Android : Architecture - Compilation - Debug

Sylvain Jubertie - Université d'Orléans

2011-2012

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへぐ

- 2 Creation d'un projet
- 3 Compilation d'un projet
- 4 Installation d'une application
- 5 Debugging
- 6 Distribution

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへぐ

- 2 Creation d'un projet
- 3 Compilation d'un projet
- 4 Installation d'une application
- 5 Debugging
- 6 Distribution

Installation du SDK et du NDK

1 Télécharger les archives suivantes :

- Android SDK : Software Development Kit
- Android NDK : Native Development Kit

à partir du site http://developer.android.com

- 2 décompresser les archives
- 3 mettre à jour la variable d'environnement PATH :

PATH=\$PATH:path_to_android-sdk/tools: \
path_to_android-sdk/platform-tools: \
path_to_android-ndk

Attention

Ces archives ne sont pas suffisantes pour développer !

Installation des plateformes

Chaque développement pour une version d'Android nécessite l'installation de la plateforme correspondante :

- 1 lancer la commande android
- 2 dans la rubrique Available packages choisir les plateformes Android cibles à installer : SDK Platform, samples, doc



◆□▶ ◆□▶ ◆□▶ ◆□▶ ●□

Emulateur

Le SDK fournit un émulateur de périphérique Android configurable :

- version d'Android
- dimension de l'écran
- taille mémoire
- support de périphériques

...

Une configuration pour cet émulateur est se nomme un AVD : Android Virtual Device.

Création d'un AVD Android Virtual Device

Dans la rubrique Virtual devices, choisir New... et configurer un AVD : nom, version d'Android (Target), ...



Une fois créé, l'AVD peut être lancé par Start...



◆□▶ ◆□▶ ◆ 臣▶ ◆ 臣▶ ○ 臣 ○ の Q @

Inconvénient

L'émulateur est extrêmement lent ! Plus loin : test/debuggage directement sur le *device*.

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへぐ

1 Installation des outils

- 2 Creation d'un projet
- 3 Compilation d'un projet
- 4 Installation d'une application
- 5 Debugging
- 6 Distribution

▲□▶ ▲圖▶ ★ 国▶ ★ 国▶ - 国 - のへで

└─ Choix de la plateforme cible

Choix de la plateforme cible

Un projet est créé pour une cible parmi les cibles installées. La liste des cibles s'obtient par la commande : android list targets

Exemple

```
$ android list targets
Available Android targets:
id: 1 or "android-13"
    Name: Android 3.2
    Type: Platform
    API level: 13
    Revision: 1
    Skins: WXGA (default)
```

Application

Création d'un projet

Dans un nouveau dossier :

android create project \
--target <target_ID> \
--name <project_name> \
--path path/to/your/project \
--activity <activity_name> \
--package <package_namespace>

▲ロト ▲帰ト ▲ヨト ▲ヨト 三日 - の々ぐ

Application

Arborescence d'un projet

Dossiers créés automatiquement lors de la création d'un projet :

- bin : binaires générés
- libs : bibliothèques
- res : fichiers de ressources (icones, layout, ...)
- src : fichiers sources (code Java)

▲ロト ▲帰ト ▲ヨト ▲ヨト 三日 - の々ぐ

Application

Fichiers générés

Les fichiers suivants sont créés automatiquement lors de la création d'un projet :

- AndroidManifest.xml
- build.properties
- build.xml
- default.properties
- local.properties
- proguard.cfg

Application

Modification d'un projet

En cas de modification du nom d'un projet, de la cible ou du chemin :

```
android update project \
---name <your_project_name> \
---target <target_ID> \
---path path/to/your/project
```

Bibliothèque

Création d'une bibliothèque

De manière similaire à un projet standard :

```
android create lib-project --name <your_project_name> \
--target <target_ID> \
--path path/to/your/project \
--package <your_library_package_namespace>
```

Intégration de code natif

Intégration de code natif

- repose sur JNI Java Native Interface
- supporte les jeux d'instructions : ARMv5TE, ARMv7-A,x86
- réutilisation de bibliothèques C/C++
- amélioration de performance sur des codes de calculs
- OpenGL
- NEON
- attention à la gestion mémoire !

└─ Intégration de code natif

Principe de l'intégration de code natif

- **1** Ecrire un code natif C/C++ + wrapper JNI
- 2 Le code natif C/C++ est compilé sous forme de bibliothèque dynamique .so
- 3 Une bibliothèque est générée par architecture désirée
- Le ou les bibliothèques .so sont intégrées à l'application Android
- La bibliothèque correspondant au matériel et à la configuration du système Android hôte est automatiquement appelée lors de l'appel à la méthode native

Implications

- Augmentation de la taille de l'application
- Si une bibliothèque est générée pour le jeu ARMv7-A,
 l'application no fonctionnera pas sur les processour plus

Intégration de code natif

Intégration dans un projet Android

- Créer un dossier jni à la racine du projet, puis dans ce dossier,
- 2 placer les fichiers C/C++
- 3 créer un fichier Android.mk (fichier Makefile) qui permettra de générer la bibliothèque
- Dans le code Java de l'application, créer un wrapper vers les fonctions natives (partie abordée plus tard)

- 2 Creation d'un projet
- 3 Compilation d'un projet
- 4 Installation d'une application
- 5 Debugging
- 6 Distribution

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへぐ

Android : Architecture - Compilation - Debug

Compilation d'un projet



Compilation d'un projet

Etapes et commandes pour la compilation

- Si du code natif est présent, générer la ou les bibliothèques : ndk-build Les bibliothèques sont placées dans un sous-dossier du dossier lib
- 2 Compiler le code Java et générer un package .apk : ant {debug | release} Choisir debug ou release suivant le mode désiré

Compilation d'un projet

Modes de compilation

- debug : l'application est automatiquement signée avec une clé de debug connue
- release : l'application n'est pas signée

Le mode debug permet de déployer rapidement une application lors de la phase de debug.

Le mode release indique que l'on souhaite diffuser l'application générée qui doit alors être signée avec une clé privée (cf section Distribution).

Compilation d'un projet

Explications sur l'alignement

Pour améliorer les performances des applications il convient d'aligner correctement les données en mémoire pour diminuer le nombre de lecture.

Cette contrainte est liée au bus mémoire qui accède aux données par blocs contigüs de 32 bits (en général) soit 4 octets.

Alignement de l'application

Pour aligner sur 4 octets, utiliser la commande : zipalign -v 4 unsigned.apk signed.apk

- 2 Creation d'un projet
- 3 Compilation d'un projet
- 4 Installation d'une application
- 5 Debugging

6 Distribution

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへぐ

Installation d'une application

Installation

Une application est installée sur un AVD ou sur un matériel à l'aide de la commande : adb install <package>.apk

Si plusieurs AVD ou matériels

Il est possible d'utiliser les options -s et -d pour spécifier respectivement les identifiants de l'AVD ou du périphérique cible.

Installation d'une application

Liste des périphériques

adb devices

◆□▶ ◆□▶ ◆三▶ ◆三▶ 三三 のへぐ

Debugging

1 Installation des outils

- 2 Creation d'un projet
- 3 Compilation d'un projet
- 4 Installation d'une application

5 Debugging

6 Distribution

Debugging

▲□▶ ▲□▶ ▲□▶ ▲□▶ ▲□ ● ● ●

ADB Android Debug Bridge

L'outil en ligne de commande adb permet d'obtenir des informations de debuggage.

Distribution

1 Installation des outils

- 2 Creation d'un projet
- 3 Compilation d'un projet
- 4 Installation d'une application
- 5 Debugging

6 Distribution

Distribution

Génération d'une clé

keytool -genkey -v -keystore path_to_keystore -alias
rkey -keyalg RSA -keysize ... -validity ...

Signature

jarsigner -verbose -keystore path_to_keystore -signedjar signed.apk unsigned.apk rkey

▲ロト ▲帰ト ▲ヨト ▲ヨト 三日 - の々ぐ