasNextLine()) {
                  inString += infile.nextLine() + "\n";
            }
            tvMessage.append(inString);
            infile.close();
      } catch (FileNotFoundException e) {
            tvMessage.setText( "Error: " + e.getMessage());
      } catch (IOException e) {
            tvMessage.setText( "Error: " + e.getMessage());
      }
}
```

Android Files

Example 4: Some more ideas on using the Scanner/PrintWriter classes.

```
// writing
FileOutputStream fos = openFileOutput("XYZ",
Context.MODE_PRIVATE);

PrintWriter outfile= new PrintWriter( fos );
outfile.println("Hola Android");
outfile.close();


// reading
 InputStream is = openFileInput("XYZ");
 Scanner infile= new Scanner(is);
 String inString= "";
 while(infile.hasNextLine()) {
    inString = infile.nextLine();
 }
```

Files

Questions ?

Files

Accessing the SD card

```
String spPath = “”;

sdPath = Environment.getExternalStorageDirectory()
            .getAbsolutePath()
            + “/myFileName.txt” ;
```