**Tibi**

Q1: DDMS (Dalvik Debug Monitors Server) debugging

A1: Resursele ajutatoare din curs se regasesc la

1. Pag 226-243 in curs
2. Vineri mai mult insight despre kernel debugging (Dmesg), native debugging (GDB) si platform/app debugging ( via DDMS si JDWP – Java Debug Wire Protocol )
3. Instructiuni de pas cu pas debugging NDK - <http://mhandroid.wordpress.com/2011/01/23/using-eclipse-for-android-cc-debugging/>
4. Verifica articolul lui Larry - <http://www.eweek.com/c/a/Linux-and-Open-Source/How-to-Set-Up-Android-Platform-Development-and-Debugging/>
5. Un articol legat de alternative tombstone, ndk-stack si addr2line - <http://bytesthink.com/blog/?p=133>

**Razvan**

Q1: JNI

A1: Resursele ajutatoare din curs se regasesc la:

* Pagina 53-95 NDK si JNI
* Exemplul cu FibonacciNative se gaseste la pagina 76
* Implementarea de java library folosind JNI se gaseste la pagina 219
* Ca referinta ia si exemplele din /android/ndk/samples
* The New Circle ofera si un tutorial mai simplificat la <https://thenewcircle.com/s/post/49/using_ndk_to_call_c_code_from_android_apps>

**Eduard**

Q1: Interactiune cu aplicatii (QA si nu numai)

A2: Official UI testing <http://developer.android.com/tools/testing/testing_ui.html> with alternatives Robotium - <https://code.google.com/p/robotium/> and Testdroid/Testrecorder - <http://bitbar.com/>

A2: Introducere in monkeyrunner (alternative)

Q2:

* Scurta introducere via <http://developer.android.com/tools/help/monkeyrunner_concepts.html>
* Mai mult despre Testdroid/Testrecorder - <http://bitbar.com/>

**Bogdan**

Q1: Debugging low level, subcomponentele sistemului.

A1: Resursele ajutatoare din curs se regasesc la

1. Pag 226-243 in curs
2. Vineri mai mult insight despre kernel debugging (Dmesg), native debugging (GDB) si platform/app debugging ( via DDMS si JDWP )
3. Instructiuni de pas cu pas debugging NDK - <http://mhandroid.wordpress.com/2011/01/23/using-eclipse-for-android-cc-debugging/>
4. Verifica articolul lui Larry - http://www.eweek.com/c/a/Linux-and-Open-Source/How-to-Set-Up-Android-Platform-Development-and-Debugging/
5. Un articol legat de alternative tombstone, ndk-stack si addr2line - <http://bytesthink.com/blog/?p=133>

Q2: Sistemul de build (modul de organizare)

A2: Resursele ajutatoare din curs se regasesc la pag 226-243

**Viorel**

Q1: Debugging

A1: Resursele ajutatoare din curs se regasesc la

1. Pag 226-243 in curs
2. Vineri mai mult insight despre kernel debugging (Dmesg), native debugging (GDB) si platform/app debugging ( via DDMS si JDWP )
3. Instructiuni de pas cu pas debugging NDK - <http://mhandroid.wordpress.com/2011/01/23/using-eclipse-for-android-cc-debugging/>
4. Verifica articolul lui Larry - http://www.eweek.com/c/a/Linux-and-Open-Source/How-to-Set-Up-Android-Platform-Development-and-Debugging/
5. Un articol legat de alternative tombstone, ndk-stack si addr2line - <http://bytesthink.com/blog/?p=133>

**Robert**

Q1: Nou director in systemul de build

A1: Am realizat acest lucru in capitolul customizing Android si build from source – la pagina 156-168 si 208-215.

Q2: Ueventd Daemon.

A2: Te pot ajuta cu:

* ueventd.rc este echivalentul Android udev rules, insa cu scop limitat la a mentiona permisiunile si detinatorul sistemelor de fisiere, directoare /ueventd.rc asculta via un socket event-uri de la kernel.
* Digg into the source code ☺ <https://android.googlesource.com/platform/system/core/+/master/init/ueventd.c>

**Alex**

Q1: Native

A2: Resursele ajutatoare din curs se regasesc la:

* Pagina 53-95 NDK si JNI
* Exemplul cu FibonacciNative se gaseste la pagina 76
* Implementarea de java library folosind JNI se gaseste la pagina 219
* Ca referinta ia si exemplele din /android/ndk/samples
* The New Circle ofera si un tutorial mai simplificat la <https://thenewcircle.com/s/post/49/using_ndk_to_call_c_code_from_android_apps>

**Catalin**

Q1: NDK si reguli de SELinux 4.4 (sockets, reguli de protectie trebuie definite)

A1:

Mai multe informatii la:

* Din Android 4.3, e doar in Permissive mode, long-shot project
* Pagina oficiala <http://source.android.com/devices/tech/security/se-linux.html>
* <http://www.xda-developers.com/android/easily-change-your-android-selinux-mode/>

Q2: IMA (integrity measurement architecture) pentru Android 4.4

A2:

Mai multe informatii la:

* Digg in cod: <https://android.googlesource.com/kernel/omap.git/+/android-omap-tuna-3.0-ics-mr1/security/integrity/ima/ima_api.c>
* Beyond Kernel-level Integrity Measurement: Enabling Remote Attestation for the Android Platform - <http://profsandhu.com/zhang/pub/trust10-android.pdf>
* Building Efficient IntegrityMeasurement and Attestation forMobile Phone Platforms - <http://goo.gl/dn8FgU>

Q3: Securitate

A3: Am acoperit in capitolul de securitate 124-155 + goodies legata de multiuser, aplicatia de securitate.

Goodies:

Decompilare aplicatie Android:

* APKTool D BANK.Apk
* Jar xvf BANK.apk classes.dex
* dex2jar.sh classes.dex
* deschidem cu un java decompiler

Copiere de pe device non-rooted a aplicatiilor instalate:

* pm list packages -f
* adb pull /data/app/com.android.chrome-1.apk

**Mihai**

Q1: Layoutul de partitii

A1: Mai multe in curs la pagina 134.

Q2: Binder

A2: Resursele din curs legat de Binder se gasesc la

* Binder (pagina 96-123)
* Exemplul facut de noi legat de Java binder service si un app care il consuma se gaseste la pagina 221, 222
* Am exemplificat si prin adb shell am start -a android.intent.action.VIEW; service call phone 2 s16 "+15084157509"; precum si analiza de folisre in Java a com.android.internal.telephony.ITelephony.Stub

**Costin**

Q1: Debugging si Binder, security

A1:

Legat de Binder, resursele din curs legat de Binder se gasesc la

* Binder (pagina 96-123)
* Exemplul facut de noi legat de binder service si un app care il consuma se gaseste la pagina 221, 222
* Am exemplificat si prin adb shell am start -a android.intent.action.VIEW; service call phone 2 s16 "+15084157509"; precum si analiza de folisre in Java a com.android.internal.telephony.ITelephony.Stub

Legat de debugging, resursele ajutatoare din curs se regasesc la

1. Pag 226-243 in curs
2. Vineri mai mult insight despre kernel debugging (Dmesg), native debugging (GDB) si platform/app debugging ( via DDMS si JDWP )
3. Instructiuni de pas cu pas debugging NDK - <http://mhandroid.wordpress.com/2011/01/23/using-eclipse-for-android-cc-debugging/>
4. Verifica articolul lui Larry - <http://www.eweek.com/c/a/Linux-and-Open-Source/How-to-Set-Up-Android-Platform-Development-and-Debugging/>
5. Un articol legat de alternative tombstone, ndk-stack si addr2line - <http://bytesthink.com/blog/?p=133>

Legat de securitate:

* am acoperit in capitolul de securitate 124-155
* + goodies legata de multiuser, aplicatia de securitate.

Q2: Input method managers

A2:

* Platforma Android platform ofera Input Method Framework (IMF) care iti permite crearea de on-screen input methods precum software keyboards via Android input method editors (IMEs)
* Create an input method - <http://developer.android.com/guide/topics/text/creating-input-method.html>
* Verifica si InputMethodManager la <http://developer.android.com/reference/android/view/inputmethod/InputMethodManager.html>

Q3: Opaque Binary Blob (OBB)

A3:

* Documentatia de OBB expension packs se gaseste la <http://developer.android.com/google/play/expansion-files.html>
* Modul de folosire a utilitarului *jobb* pentru a construi expension packs Opaque Binary Blob (OBB) - <http://developer.android.com/tools/help/jobb.html>

**Alex**

Q1: Debugging si Binder

A1:

Legat de Binder, resursele din curs legat de Binder se gasesc la

* Binder (pagina 96-123)
* Exemplul facut de noi legat de binder service si un app care il consuma se gaseste la pagina 221, 222
* Am exemplificat si prin adb shell am start -a android.intent.action.VIEW; service call phone 2 s16 "+15084157509"; precum si analiza de folisre in Java a com.android.internal.telephony.ITelephony.Stub

Legat de debugging, resursele ajutatoare din curs se regasesc la

1. Pag 226-243 in curs
2. Vineri mai mult insight despre kernel debugging (Dmesg), native debugging (GDB) si platform/app debugging ( via DDMS si JDWP )
3. Instructiuni de pas cu pas debugging NDK - <http://mhandroid.wordpress.com/2011/01/23/using-eclipse-for-android-cc-debugging/>
4. Verifica articolul lui Larry - http://www.eweek.com/c/a/Linux-and-Open-Source/How-to-Set-Up-Android-Platform-Development-and-Debugging/
5. Un articol legat de alternative tombstone, ndk-stack si addr2line - <http://bytesthink.com/blog/?p=133>

**Rares**

Q1: Debugging si interactiune intre kernel HAL si framework

A1:

Legat de interactiune intre kernel HAL si framework, resursele din curs legat de Binder se gasesc la

* Layerul nativ cu Bionic, HAL, Dalvik la pagina 16-23
* Native app si native library la pagina 217
* Native daemon la pagina 218
* Java library pentru Application Framework la pagina 219-220
* Custom Java Binder Service si o aplicatie care il consuma la pagina 221-222

Legat de debugging, resursele ajutatoare din curs se regasesc la

1. Pag 226-243 in curs
2. Vineri mai mult insight despre kernel debugging (Dmesg), native debugging (GDB) si platform/app debugging ( via DDMS si JDWP )
3. Instructiuni de pas cu pas debugging NDK - <http://mhandroid.wordpress.com/2011/01/23/using-eclipse-for-android-cc-debugging/>
4. Verifica articolul lui Larry - http://www.eweek.com/c/a/Linux-and-Open-Source/How-to-Set-Up-Android-Platform-Development-and-Debugging/
5. Un articol legat de alternative tombstone, ndk-stack si addr2line - <http://bytesthink.com/blog/?p=133>

Q2: Secventa de boot, stop

A2:

* Secventa de boot o avem descrisa extins la pagina 173-178
* Secventa de re-boot este analizata in clasa si este realizata prin analizarea logurilor telefonului, incepe cu ShutdownThread .

**Daniel**

Q1: Makefile

A1: In curs informatia legata de makefile se gaseste la pagina 211

**Vali**

Q1: JNI, aplicatii si servicii native

A1:

Legat de intreaga integrare native, resursele din curs legat gasesc la:

* Layerul nativ cu Bionic, HAL, Dalvik la pagina 16-23
* Native app si native library la pagina 217
* Native daemon la pagina 218
* *Optional*: Java library pentru Application Framework la pagina 219-220
* *Optional*: Custom Java Binder Service si o aplicatie care il consuma la pagina 221-222

**Adrian**

Q1: Binder

A1:

Legat de Binder, resursele din curs legat de Binder se gasesc la

* Binder (pagina 96-123)
* Exemplul facut de noi legat de binder service si un app care il consuma se gaseste la pagina 221, 222
* Am exemplificat si prin adb shell am start -a android.intent.action.VIEW; service call phone 2 s16 "+15084157509"; precum si analiza de folisre in Java a com.android.internal.telephony.ITelephony.Stub

Q2: ART

A2: Resurse:

* Oficial avem <http://source.android.com/devices/tech/dalvik/art.html> si cam atat
* Check inovex post on it <https://plus.google.com/+MatthiasSchaff/posts/U1jhMAv3d7R>

Q3: Modes de operare Dalvik

A3: Nu exista parametrii de lansare Dalvik in modul in care ii cauti, poti folosi insa setarea de proprietati, check <http://www.netmite.com/android/mydroid/2.0/dalvik/docs/embedded-vm-control.html>

Q4: Kernel goldfish

A4: Probabil nu te asteptai la asta , <https://android.googlesource.com/kernel/goldfish/+/android-goldfish-3.4>

**Adriana**

Q1: Runtime

A1: Am discutat despre Dalvik la 16-23.

Q2: ART

A2: Resurse:

* Oficial avem <http://source.android.com/devices/tech/dalvik/art.html> si cam atat
* Check inovex post on it <https://plus.google.com/+MatthiasSchaff/posts/U1jhMAv3d7R>

Q3: Native daemons

A3:

Resursele din curs legat native daemons se gasesc la :

* Native app si native library la pagina 217
* Native daemon la pagina 218

**Stefan**

Q1: Build de Nexus 5 din sursele AOSP si punere pe telefon?

A1: Pentru fiecare tip de telefoane avem un mod aproape unic de flashing, pentru majoritatea device-urilor Nexus avem pasii:

* Copiem un fisier update.zip in radacina sd card
* reboot in recovery mode folosind combinatiile de telefon (power+vol up?!)
* alegem update.zip
* flash device
* reboot device

**Cum luam imaginea compilata?**

*make updatepackage*

*fastboot -w update $ANDROID\_PRODUCT\_OUT/$TARGET\_PRODUCT-img-eng.$USER.zip*

* documentatia oficiala de flashing e la <http://source.android.com/source/building-devices.html> insa totul depinde de bootloader
* 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 4 si Nexus 5 - <https://developers.google.com/android/nexus/drivers>
* Howto Build Android KitKat (4.4) for the Google Nexus 5 - <http://nosemaj.org/howto-build-android-kitkat-nexus-5>

Q2: Extindere de servicii Android cu functionalitati.

A2:

Resursele din curs legat de servicii Android native sau in Java se gasesc la

* Exemplul facut de noi legat de Java binder service si un app care il consuma se gaseste la pagina 221, 222
* Native app si native library la pagina 217
* Native daemon/service la pagina 218

**Irina**

Q1: Native

A1: NDK in curs la pagina 53-95. Solutia oferita prin system este bazata pe urmatoarea interactiune:

* marakana HAL
* folosire in binar
* folosita in un Daemon
* expusa prin JNI in Application Framework
* Android app Java care foloseste Application Framework)

Q2: Binder

A2:

Resursele din curs legat de Binder se gasesc la

* Binder (pagina 96-123)
* Exemplul facut de noi legat de binder service si un app care il consuma se gaseste la pagina 221, 222
* Am exemplificat si prin adb shell am start -a android.intent.action.VIEW; service call phone 2 s16 "+15084157509"; precum si analiza de folisre in Java a com.android.internal.telephony.ITelephony.Stub