**Budankov Alexey**

Q1: How to send messages to logcat from native code?

A1: See the examples in our course, chapter 9.7-9.8 (slide 218). Code example:

*SLOGE("Failed to open %s: %s", LOG\_FILE, strerror(errno)); /\* 1 \*/*

*SLOGV("Flushing %s", LOG\_FILE);*

*SLOGI("Flushed log (%d, %d of %d bytes). Waiting %d second before the next flush.", count, usedSize, totalSize, frequency);*

etc

platform implementation done in

*#include <cutils/log.h>*

*#include <cutils/logger.h>*

Q2: Can the logs be deactivated system wide?

A2:

**Lyalin Sergey**

Q1: how to query the power consumption for the whole system and also for particular parts (CPU, display etc)

A1: Check the tech paper from Google <https://source.android.com/devices/tech/power.html>

Check the official platform implementation of Battery tab in Settings app, the project app is located at <https://github.com/android/platform_packages_apps_settings> and the class doing all the magic is <https://github.com/android/platform_packages_apps_settings/blob/master/src/com/android/settings/BatteryInfo.java>

Check the Intel PowerTop open source project at <https://01.org/powertop/> and sources at: <https://github.com/fenrus75/powertop>

The *BatteryInfo.java* has an included *BroadcastReceiver* for *Intent.ACTION\_BATTERY\_CHANGED* and is taking from the passed Intent most of

the informations need it like temperature, etc.

Check the Power Tutor application website at : <http://ziyang.eecs.umich.edu/projects/powertutor/> and the source code is available at: <https://github.com/msg555/PowerTutor>

Check also the SysPower app website <https://code.google.com/p/syspower/> and the source code is available at: <https://code.google.com/p/syspower/>

Also command line alternatives + java code alternatives:

*adb shell dumpsys batteryinfo*

*public static void dumpBatteryInfo() {*

*try {*

*String cmd = "dumpsys battery";*

*Process script = Runtime.getRuntime().exec(cmd);*

*BufferedReader in = new BufferedReader(*

*new InputStreamReader(script.getInputStream()));*

*String line = null;*

*while ((line = in.readLine()) != null) {*

*Log.i ("BATTERY","Battery stats: " + line);*

*}*

*} catch (Exception ex) {*

*}*

*}*

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

**Panov Roman**

Q1: UI development, some example of custom controls;

A1: Check the example provided in Intel internals github - <https://github.com/mailat/internals-2013-11-11/tree/master/Wishes>

Examples provided in the git repository:

* 00\_CustomViewDraw – custom Pie component example via SDK sample
* 00\_Paint\_2D – Marakana example on how you can pain on an Canvas, a custom component DrawView
* 12\_Styles – how you can use styles, inherit styles
* OaiaAnimata – a custom SurfaceView with a animated threaded sheep
* WheelMenu – custom control with a plate that can be moved

**Korobycin Slava**

Q1: Content Providers example

A1: Check the example provided in Intel internals github - <https://github.com/mailat/internals-2013-11-11/tree/master/Wishes>

Examples provided in the git repository:

* 02\_ContentProvider – access to the list of Contacts via Contacts Provider (see ContactsActivity.java)
* 02\_ContentProvider – access to the images in gallery via MediaStore Provider (see ImageGalleryActivity.java)

**Cheportuzov Artyom**

Q1: Simple app to react on on poweroff and reboot

A1: Check the example provided in Intel internals github - <https://github.com/mailat/internals-2013-11-11/tree/master/Wishes>

I am using in these cases 2 separate Broadcast receivers ( reference <http://developer.android.com/reference/android/content/BroadcastReceiver.html> ) for catching 2 events. In the AndroidManifest.xml we must have the permission to catch these events. The application has minimal interface and is logging these states.

Examples provided in the git repository:

* 13\_BroadcastReceiver – catch the reboot event (see RebootReceiver.java)
* 13\_BroadcastReceiver – catch the poweroff event (see PoweroffReceiver.java)

**Kukanov Alexey**

Q1: 3rd party library used in my application, native library

A1: In the course materials (slide 200) we have in the 4th day a complete example on “Exposing our Native Library via Java (i.e. JNI)”. Additionally in the ANDROID/NDK/SAMPLES folder we have some examples on how the communication is done.

The theoretical part into the course is located at slide 54.

**Deev Vladimir**

Q1: After flashing a device how to test some features ( not explorative testing, but ui testing )?

A1:

**Kasatkina Olga**

Q1: Automatic test UI

A1:

**Kozyrev Alexander**

Q1: How to manage the activity stack in c++ ?

A1:

**Krasichkov Eugene**

Q1: If ( not Inject own library into a not rooted process without debuggable permissions )

* else { react on other applications launches }

A1:

**Kukanova Svetlana**

Q1: I’m interested in setting this property - *debug.atrace.tags.enableflags*

and notifying the system that the property has changed.

atrace.cpp in 4.4 or 4.3 (in 4.2.2 it’s atrace.c and it doesn’t do notification)

sets this property with some propert\_set func (from <cutils/properties.h>)

and then notifies the system with this:

#include <binder/IBinder.h>

#include <binder/IServiceManager.h>

#include <binder/Parcel.h>

….

sp<IServiceManager> sm = defaultServiceManager();

Vector<String16> services = sm->listServices();

for (size\_t i = 0; i < services.size(); i++) {

sp<IBinder> obj = sm->checkService(services[i]);

if (obj != NULL) {

Parcel data;

if (obj->transact(IBinder::SYSPROPS\_TRANSACTION, data,

NULL, 0) != OK)

…

but atrace is a part of Android build.

Would be great to find a way to do all the same from a c++ app built with NDK.

Or to have 100% evidence that it’s not possibleJ

Actually doing this from java code could also be an option, though I’m not sure we’ll be able to make use of it, because calling java would probably be slow.

The last option is to ask the user to set this property in the developer options, but this is not that niceJ

A1:

**Malyshev Andrey**

Q1: Mechanism pattern for „sliding the screens“

A1:

Q2: Gestures?

A2: See next an answer A3. A good example with a small app for gesture detection is to be found on the official Android page : <http://developer.android.com/training/gestures/detector.html> and basically is using the same mechanism as described in A3.

Q3: Event mechanism, how to intercept different types of touch gestures (singleplay, long play, zoom etc) – complex

A3: Check the events in the project Wheel Menu from the github. The class where this is visible is located in *customcontrols/WheelMenuSurface.java* in the method *onTouchedEvent*:

*public boolean onTouchEvent(MotionEvent event) {*

*if (event.getAction() == MotionEvent.ACTION\_DOWN)*

*if (event.getAction() == MotionEvent.ACTION\_UP)*

*if (event.getAction() == MotionEvent.ACTION\_MOVE)*

Q4: Sandboxing and execute system calls (what is possible)

A4:

Q5: How to run the compiled emulator on the Windows, which files are need it?

A5:

Q6: How to upload custom images to the phone?

A6: For each device groups are different way of flashing a device. For most nexus devices is enough to follow the steps bellow:

* have a file update.zip in the root of the device
* reboot in recovery mode using some combinations of keyboard
* choose the update.zip file for flashing the device
* flash the device
* reboot the device

Official documentation is to be found on <http://source.android.com/source/building-devices.html> All depends on the bootloader and it is also possible for some devices to not be able to flash an image! My recommended resource is the one for compiling the Android 4.4 for Nexus 4 with steps also for installation and the fastboot command line at the end:

* How to build Android AOSP for Nexus 4 - <http://nosemaj.org/howto-build-android-nexus-4>
* How to build Android 4.3 for Nexus 4 - <http://nosemaj.org/build-android-4-3-nexus-4>

Nexus 5 drivers are already available at <https://developers.google.com/android/nexus/drivers> so theoretically is possible to do a Android 4.4 build for Nexus 5!

**Matrosova Ekaterina**

Q1: I run the shell in android phone and I run the apps from command line, how I can debug these crashes, dumps etc

A1:

**Perepelkin Sergey**

Q1: NDK samples

A1:

**Ryabtsev Dmitry**

Q1: Bad html 5 apps and reasons why

A1: TBD by marius

**Tutin Michael**

Q1: How to communicate between Android layers; how to pass data

A1:

**Lazarev Evgeny**

Q1: Show on the top of the application a visible layer with information

A1:

**Leskinen Eugene**

Q1: Java and native parts in the app; the apps should use some Android specific features

A1:

**Aranovsky Sergei**

Q1: how I can subscribe for some events from native code

A1:

Q2: RenderScript example

A2:

Q3: Split screen apps running in the same time

A3: