Installation du support Android pour Qt5 (Ubuntu 16.04)

Thierry Vaira <tvaira@free.fr> http://tvaira.free.fr/

2017-2019 (rev. 1.1)

Table des matières

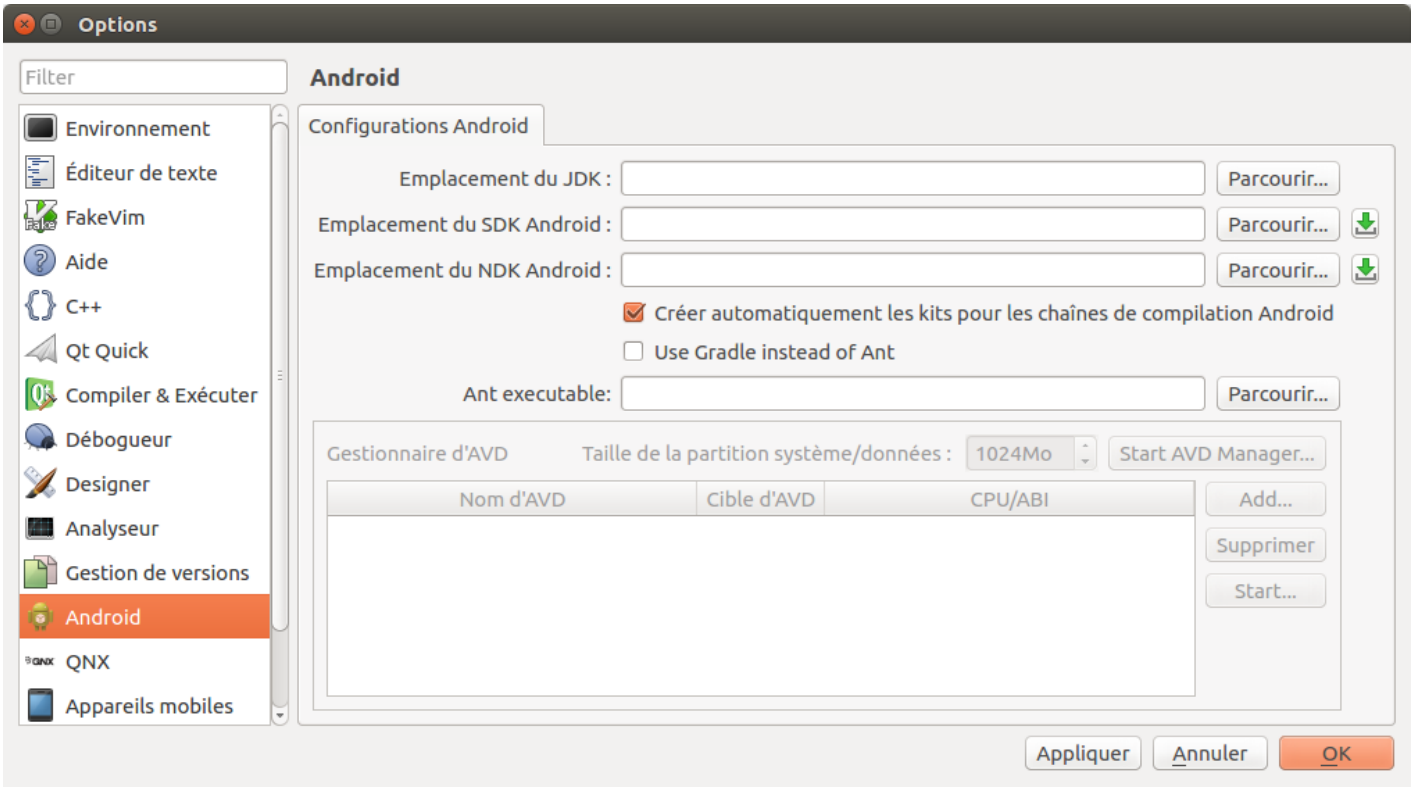
Installation du support Android pour Qt5 (Ubuntu 16.04)

Présentation

Le support d'Android pour Qt5 nécessite :

- un environnement de développement **Qt5**
- le kit de développement **Java SDK**
- l'**Android SDK** qui est un ensemble complet d'outils de développement incluant un débogueur, des bibliothèques logicielles, un émulateur basé sur QEMU, de la documentation, des exemples de code et des tutoriaux.
- l'**Android NDK** (*Android Native Development Kit*) qui est une API du système d'exploitation Android permettant de développer directement dans le langage C/C++ du matériel cible, par opposition au Android SDK qui est une abstraction en *bytecode* Java, indépendante du matériel.

La configuration du support Android se fera directement dans Qt Creator :



Remarque : **Android est un système d'exploitation mobile basé sur le noyau Linux et développé actuellement par Google. Android Studio est un environnement de développement pour développer des applications Android** en Java. Android Studio permet principalement d'éditer les fichiers Java et les fichiers de configuration d'une application Android.

Liens :

- Les différentes versions d'Android
- Support Qt Android
- Support Qt Android (Archives)

Version de Qt 5

Tous les modules Qt sont pris en charge à l'exception de Qt WebKit, Qt NFC, QSerialPort et des modules spécifiques à la plate-forme (Qt Mac Extras, Qt Windows Extras et Qt X11 Extras). De plus, le module Qt Multimedia Widgets n'est pas pris en charge sur Android, ce qui signifie que l'affichage vidéo est uniquement disponible à l'aide des éléments VideoOutput et Video QML.

Pour Qt 5.6 (ou une version antérieure), il faudra disposer des API Android 10, 11, 16 et 18. À partir de Qt 5.7, les API Android >= 18 fonctionneront.

Suivant vos besoins :

- L'API Android 11 est requise pour QtMultimedia.
- L'API Android 18 est nécessaire pour QtBluetooth (à partir de Qt 5.5).
- Les API Android 10, 11 et 16 sont requises pour QtBase.

Pour certaines version de Qt5 avec la chaîne d'outils gcc, vous aurez besoin de la version **r10e** d'Android NDK. Dans ce cas, l'API Android la plus élevée prise en charge par la version r10e est la 21.

Si votre version de Qt est antérieure à 5.9, il faudra prendre la version v25.2.5 (ou une version antérieure) des outils de développement d'Android SDK.

```
$ cd $HOME/Android/Sdk
$ wget http://dl.google.com/android/ndk/android-ndk-r10e-linux-x86.bin
$ chmod +x android-ndk-r10e-linux-x86.bin
$ ./android-ndk-r10e-linux-x86.bin
```

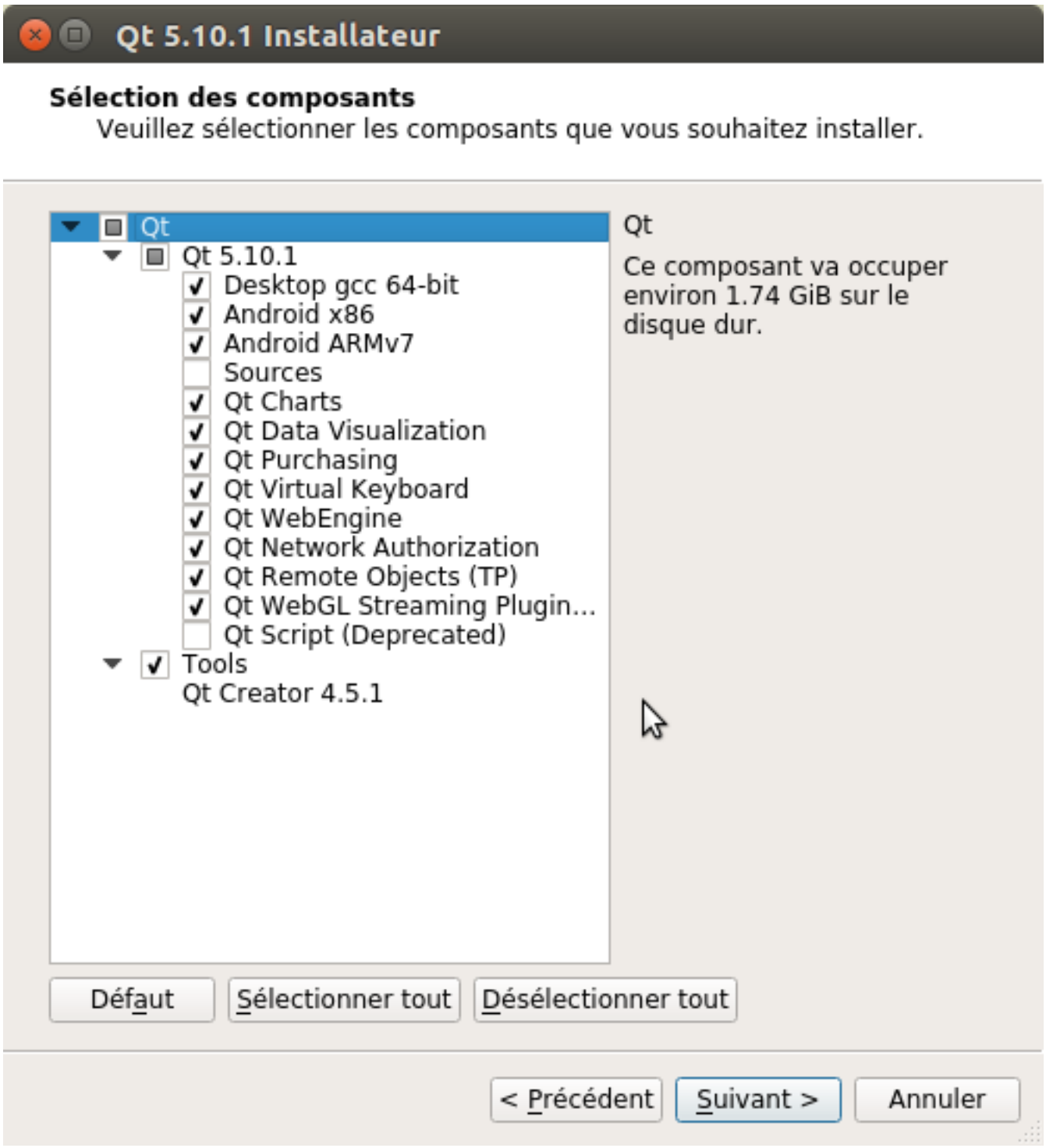


```
$ cd $HOME/Android/Sdk
$ wget https://dl.google.com/android/repository/tools_r25.2.5-linux.zip
$ mv tools tools.old
$ unzip tools_r25.2.5-linux.zip
```

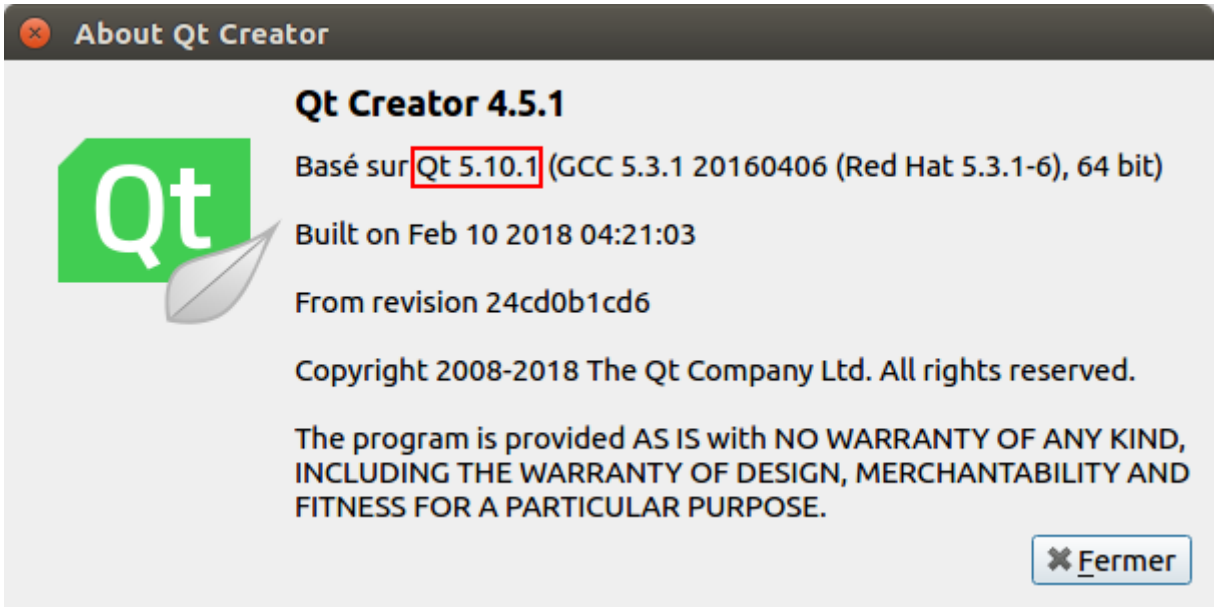
Installation Qt 5.10

On va installer la version Qt 5.10.1 sur une Ubuntu 16.04, disponible sur le site de Qt :

```
$ wget https://download.qt.io/archive/qt/5.10/5.10.1/qt-opensource-linux-x64-5.10.1.run
$ chmod +x qt-opensource-linux-x64-5.10.1.run

// Mono-utilisateur dans $HOME/Qt5.10.1/
$ ./qt-opensource-linux-x64-5.10.1.run

Ou

// Multi-utilisateur dans /opt/Qt5.10.1/
$ gksudo ./qt-opensource-linux-x64-5.10.1.run
```

Et on obtient au final :



Installation Java SE Development Kit

JRE (*Java Runtime Environement*) est le kit destiné au client pour pouvoir exécuter un programme Java. Il se compose essentiellement d'une machine virtuelle Java (JVM) capable d'exécuter le *bytecode* et les bibliothèques standard de Java.

Le kit destiné au programmeur, appelé avant JDK (*Java Development Kit*) et renommé depuis la version 1.2.2 en SDK (*Standard Development Kit*), est composé d'un JRE, d'un compilateur, de nombreux programmes utiles, d'exemples de programmes Java et des les sources de toutes les classes de l'API.

Le développement Android nécessite le SDK Java, le JRE seul n'est pas suffisant. **On a besoin de la version 1.8 de Java SDK.**

Si Java est déjà installé sur le poste, sa version peut être vérifée en ligne de commande par :

```
$ javac -version
```

Deux possibilités d'installation :

— depuis les paquets présents dans le dépôt :

```
$ sudo apt install openjdk-8-jdk    (ou default-jdk)

$ javac -version
javac 1.8.0_191
```

— ou sinon télécharger Java SE Development Kit 8 puis l'installer :

```
$ sudo cp jdk-8u102-linux-x64.tar.gz /usr/local

$ cd /usr/local/

$ sudo tar zxvf jdk-8u102-linux-x64.tar.gz

$ sudo rm jdk-8u102-linux-x64.tar.gz

$ vim $HOME/.bashrc

export PATH=/usr/local/jdk1.8.0_102/bin:$PATH

$ source $HOME/.bashrc
```


Installation du SDK Android (Android Studio)

Le SDK d'Android est fourni avec Android Studio.

Pour les systèmes 64-bits, il faudra tout d'abord installer les bibliothèques 32-bits suivantes :

```
$ sudo apt-get install libstdc++6:i386 libgcc1:i386 zlib1g:i386 libncurses5:i386
```

Et pour l'émulateur :

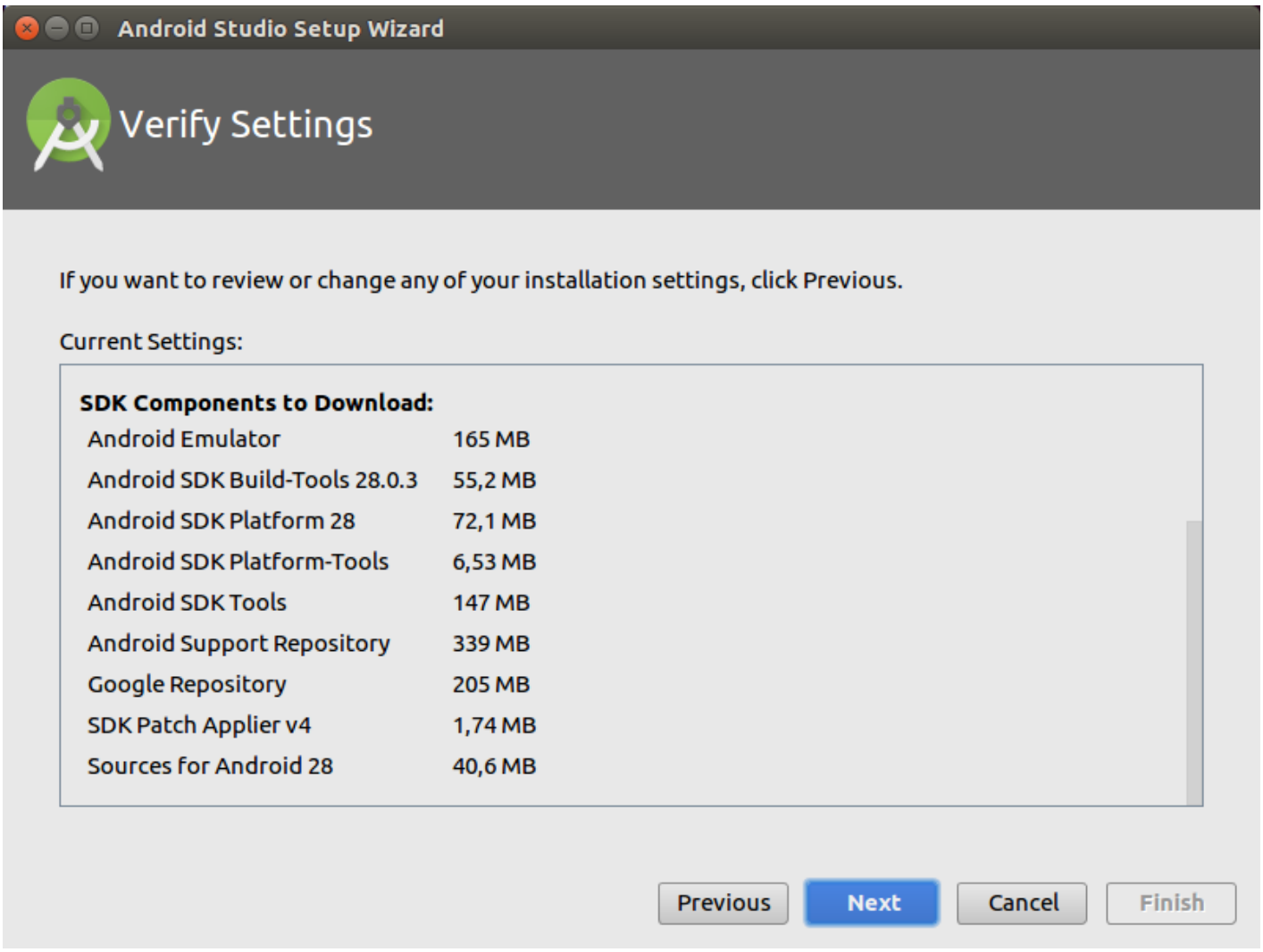
```
sudo apt-get install libsdl1.2debian:i386
```

Télécharger Android Studio puis l'installer :

```
$ sudo mv android-studio-ide-171.4443003-linux.zip /usr/local/

$ cd /usr/local/

$ sudo unzip android-studio-ide-171.4443003-linux.zip

$ sudo rm android-studio-ide-171.4443003-linux.zip
```

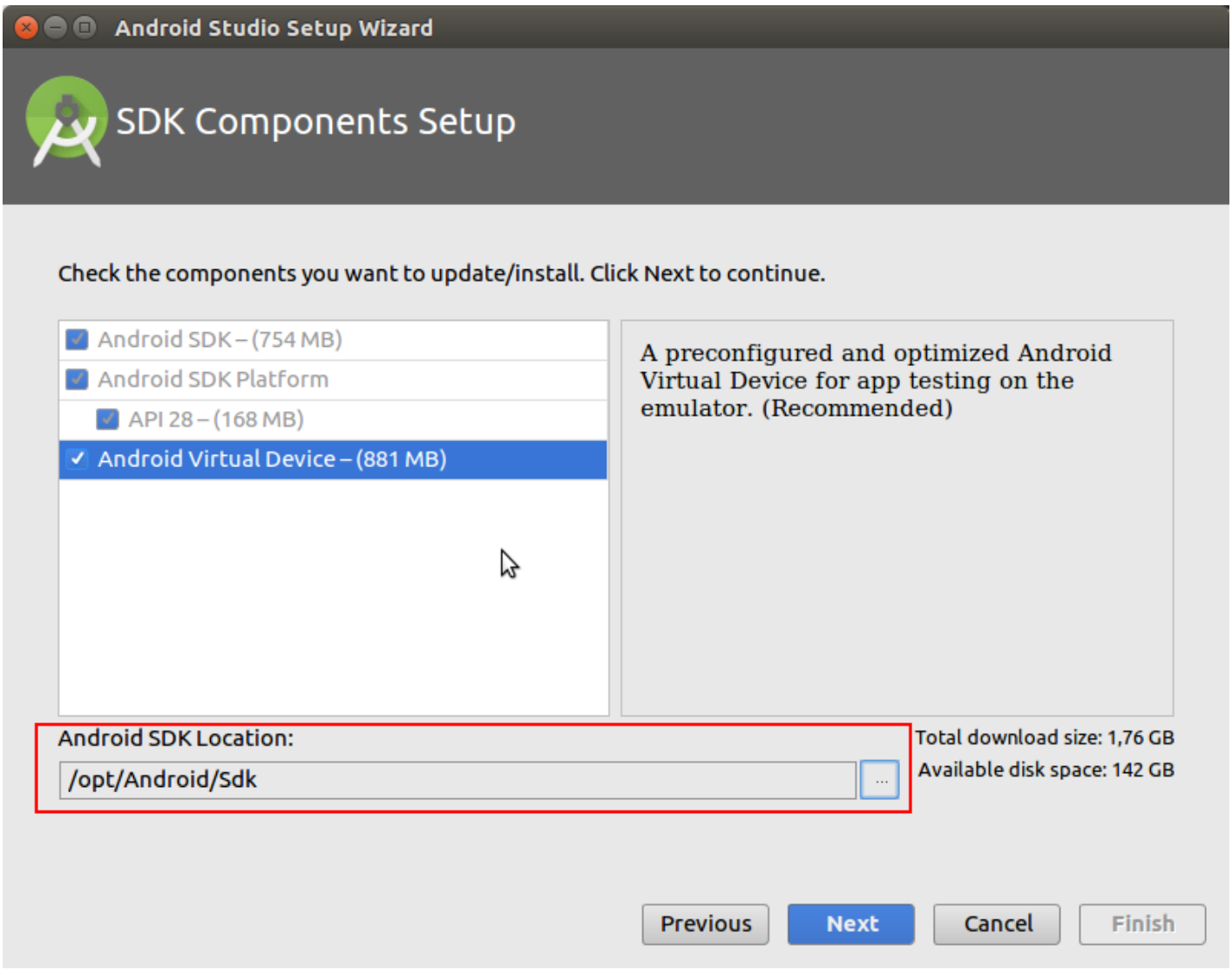
— Pour une installation mono utilisateur dans `$HOME/Android` :

```
$ vim $HOME/.bashrc

export PATH=$PATH:/usr/local/android-studio/bin:$HOME/Android/Sdk/platform-tools:$HOME/Android/Sdk/tools:$HOME/
    Android/Sdk/tools/bin

$ source $HOME/.bashrc

$ studio.sh
```

— Pour une installation multi utilisateur dans `/opt/Android` :

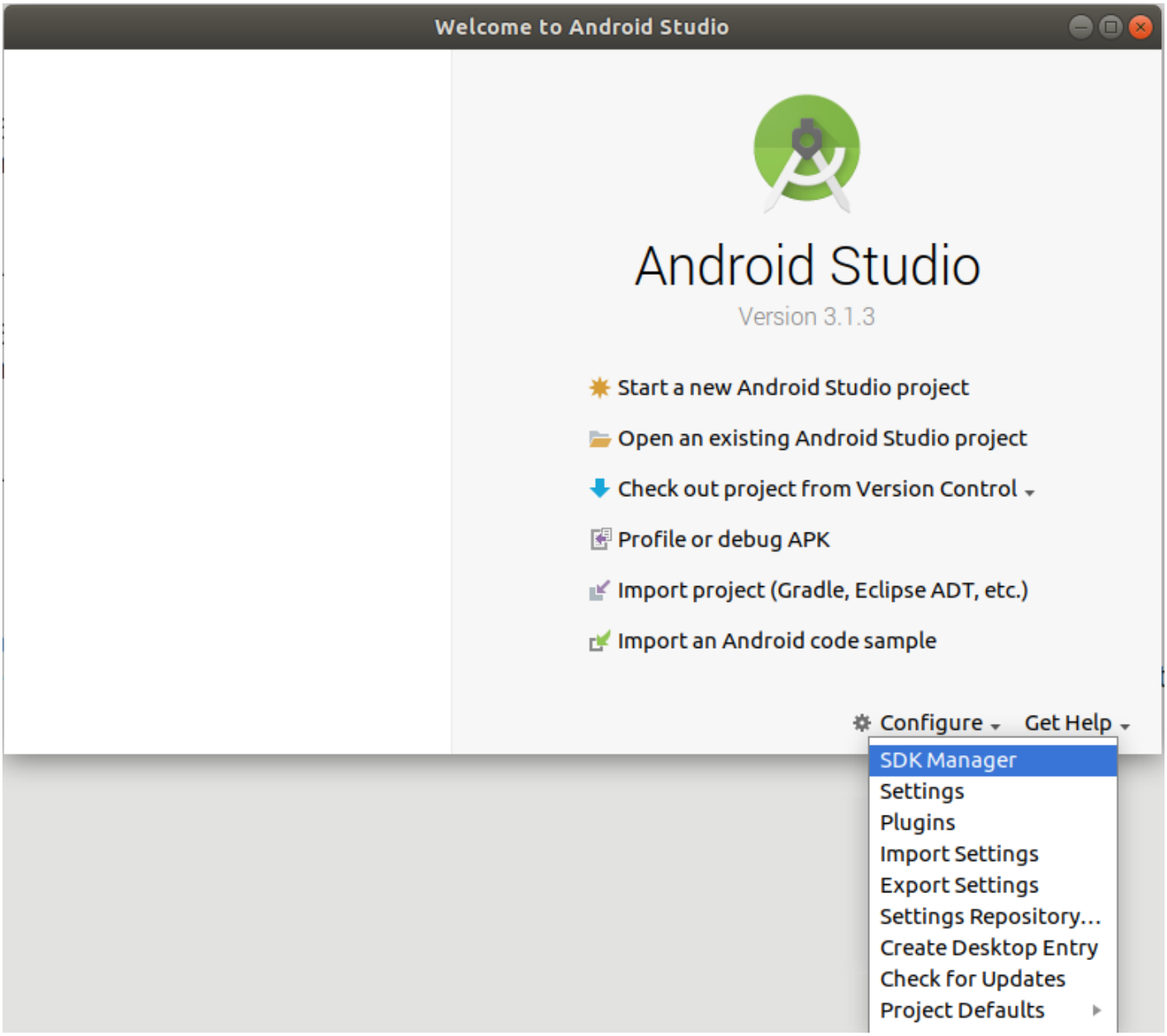
```
$ vim $HOME/.bashrc

export PATH=$PATH:/usr/local/android-studio/bin:/opt/Android/Sdk/platform-tools:/opt/Android/Sdk/tools:/opt/
    Android/Sdk/tools/bin

$ source $HOME/.bashrc

$ gksudo /usr/local/android-studio/bin/studio.sh
```

Le SDK Android est constitué de paquets modulaires qui peuvent être téléchargés séparément au moyen de l'outil Android **SDK Manager**. Cet outil est particulièrement pratique lorsqu'il est nécessaire de faire une mise à jour liée à une évolution du niveau d'API. Plusieurs niveaux d'API peuvent cohabiter.

On y accède en démarrant l'IDE Android Studio :

```
$ studio.sh
```

Ou en utilisant la commande :

```
$ sdkmanager
```

Remarque : la commande `andoid` est devenue obsolete et elle est remplacée par `sdkmanager` (25.2.3 et supérieur). Elle permet de visualiser, d'installer, de mettre à jour et de désinstaller des *packages* pour le SDK d'Android. On peut aussi le faire directement à partir d'Android Studio. La commande se trouve dans `$HOME/Android/Sdk/tools/bin`.

Installation d'Android NDK

Liens :

- NDK Downloads
- NDK Archives

On a besoin de la version 17c du NDK.

- Pour une installation mono utilisateur dans `$HOME/Android/Sdk` :

```
$ cd $HOME/Android/Sdk
$ wget https://dl.google.com/android/repository/android-ndk-r17c-linux-x86_64.zip
```



```
$ unzip android-ndk-r17c-linux-x86_64.zip
```

— Pour une installation multi utilisateur dans `/opt/Android/Sdk` :

```
$ wget https://dl.google.com/android/repository/android-ndk-r17c-linux-x86_64.zip
$ sudo mv android-ndk-r17c-linux-x86_64.zip /opt/Android/Sdk/
$ cd /opt/Android/Sdk/
$ sudo unzip android-ndk-r17c-linux-x86_64.zip
$ sudo rm -rf android-ndk-r17c-linux-x86_64.zip
```

Tests

On dispose d'outils en ligne de commande (CLI) : https://developer.android.com/studio/command-line/

— Android SDK Platform Tools : adb (dans ~/Android/Sdk/platform-tools/)

Passer le smartphone ou la tablette en mode développeur (Paramètres -> A propos du téléphone -> Numéro de build, appuyez plusieurs fois) puis activer le débugage USB (Paramètres -> Options pour les développeurs).

Brancher un terminal mobile Android via le port USB et vérifier qu'il est détecté par le système :

```
$ dmesg
...

$ lsusb
...
```

Vérifer ensuite qu'il est reconnu par l'environnement Android de votre PC :

```
$ adb devices
List of devices attached
9b4364c88b62f3f6    device
```

Remarque : Sous Linux, il sera peut-être nécessaire de créer un fichier de règles *udev* qui contient une configuration USB pour chaque type de périphérique réel. En tant que root, vous pouvez créer le fichier `/etc/udev/rules.d/51-android.rules` et y inscrire la ligne suivante (en précisant en hexadécimal votre *idVendor*) :

```
$ sudo vim /etc/udev/rules.d/51-android.rules
SUBSYSTEM=="usb", ATTR{idVendor}=="XXXX", MODE="0666"
```

— Android SDK Tools : sdkmanager (dans ~/Android/Sdk/tools/bin)

```
$ cd ~/Android/Sdk/tools/bin

$ ./sdkmanager --list
Installed packages:=====================] 100% Computing updates...
  Path                        | Version      | Description                    |
  -------                     | -------      | -------                        |
  build-tools;28.0.3          | 28.0.3       | Android SDK Build-Tools 28.0.3 |
  emulator                    | 28.0.23      | Android Emulator               |
  extras;android;m2repository | 47.0.0       | Android Support Repository     |
  extras;google;m2repository  | 58           | Google Repository              |
  ndk-bundle                  | 19.0.5232133 | NDK                            |
  patcher;v4                  | 1            | SDK Patch Applier v4           |
  platform-tools              | 28.0.1       | Android SDK Platform-Tools     |
  platforms;android-28        | 6            | Android SDK Platform 28        |
  sources;android-28          | 1            | Sources for Android 28         |
  tools                       | 26.1.1       | Android SDK Tools              |
...
```

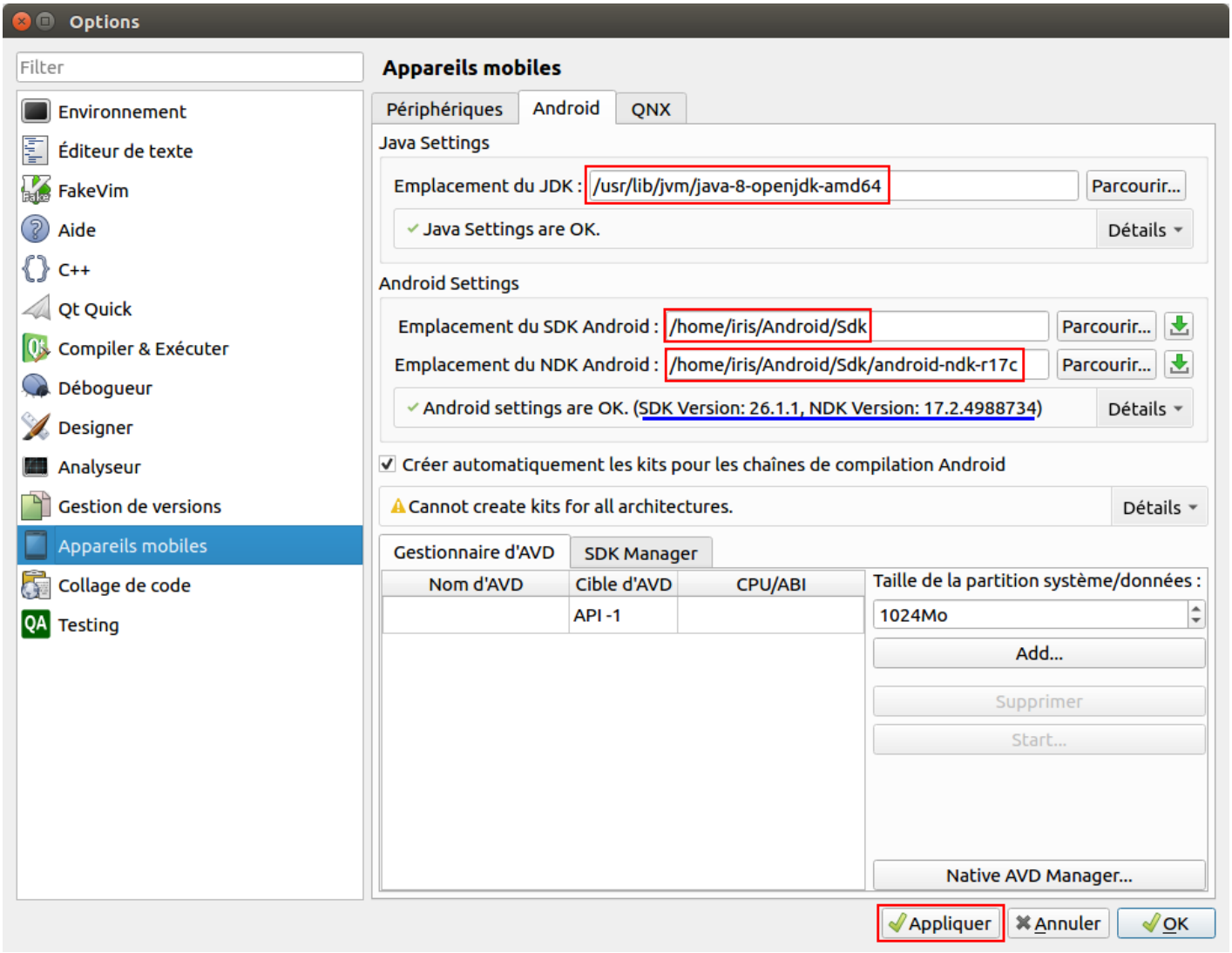
— Android SDK Tools : avdmanager (dans ~/Android/Sdk/tools/bin)

```
$ ./avdmanager list avd
Available Android Virtual Devices:
```


Configuration

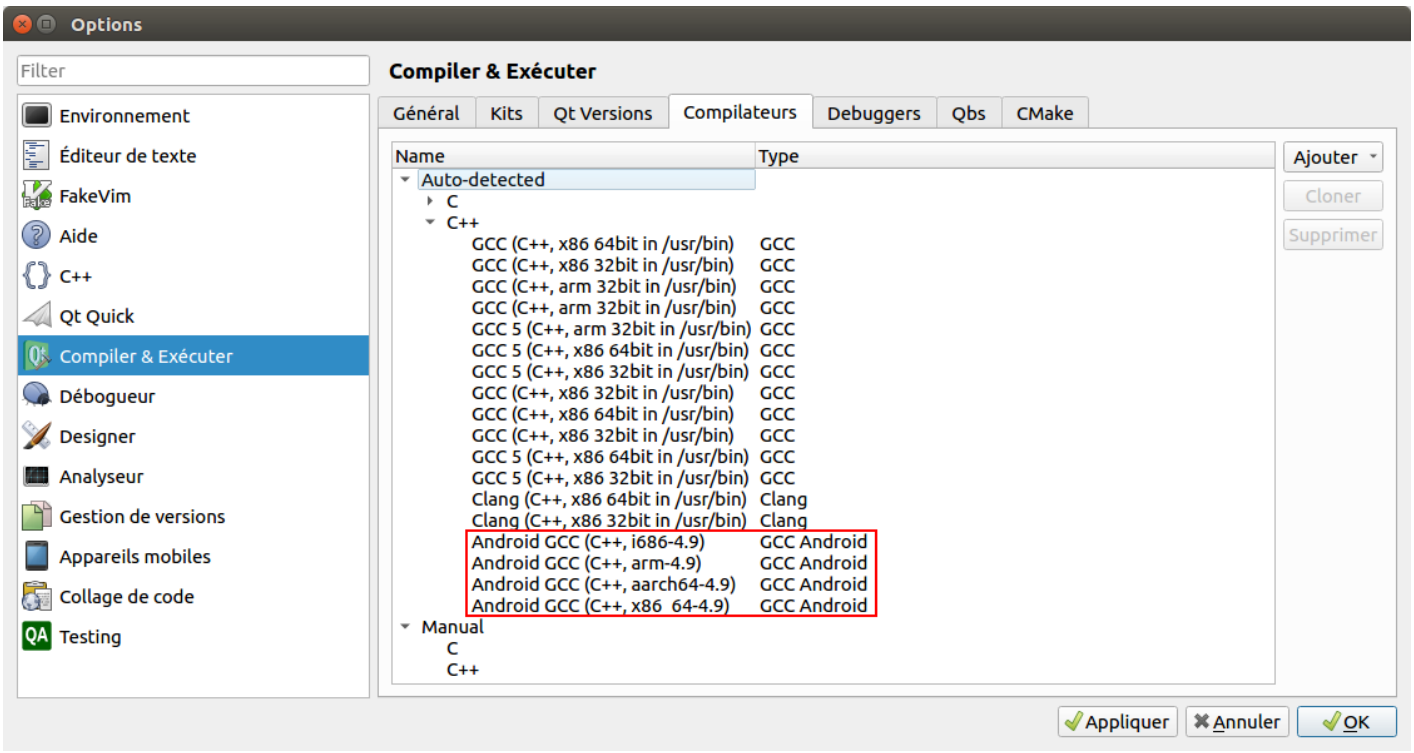
Dans Qt Creator, il faudra renseigner les ressources suivantes :

— le kit de développement Java SDK
— l'Android SDK
— l'Android NDK

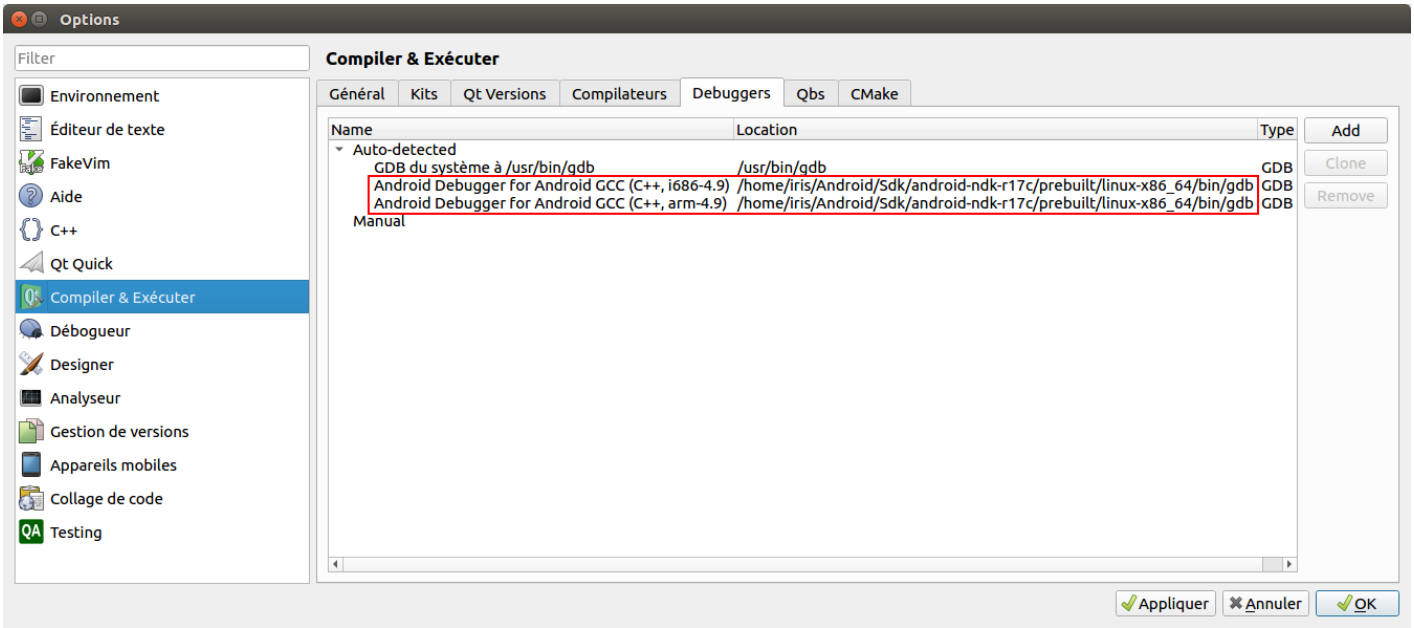


Cela va permettre de générer les kits de développement pour Android :

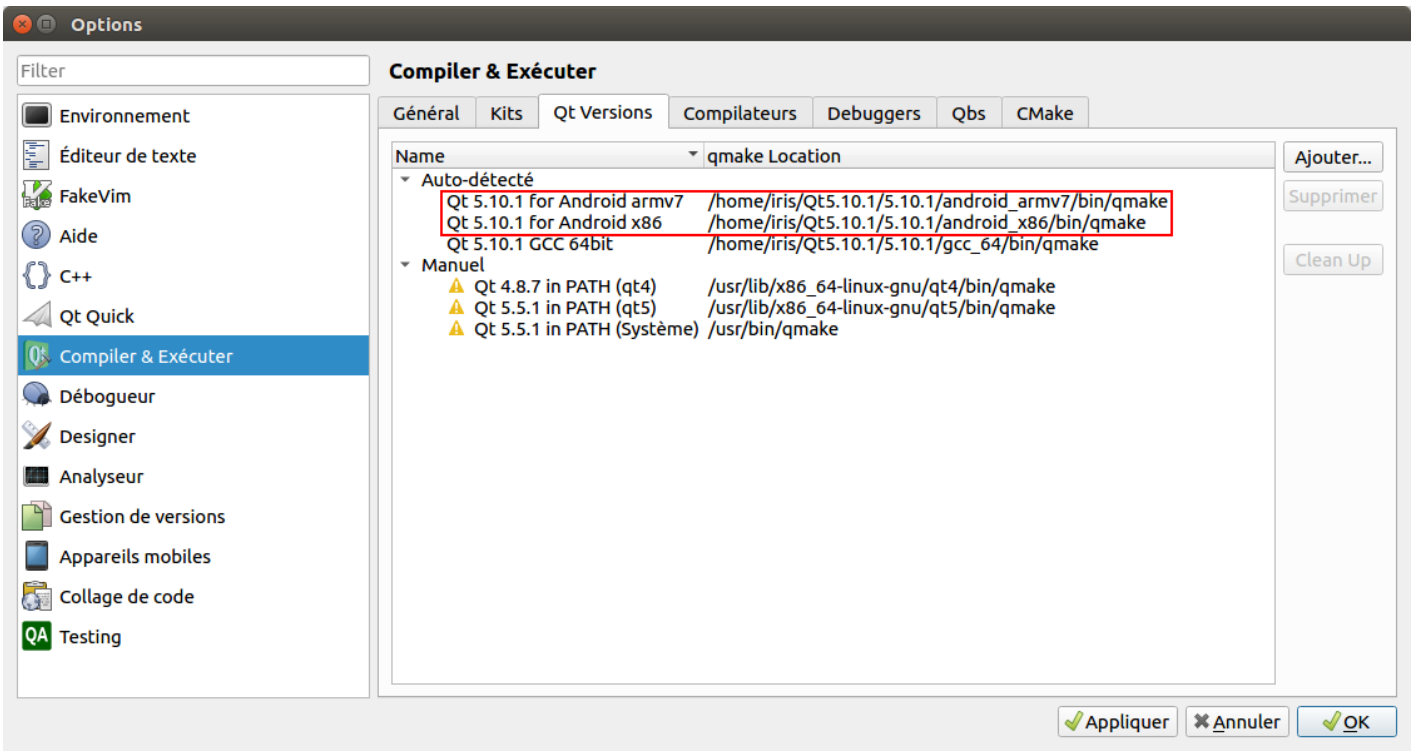
— le(s) compilateur(s) :



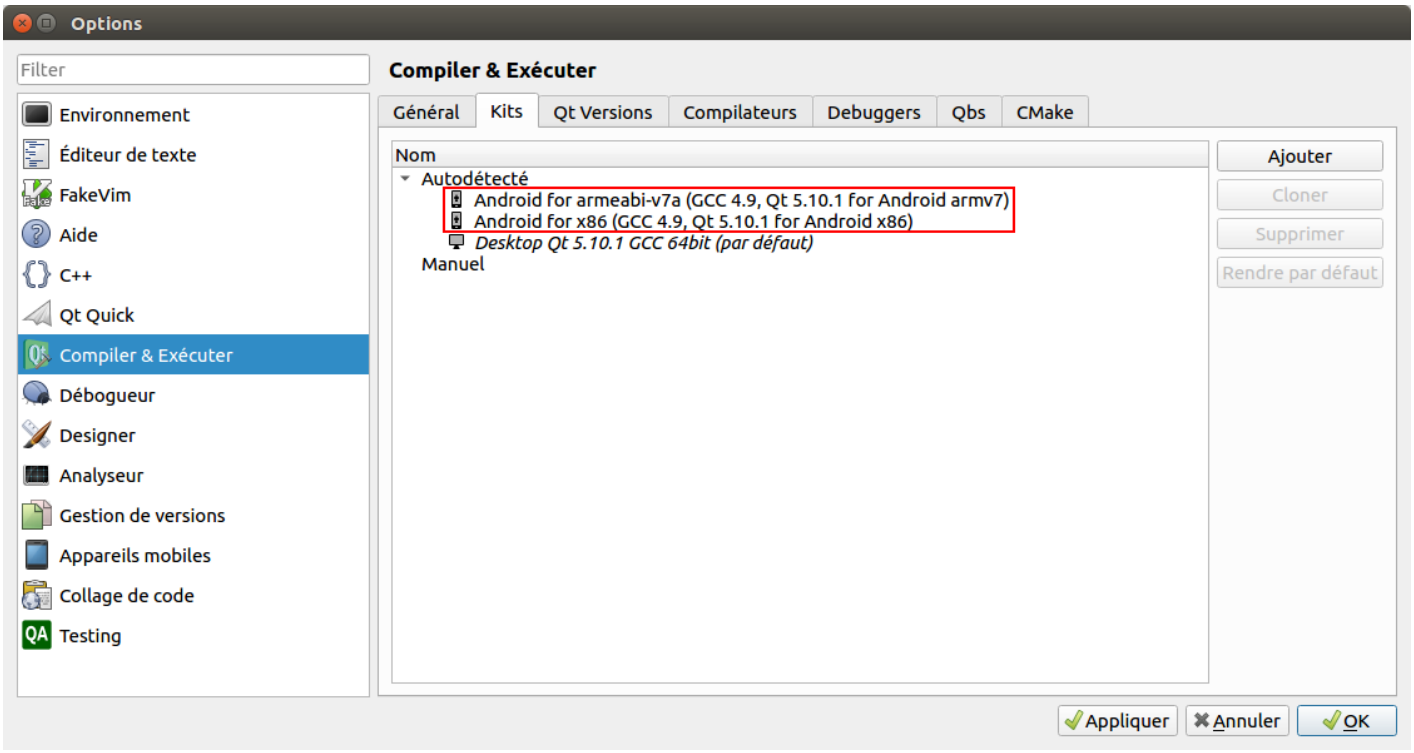
— le(s) débogueur(s) :



— le(s) chaîne(s) de fabrication Qt :

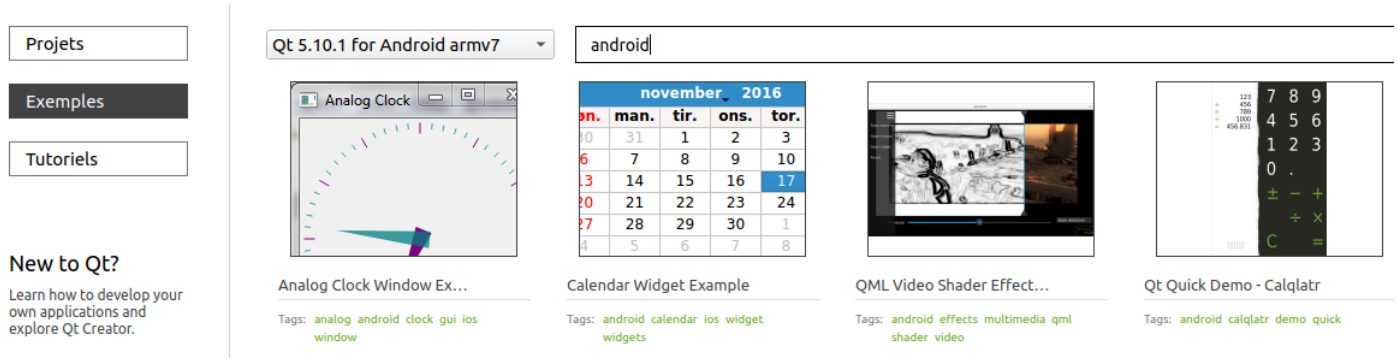


— les kits de développements :



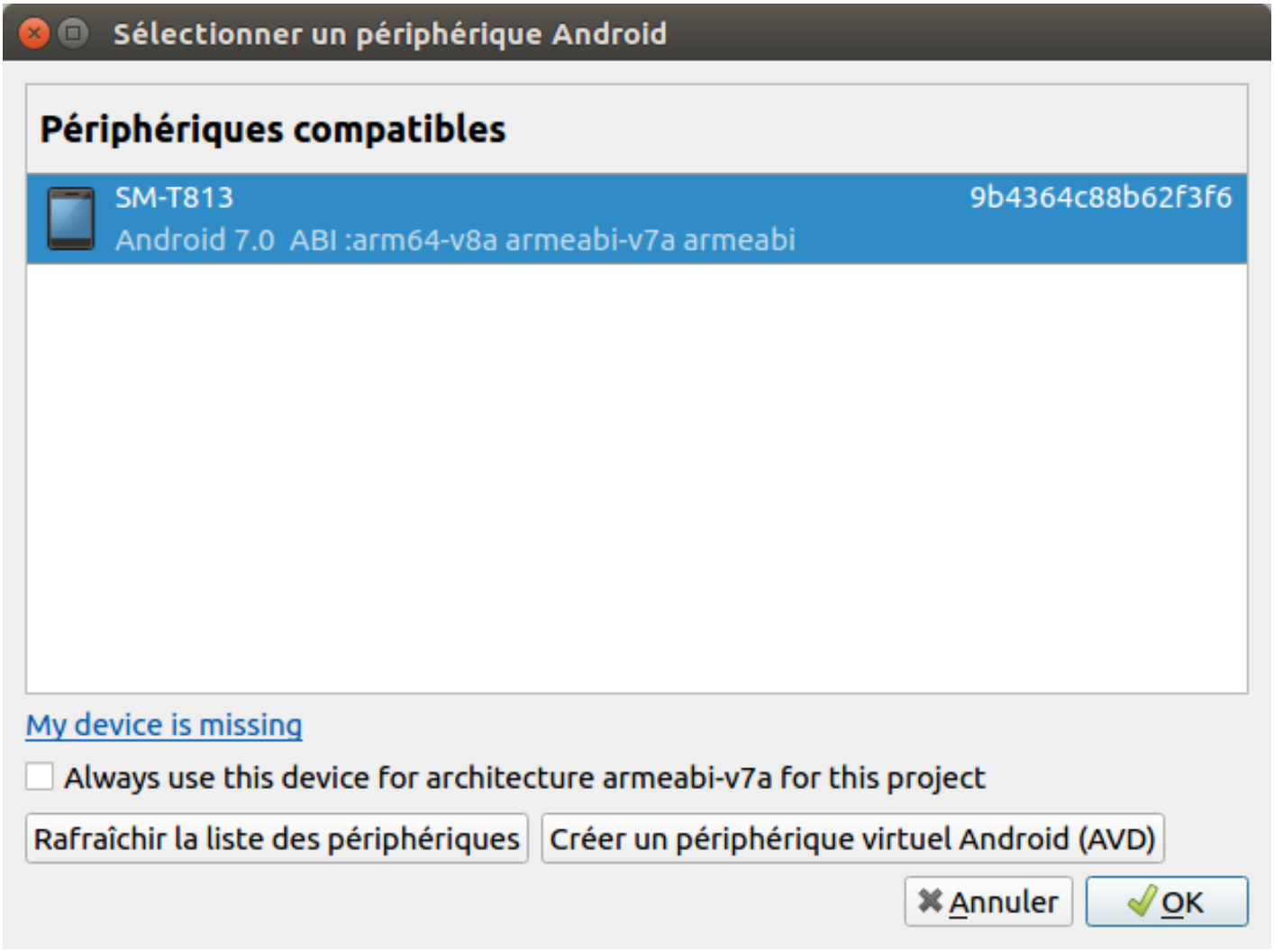
Test

Pour tester, il suffit de prendre un exemple fourni par Qt :



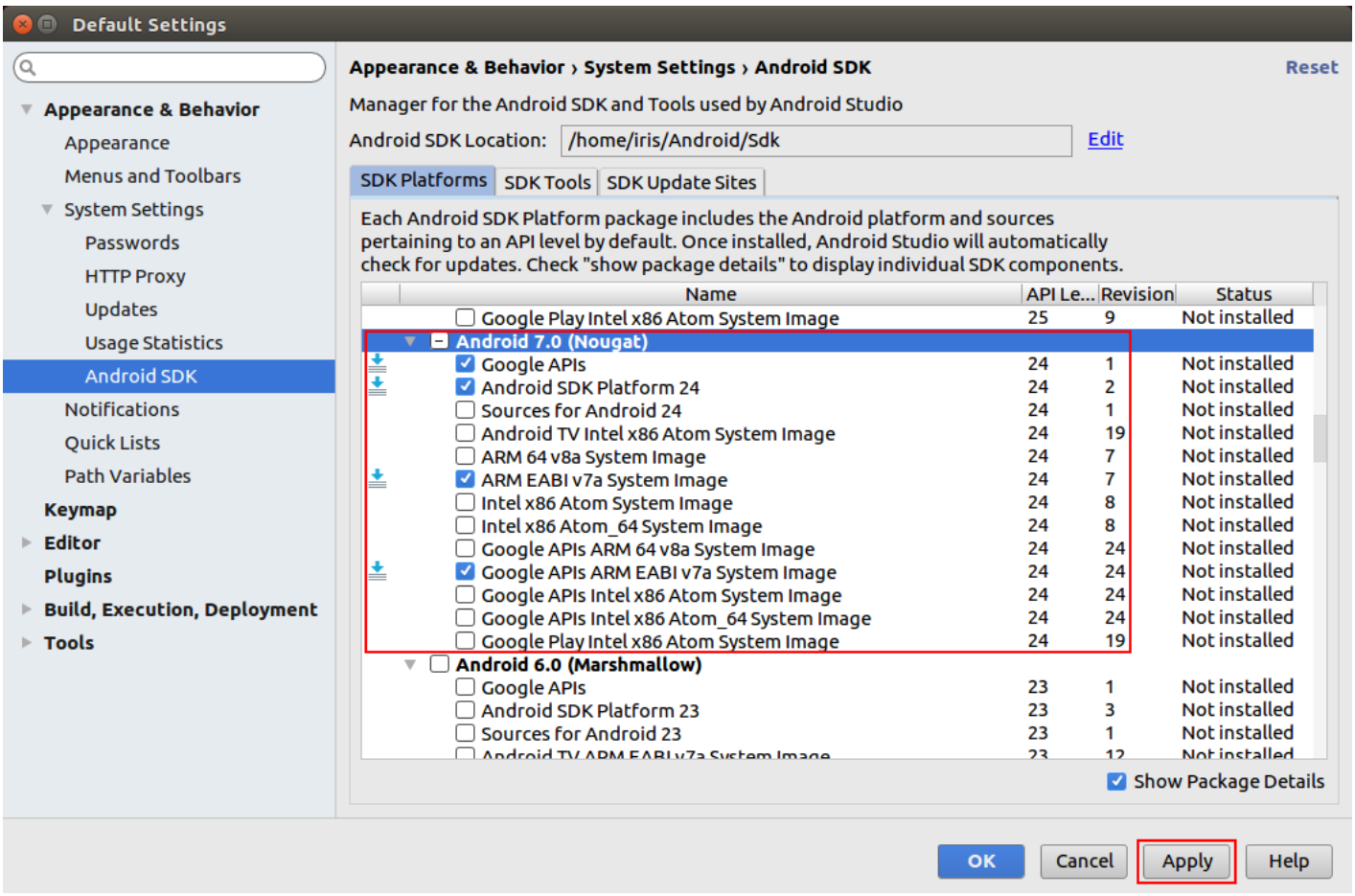
On sélectionne un kit de développement d'Android :

On branche le terminal mobile Android puis on fabrique l'application et on l'exécute sur le périphérique :



Emulateur

Il est aussi possible d'utiliser un émulateur qu'il faudra préalablement créer et configurer.
Pour cela, on téléchargera tout d'abord une image pour une API via le SDK Manager :



La gestion de l'émulateur peut se faire à partir d'Android Studio ou en utilisant la commande avdmanager :

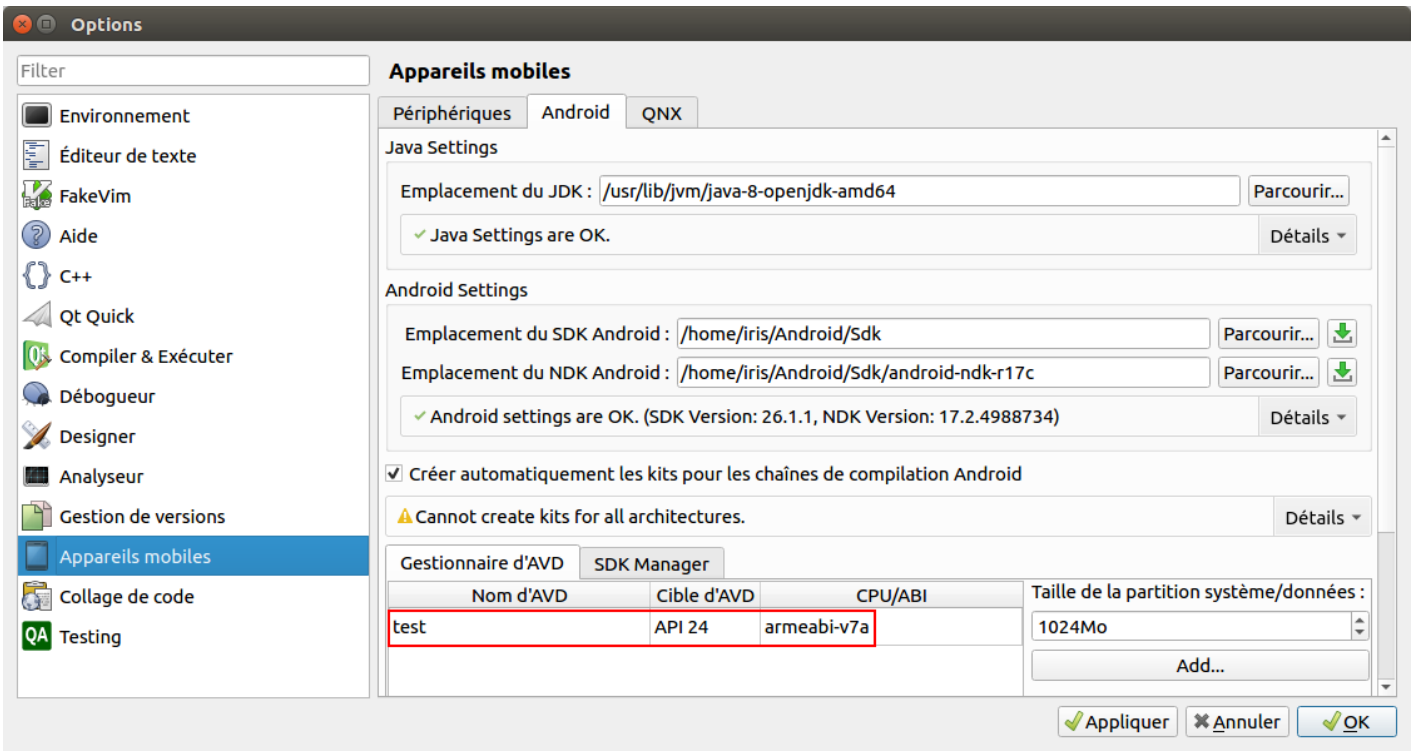
— version arm :

```
$ ./avdmanager create avd -n test -k "system-images;android-24;default;armeabi-v7a"

$ ./avdmanager list avd
Available Android Virtual Devices:
    Name: test
    Path: ~/.android/avd/test.avd
  Target:
          Based on: Android 7.0 (Nougat) Tag/ABI: default/armeabi-v7a
```

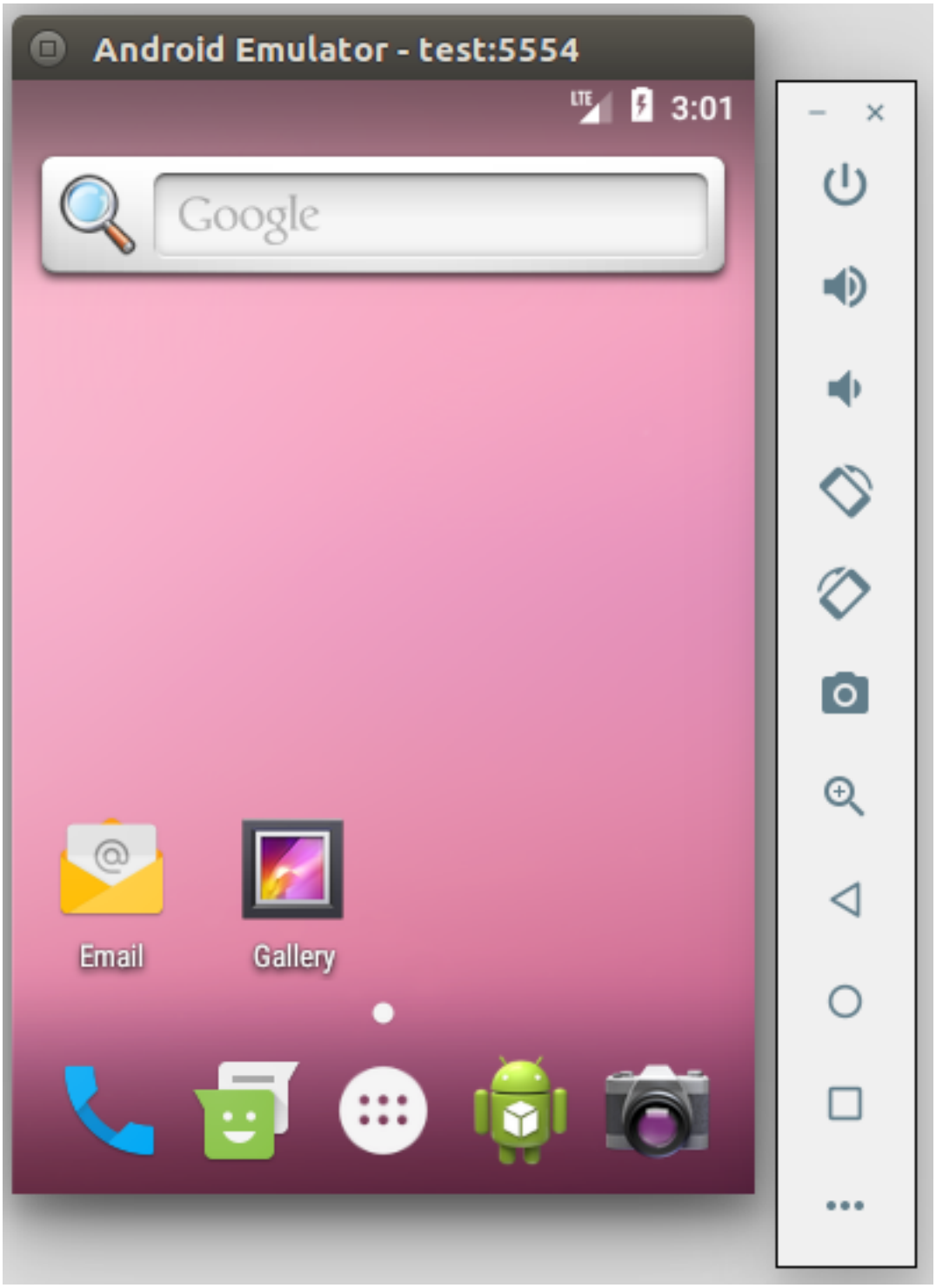
— version x86 (beaucoup plus rapide) :

```
$ ./avdmanager create avd -n test_x86 -k "system-images;android-24;default;x86_64"
```

L'émulateur est détecté et utilisable directement dans Qt :



Le choix de la cible se fera au moment de l'exécution de l'application, puis Qt démarrera l'émulateur et ensuite l'application :



Il est aussi possible de lancer l'émulateur en ligne de commande :

```
$ cd ~/Android/Sdk/emulator

$ $ ./emulator -avd test
```

Retour

