

Android et les commandes externes

1. adb

adb: Android Debug Bridge

C'est un outil de commandes pour communiquer avec un appareil

Il sert à :

- Installer des applications
- Déboguer ces applications
- Utiliser un Shell Unix pour exécuter des commandes
- Éviter d'utiliser Android Studio pour tout ce qui n'est pas programmation

Structure d'adb

Un client : il sert à envoyer des commandes. Cette partie est exécutée sur votre machine de développement.

Un service (le daemon `adbd`) : exécute les commandes envoyées sur l'appareil. Cette partie est un service s'exécutant en arrière-plan (background) de l'appareil.

Un serveur : gère la communication entre le client et l'appareil. Cette partie est un service qui s'exécute en arrière-plan (background) de votre machine de développement.

Utilisation d'adb sur vos appareils

Pour utiliser « Android Debug Bridge », il faut activer le mode « USB Debugging ».

Pour ce faire, il faut accéder aux options de développeur:

Settings -> About Phone -> puis taper 7 fois « Build number ».

À noter que certains appareils nécessitent de taper moins de 7 fois.

Sur l'émulateur, il faut faire :

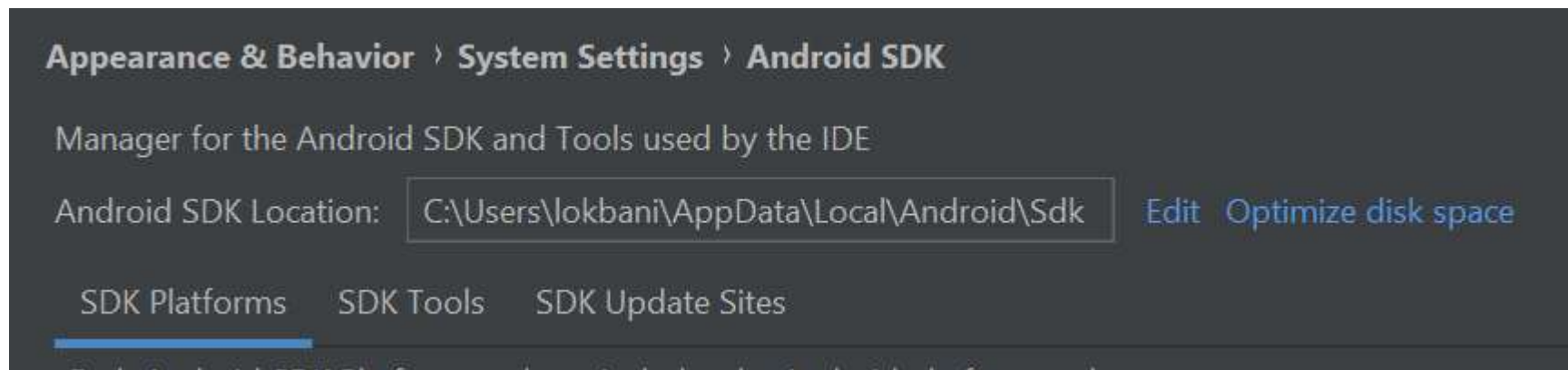
Settings -> « About emulated device » -> puis taper autant de fois sur « Build number ».

Exécuter un client adb

L'exécutable « adb » se trouve dans le dossier « ANDROID_SDK/platform-tools »

Pour trouver votre chemin « ANDROID_SDK », accéder à Android Studio:

Files → Settings → Appearance & Behavior → System Settings → Android SDK → Android SDK Location



Il faut ajouter le chemin vers le dossier « platform-tools » dans la variable d'environnement « path », si l'on veut éviter de spécifier le chemin complet vers la commande « adb ».

Pour lister tous les appareils détectés par le serveur, exécuter la commande « adb devices » :

```
C:\>adb devices
List of devices attached
emulator-5554    device
emulator-5556    device
```

L'option « -l » permet d'avoir plus de détails ...

```
C:\>adb devices -l
List of devices attached
emulator-5554    device product:sdk_gphone64_x86_64 model:sdk_gphone64_x86_64 device:emu64x transport_id:1
emulator-5556    device product:sdk_gphone64_x86_64 model:sdk_gphone64_x86_64 device:emu64x transport_id:2
```

Nom et le port de l'émulateur : emulator-5554

À chaque appareil est associé un identifiant, « transport_id:1 » est par exemple associé à l'émulateur « emulator-5554 ».

Les deux émulateurs sont maintenant actifs, d'où l'état de la connexion « device ».

Il y a 3 états : « device » « offline » « no device ».

Description de l'appareil (obtenu avec -l):

product:sdk_gphone64_x86_64

model:sdk_gphone64_x86_64

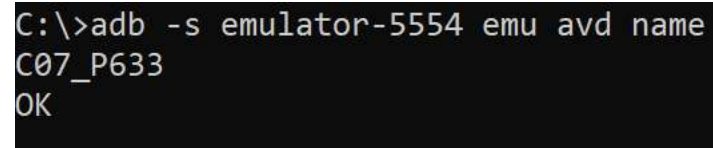
device:emu64x

transport_id:1

L'option « -s » dans la commande « adb -s » permet de sélectionner un émulateur en particulier. Cette option est nécessaire quand deux émulateurs ou plus sont actifs.

L'argument « emu » permet d'exécuter une commande sur l'émulateur. Par exemple la commande « adb -s emulator-5554 emu avd name » permet d'obtenir le nom associé à l'émulateur. L'émulateur est nommé au moment de sa création.

L'émulateur « emulator-5564 » porte le nom de « C07_P633 ».



```
C:\>adb -s emulator-5554 emu avd name
C07_P633
OK
```

Vous avez accès à la liste des commandes disponibles par :

« adb -s emulator-5554 emu avd /? »

Pour ouvrir un terminal sur l'émulateur, on exécute la commande « adb shell » :

```
C:\>adb -s emulator-5554 shell
emu64x:/ $ ls
acct      bugreports  data          etc            lost+found  odm_dkkm     product       sys            vendor
adb_keys  cache       data_mirror   init           metadata    oem          sdcard        system         vendor_dkkm
apex      config      debug_ramdisk init.environ.rc mnt         postinstall  second_stage_resources system_dkkm
bin       d           dev           linkerconfig   odm         proc         storage       system_ext
emu64x:/ $ id
uid=2000(shell) gid=2000(shell) groups=2000(shell),1004(input),1007(log),1011(adb),1015(sdcard_rw),1028(sdcard_r),1078(ext_data_rw),1079(ext_ob
b_rw),3001(net_bt_admin),3002(net_bt),3003(inet),3006(net_bw_stats),3009(readproc),3011(uhid),3012(readtracefs) context=u:r:shell:s0
emu64x:/ $
```

On peut exécuter une commande « shell » sans être obligé d'ouvrir un terminal sur l'émulateur. Dans cet exemple, nous avons affiché le contenu du répertoire à l'aide de la commande « ls » ainsi que les identifiants de l'utilisateur courant à l'aide de la commande « id ».

On peut copier un fichier dans l'émulateur à l'aide de la commande « push », comme suit :

`adb push local distant //` permet de copier de l'ordinateur vers l'appareil

Où « local » est le chemin sur votre machine et « distant » est le chemin sur l'appareil.

```
D:\>adb -s emulator-5554 shell ls /sdcard
Alarms
Android
Audiobooks
DCIM
Documents
Download
Movies
Music
Notifications
Pictures
Podcasts
Recordings
Ringtones
distant.txt

D:\>adb -s emulator-5554 push local.txt /sdcard/.
local.txt: 1 file pushed, 0 skipped. 0.1 MB/s (20 bytes in 0.000s)

D:\>adb -s emulator-5554 shell ls /sdcard
Alarms
Android
Audiobooks
DCIM
Documents
Download
Movies
Music
Notifications
Pictures
Podcasts
Recordings
Ringtones
distant.txt
local.txt
```

On peut aussi copier localement un fichier de l'émulateur à l'aide de la commande « pull », comme suit :

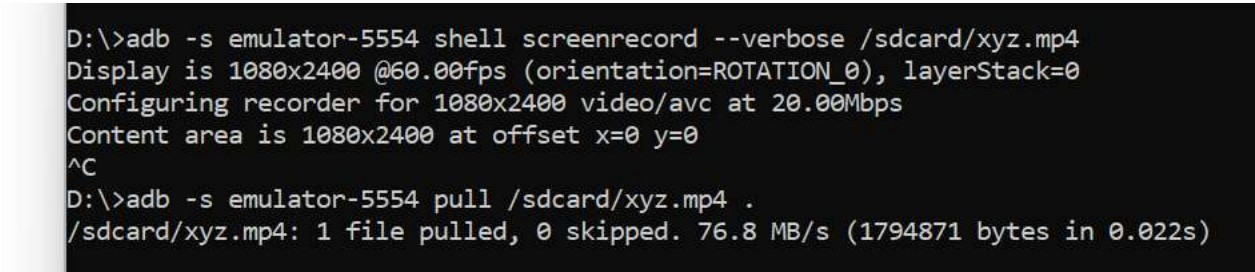
`adb pull distant local //` permet de copier de l'appareil vers l'ordinateur

Où « local » est le chemin sur votre machine et « distant » est le chemin sur l'appareil.

```
D:\>adb -s emulator-5554 pull /sdcard/distant.txt .  
/sdcard/distant.txt: 1 file pulled, 0 skipped. 0.0 MB/s (35 bytes in 0.002s)
```

On peut enregistrer une vidéo en ligne de commandes comme suit :

`adb shell screenrecord --verbose « chemin_distant »`



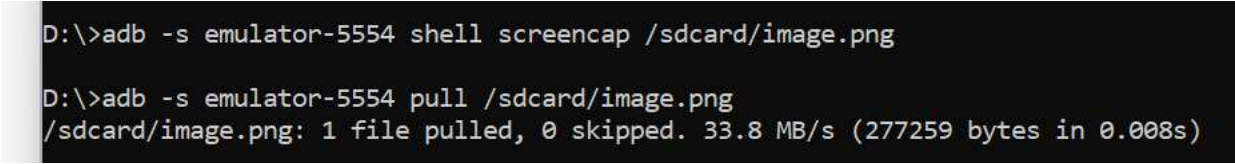
```
D:\>adb -s emulator-5554 shell screenrecord --verbose /sdcard/xyz.mp4
Display is 1080x2400 @60.00fps (orientation=ROTATION_0), layerStack=0
Configuring recorder for 1080x2400 video/avc at 20.00Mbps
Content area is 1080x2400 at offset x=0 y=0
^C
D:\>adb -s emulator-5554 pull /sdcard/xyz.mp4 .
/sdcard/xyz.mp4: 1 file pulled, 0 skipped. 76.8 MB/s (1794871 bytes in 0.022s)
```

L'ajout de la commande « --verbose » est optionnel. Elle nous permet d'avoir plus d'informations sur les paramètres de l'enregistrement. Nous utilisons les touches « CTRL + C » pour arrêter l'enregistrement.

On peut récupérer par la suite le fichier à l'aide de la commande « pull ».

On peut prendre une capture d'écran comme suit :

adb shell screencap « chemin_distant »

A terminal window with a black background and white text. It shows two commands being executed. The first command is 'D:\>adb -s emulator-5554 shell screencap /sdcard/image.png'. The second command is 'D:\>adb -s emulator-5554 pull /sdcard/image.png', followed by its output: '/sdcard/image.png: 1 file pulled, 0 skipped. 33.8 MB/s (277259 bytes in 0.008s)'.

```
D:\>adb -s emulator-5554 shell screencap /sdcard/image.png
D:\>adb -s emulator-5554 pull /sdcard/image.png
/sdcard/image.png: 1 file pulled, 0 skipped. 33.8 MB/s (277259 bytes in 0.008s)
```

L'image a l'extension « png ».

On peut récupérer par la suite le fichier à l'aide de la commande « pull ».

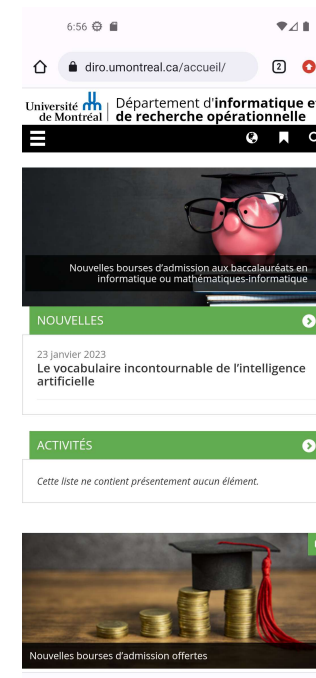
am: activity manager

On démarre un intent sur la ligne de commande à l'aide du gestionnaire d'activités « am » pour l'émulateur « emulator-5554 ».

```
adb -s emulator-5554 shell am start -a android.intent.action.VIEW -d "https://diro.umontreal.ca"
```

```
D:\>adb -s emulator-5554 shell am start -a android.intent.action.VIEW -d "https://diro.umontreal.ca"
Starting: Intent { act=android.intent.action.VIEW dat=https://diro.umontreal.ca/... }
```

Regarder maintenant le résultat sur l'écran de l'émulateur.



pm: package manager

On installe et l'on désinstalle une application à l'aide du gestionnaire de paquetage « pm ».

Nous allons installer le navigateur « FireFox » dont sa version Android est disponible à partir de ce site web : <https://github.com/mozilla-mobile/fenix/releases>

Nous allons télécharger la version « fenix-zzz-x86_64.apk » où « zzz » est le numéro de la dernière version disponible.

La commande directe ne fonctionne pas par convention ...

<https://issuetracker.google.com/issues/80270303#comment11>

Si la dernière version disponible est « 109.2.0 » :

```
adb -s emulator-5554 shell pm install fenix-109.2.0-x86_64.apk
```

```
D:\>adb -s emulator-5554 shell pm install fenix-109.2.0-x86_64.apk
Error: Unable to open file: fenix-109.2.0-x86_64.apk
Consider using a file under /data/local/tmp/
Error: Can't open file: fenix-109.2.0-x86_64.apk
```

Comme il est indiqué dans le message, il faut d'abord copier le fichier localement sur l'émulateur pour avoir la possibilité de l'installer par la suite.

```
D:\>adb -s emulator-5554 push fenix-109.2.0-x86_64.apk /data/local/tmp
fenix-109.2.0-x86_64.apk: 1 file pushed, 0 skipped. 123.3 MB/s (89681747 bytes in 0.694s)

D:\>adb -s emulator-5554 shell pm install /data/local/tmp/fenix-109.2.0-x86_64.apk
Success
```

Vérifier que le navigateur est bien installé sur votre appareil ...

L'autre option, encore plus simple :

```
adb -s emulator-5554 install fenix-109.2.0-x86_64.apk
```

```
D:\>adb -s emulator-5554 install fenix-109.2.0-x86_64.apk
Performing Streamed Install
Success
```

Pour désinstaller l'application, commencez par chercher le nom du paquetage à l'aide de la commande :

```
adb -s emulator-5554 shell pm list packages
```

Le paquetage Firefox a comme nom « org.mozilla.firefox ».

Pour retirer l'application :

```
adb -s emulator-5554 shell pm uninstall org.mozilla.firefox
```

ou bien

```
adb -s emulator-5554 uninstall org.mozilla.firefox
```

À noter qu'il est possible de préserver les données de l'application au moment de la désinstallation. Nous vous invitons à examiner la documentation à ce sujet.

```
D:\>adb -s emulator-5554 shell pm uninstall org.mozilla.firefox  
Success
```

```
D:\>adb -s emulator-5554 uninstall org.mozilla.firefox  
Success
```

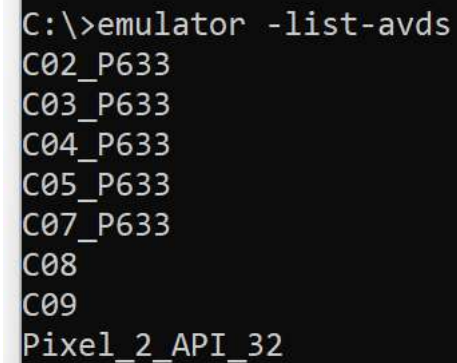
2. Émulateur

Il est possible de lancer un émulateur déjà créé à partir de la ligne de commandes, sans être obligé de passer par l'interface d'Android Studio.

Pour ce faire accéder au dossier `ANDROID_SDK/emulator` (ou bien ajouter le chemin dans la variable système « path »).

Obtenez la liste des émulateurs disponibles, par leur nom :

`emulator -list-avds`



```
C:\>emulator -list-avds
C02_P633
C03_P633
C04_P633
C05_P633
C07_P633
C08
C09
Pixel_2_API_32
```

Lancer l'émulateur désiré :

emulator -avd « NOM_AVD »

```
C:\>emulator -avd C05_P633
INFO      | Android emulator version 32.1.11.0 (build_id 9536276) (CL:N/A)
INFO      | Found systemPath C:\Users\lokbani\AppData\Local\Android\Sdk\system-images\android-33\google_apis\x86_64\
INFO      | Storing crashdata in: C:\Users\lokbani\AppData\Local\Temp\AndroidEmulator\emu-crash.db, detection is enabled
INFO      | Duplicate loglines will be removed, if you wish to see each individual line launch with the -log-nofilter flag
.
INFO      | IPv4 server found: 192.168.132.1
INFO      | added library vulkan-1.dll
INFO      | Sending adb public key [QAAAAHmdZ8E3w5pj+rwG5C0c+uRqba5+Xm87qg3YOrdYEabwkNnSkgyu/tTZUBLwIMSim6NI5u0zrsEu9g71cq
u2ZKWdW062RujV1eKHa1xZUNdoLsDpF3x2AFmWUliwFDP16u38bo+Qpx39F3T/yDPsvLDtpARuNqLRp73v9PmOJhJ9A0ct5U2eg4RITAQMIS62kyuD0ZsxH
DcU0TzW1B0dB+j04UwPYUBvKV+5Z99Uzc0g3Ti0cV7bLXR5AJU1wHUmQ3jM+3pW/CiSa03x+1Yo2/6UQJcz1DipP9gVhKxg7pv5hW+0LW0vcBB1NPUVZIFub
BVtNzd+A5C59wIPo4Z9OYzD03CH8rFtlcg6oThQd97T5QF77yW2TLArU6kmK9EyNKUQMPeMdnKwOcUZlwa7fBMgUXLtrs8G6h9KkX0JaODCKRgBkRSWhrkae
+Eey3K4XVgjFrHPo1sajwJcgS+EWHypu7VwJquLOv/ueR217V0JmJ+6gHRMmSLmKC3JKE4K2d40gNq0jV3JeT/XGppDE9jo/Y1+3x/qajVwLiST7zyMmP9ru
MT7bscc4B9Zjd0z9TNFRtMs/628HY27jEDNihlt79uECBYUeHSSmZV/L5SNUnhSxEc6GFcFFVkpA8JhiMbvg1U1jjq3wR1C8qtevHRbZ8DjpLAlYDIvHiL/m
LImbNVNQEAQA= @unknown]
WHPX on Windows 10.0.22621 detected.
Windows Hypervisor Platform accelerator is operational
WARNING   | *** No gRPC protection active, consider launching with the -grpc-use-jwt flag.***
INFO      | Started gRPC server at 127.0.0.1:8556, security: Local, auth: none
INFO      | Advertising in: C:\Users\lokbani\AppData\Local\Temp\avd\running\pid_22564.ini
INFO      | Connecting to bluetooth mesh: localhost:8554
INFO      | Setting display: 0 configuration to: 1080x2400, dpi: 420x420
```

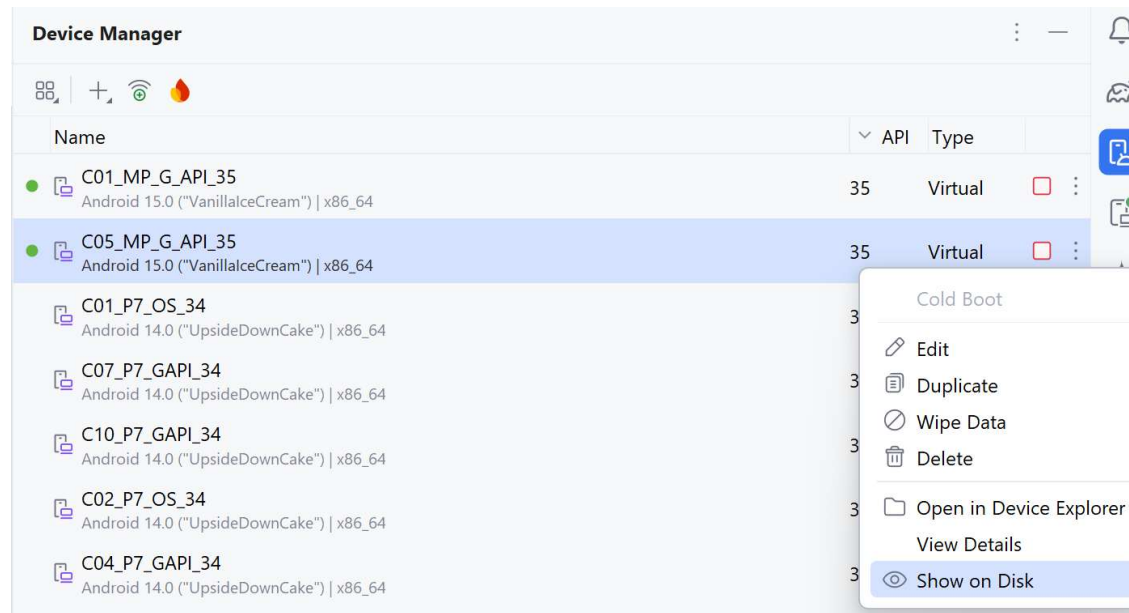
Cette option de lancement permet d'obtenir plus d'informations sur l'état de l'émulateur. On constate par exemple que l'accélération matérielle est activée et opérationnelle.

On peut avoir aussi des suggestions ...

```
INFO | Wait for emulator (pid 22564) 20 seconds to shutdown gracefully before kill;you can set environment variable A  
NDROID_EMULATOR_WAIT_TIME_BEFORE_KILL(in seconds) to change the default value (20 seconds)
```

Si l'écran est affiché en dehors du cadre de la fenêtre, commencer par fermer l'émulateur. La combinaison des touches (CTRL & c) est une façon abrupte de le faire.

Ouvrir le répertoire où se trouve la configuration de l'émulateur en cliquant sur « Show on disk » :



Dans le répertoire affiché, cliquer sur le fichier « emulator-user.ini » et modifier ces deux lignes :

```
window.x = 0  
window.y = 0
```

3. avdmanager

La commande « avdmanager » permet de créer et de lister les émulateurs disponibles. Pour utiliser cette commande, il faut :

- Ajouter le chemin vers « ANDROID_SDK/cmdline-tools/latest/bin ».
- Configurer la variable d'environnement « JAVA_HOME » vers le chemin qui contient le répertoire « jbr » dans l'installation d'Android Studio. Par exemple :

« C:\Program Files\Android\Android Studio\jbr »

Attention, il s'agit de « jbr » à ne pas confondre avec « jre ».

Si on veut lister tous les émulateurs disponibles (dans cet exemple 2 émulateurs) :

`avdmanager list avd`

```
C:\>avdmanager list avd
Available Android Virtual Devices:
  Name: C02_P633
  Device: pixel_6 (Google)
  Path: C:\Users\lokbani\.android\avd\C02_P633.avd
  Target: Google APIs (Google Inc.)
        Based on: Android 13.0 (Tiramisu) Tag/ABI: google_apis/x86_64
  Skin: pixel_6
  Sdcard: 512M
-----
  Name: C03_P633
  Device: pixel_6 (Google)
  Path: C:\Users\lokbani\.android\avd\C03_P633.avd
  Target: Google APIs (Google Inc.)
        Based on: Android 13.0 (Tiramisu) Tag/ABI: google_apis/x86_64
  Skin: pixel_6
  Sdcard: 512M
```

Nous avons le nom de l'émulateur, son emplacement sur le disque local, la version du téléphone, de l'API etc.

4. Gestion des SDK

La gestion se fait à l'aide de la commande « `sdkmanager` ». De la même manière que « `avdmanager` », il faut que le chemin vers la commande ainsi que la variable d'environnement « `JAVA_HOME` » soient configurés.

Cette commande permet d'installer, lister, mettre à jour, désinstaller des paquets.

`sdkmanager --update` permet de mettre à jour tous les paquets à la dernière version.

`sdkmanager --list` permet de lister tous les paquets disponibles et déjà installés.

`sdkmanager --list_installed` permet de lister tous les paquets déjà installés.

<https://developer.android.com/studio/command-line/sdkmanager.html>

5. Log et tâches

Récupération des logs : `adb logcat`

Afficher les tâches disponibles : `gradlew tasks`

Nous allons avoir l'occasion d'étudier ces deux commandes prochainement.

Mot de la fin

Penser à ajouter les différents chemins vers les émulateurs, les commandes (adb, etc.), SDK, dans la variable système « path » de votre ordinateur, comme nous l'avons mentionné dans le chapitre 1 du cours. Ceci va permettre un accès rapide à ces différents outils.

Il est possible d'utiliser Android Studio que pour programmer! On peut se servir des commandes disponibles pour réaliser les autres tâches connexes.

Références

Pour une liste complète des commandes « adb »:

<https://developer.android.com/tools/adb>

<https://developer.android.com/tools/sdkmanager>

Une version préliminaire de ce document a été rédigée par François Corneau-Tremblay (démon H18).