Programming the Android Platform

CMSC498G

Logistics

- Professor
  - Adam Porter
  - 4125 AVW
  - aporter@cs.umd.edu

- Course meets
  - W 3:00 – 3:50 in CSI 3118
Goals

- Learn more about
  - Mobile devices
  - Mobile device programming concepts
  - The Android platform
- Develop a complete 1-semester, undergraduate level course
  - Materials will be made open source (e.g., via Creative Commons License)
    - List of topics
    - Lecture materials
    - Assignments
    - Miscellaneous notes & links
- Develop interesting Android applications
  - Each student will do a semester project

Ground Rules

- This course will be largely student run
- Professor will lecture for first 3 weeks
- Students will be responsible for remaining lectures
  - Each student will prepare 1 lecture
- A lecture includes:
  - 45-minute presentation (with slides)
  - Laboratory exercise with Junit test cases & solutions
    - To be done by students outside of class
    - Should require 1-2 hours of programming effort
  - List of supplementary materials, if appropriate
- Submit lecture 2 weeks before presentation date
  - Presenter MUST be on time to lecture
Grading

- Student lectures (1/3)
- Final exam (1/3)
- Individual project (1/3)

The Android Platform

- A software stack for mobile devices. It includes:
  - Operating system
  - Middleware
  - Key applications
- Developers use the Android SDK to create Android applications that run on the Android Platform
- The SDK includes:
  - Libraries
  - Development tools
- Lots of documentation at http://developer.android.com
  - Start browsing today!
The Android Architecture


Linux Kernel

- Defines an abstraction layer between HW & SW
- Provides services such as:
  - Security
  - Memory & process management
  - Network stack
  - Device driver model
Linux Kernel (cont.)

• Some Android-specific components
  – Binder – IPC
  – Android shared memory (ashmem)
  – Power management – wake locks
  – Alarm driver (timers)
  – Low memory killer
  – Kernel debugger
  – Logger

Native Libraries

• C/C++ libraries accessed through the Android application framework
• Some examples:
  – System C library – bionic libe
  – Surface Manager – Display mgmt
  – Media Framework – Audio/video
  – Webkit- Web browser engine
  – OpenGL ES, SGL - Graphics engines
  – FreeType - Font rendering
  – SQLite - Relational DB engine
  – SSL – Secure sockets layer
Android Runtime

- Basic support services for executing android applications
- Comprised of
  - Core Libraries
  - Dalvik Virtual Machine

Core Libraries

- Core Java classes providing basic services used by most applications
  - java.*
  - javax.*
  - junit.*
  - org.apache.*
  - org.json.*
  - org.xml.*
  - android.*,
- Does not include some classes from the standard Java SDK
Dalvik Virtual Machine

- Android applications typically written in Java,
  - But do not run in a Java VM
- The dx program transforms java classes into .dex-formatted bytecodes
- Bytecodes then executed by the Dalvik Virtual Machine
- Each android application (or part) runs in its own process, inside its own instance of the Dalvik VM

Dalvik Virtual Machine (cont.)

- The Dalvik VM is designed to run on a handset
  - Slow CPU
  - Little RAM (64Mb total, maybe only 10Mb available at runtime)
  - No swap space
  - Limited battery life

- Some design choices
  - 1 .dex file for multiple classes
  - Modified garbage collection to improve memory sharing
  - Optimizations at installation time
  - register-based, rather than stack-based
Dalvik Virtual Machine (cont.)

- Some design choices
- Memory
  - .dex file has 1 common constant pools for multiple classes
  - Modified garbage collection to improve memory sharing
- CPU
  - Optimizations at installation time
  - register-based, rather than stack-based

Using Registers

- Expected benefits over stack-based VMs
  - Avoids (slow) instruction dispatch
  - Avoids unnecessary memory accesses
  - More efficient instruction stream
Dalvik Virtual Machine (cont.)

```java
public static long sumArray(int[] arr) {
    long sum = 0;
    for (int i : arr) {sum += i; }
    return sum;
}
```

Java Bytecode

```
0:   lconst_0
1:   istore_1
2:   aload_0
3:   astore_3
4:   aload_3
5:   arraylength
6:   istore 4
8:   icall 0
9:   istore 5
11:  iinc 5, 1
13:  if_icmpge 36
15:  iload 4
16:  iload 5
17:  iadd
18:  lstore_1
20:  iinc 5, 1
22:  goto -11
23:  lload_1
24:  lreturn
```
Dex Bytecode

0000: const-wide/16 v0, #long 0 // #0000
0002: array-length v2, v8
0003: const/4 v3, #int 0 // #0
0004: move v7, v3
0005: move-wide v3, v0
0006: move v0, v7
0007: if-ge v0, v2, 0010 // +0009
0009: aget v1, v8, v0
000b: int-to-long v5, v1
000c: add-long/2addr v3, v5
000d: add-int/lit8 v0, v0, #int 1 // #01
000f: goto 0007 // -0008
0010: return-wide v3

Tradeoffs

- Dan Bornstein of Google reported that register-based VMs used:
- (+) 30% fewer instructions
- (+) 35% fewer code units (1-byte vs. 2-byte instructions)
- (?) 35% more bytes in the instruction stream
  - but can consume instructions two bytes at a time
Application Framework

- Window Manager
  - Manages top-level window’s look & behavior
- View system
  - UI elements such as: lists, grids, text boxes, buttons, etc.
- Content Providers
  - Allows accessing & publishing data
- Activity Manager
  - Application lifecycle and common navigation stack

Application Framework (cont.)

- Package manager
  - Info about application packages
- Telephony manager
  - Info about state of telephony services
- Resource Manager
  - Access to non-code resources: e.g., strings, graphics, and layout files
- Location manager
  - Access to system location services
- Notification Manager
  - Display custom alerts in the status bar
Applications

• Standard apps include:
  – Home – main screen
  – Contacts – contacts database
  – Phone – dial phone numbers
  – Browser – uses WebKit
  – Email reader – Gmail & others
• Your App™
• Any application can be replaced