

République Tunisienne
Ministère de
l'Enseignement
Supérieur
et de la Recherche
Scientifique



الجمهورية التونسية
وزارة التعليم العالي
والبحث العلمي

Compte rendu TP ADB



Auteur

Karim Dhafer
Ahmed Khammesi

Matière

Développement Mobile sur Android

Classe

L3-DSI3

Année universitaire

2024-2025

Sommaire

Introduction	1
Objectifs	1
1. Aperçu sur ADB	1
2. Configuration de l'ADB	1
2.1 Vérification de l'ADB sur le système	1
2.2 Connection de l'ADB avec les appareils	2
2.2.1 Appareil connecté	3
2.2.2 Aucune appareil connecté	3
3. La pratique des commandes adb	3
3.1 Transfert d'image du PC à l'appareil	3
3.1.1 Vérification du transfert d'image	4
3.2 Création des fichiers sous le terminal mobile et transfert vers PC	4
3.2.1 Connection au terminal de l'appareil	4
3.2.2 Exécution des commandes Shell et création des fichiers	5
3.2.3 Récupération des fichiers sur le PC	5
3.3 Récupération des contacts du mobile au PC	6
3.3.1 Connection au terminal de l'appareil	6
3.3.2 Recherche du dossier des contacts sous l'appareil	6
3.3.3 Récupération des contacts en local	6
3.4 Installation manuelle d'un APK	7
3.4.1 Vérification préalable de l'inexistence d'APK	7
3.4.2 Positionnement sur le dossier et création d'archive	7
3.4.3 Connection au shell mobile	7
3.4.4 Vérification du signature	7
3.4.5 Installation du apk	8
3.4.6 Vérification de l'installation du apk	8
3.4.7 Lancement de l'activité	9
3.4.8 Lancement exécutive de l'activité et activation du profilage	9
3.5 Désinstallation manuelle d'un APK	10
3.5.1 Connection au shell mobile	10
3.5.2 Listement des packages mobiles	10
3.5.3 Récupération de la liste dans un fichier en local	10
3.5.4 Désintallation d'un package	11
Conclusion	11

Liste des figures

1	Vérification de l'adb sous le système	2
2	Variables d'environnement	2
3	Liste des appareils Connectés	3
4	Copie de l'image sur le PC vers l'appareil Android	3
5	Vérification de l'existence et la taille de l'image sous Android	4
6	Taille de l'image sur PC	4
7	Affectation des permissions root au adb	4
8	Connection au terminal de l'émulateur	4
9	Exécution des commandes <code>ps -ef & ls -l</code>	5
10	Vérification de l'existence des fichiers	5
11	Récupération des fichiers textes du smartphone au PC	5
12	Vérification de l'existence des fichiers en local	5
13	Contenu du fichier <code>ls_output.txt</code> en local	6
14	Vérification de l'existence de la base de données des contacts	6
15	Récupération de la base des contacts sur le PC	6
16	Vérification de l'inexistence d'apk	7
17	Navigation et compression de l'application	7
18	Vérification du signature	8
19	Installation du apk dans le mobile	8
20	Vérification de l'installation du apk	8
21	Vérification du fonctionnement du apk	8
22	Lancement de l'activité	9
23	Profilage	9
24	Exécution 20 fois	9
25	Copie du fichier de profilage vers PC	9
26	Liste des packages	10
27	Création du fichier contenant les packages	10
28	Copie du fichier des packages sur PC	10
29	Désinstallation du package	11

Introduction

Ce rapport se concentre sur l'outil ADB dans le cadre du développement sur la plateforme Android, en spécifiant la manière de manipuler un appareil Android à travers l'utilisation de cet outil.

Objectifs

- Avoir un aperçu général sur l'outil ADB.
- Configurer l'ADB sur le PC.
- Pratiquer quelques commandes `adb`.

1. Aperçu sur ADB

L'Android Debug Bridge (ADB) est un outil essentiel pour le développement et le débogage d'applications sur les appareils Android. Il permet aux développeurs et aux utilisateurs avancés de communiquer avec leurs appareils Android via une interface en ligne de commande.

Voici quelques fonctionnalités principales d'ADB :

- **Communication avec d'autres appareils :** ADB sert de pont entre un appareil Android et un ordinateur, facilitant l'exécution de commandes à distance.
- **Installation et débogage d'applications :** Les utilisateurs peuvent installer des applications, déboguer des programmes, et accéder à un shell Unix pour exécuter diverses commandes sur l'appareil.
- **Gestion des fichiers :** ADB permet de copier des fichiers entre l'ordinateur et l'appareil, ainsi que de sauvegarder des données.
- **Exécution des commandes systèmes sur l'appareil :** Les commandes ADB peuvent exécuter des différentes commandes systèmes. Par exemple, redémarrer l'appareil en mode recovery ou bootloader ce qui est utile pour des opérations avancées comme le flashage de ROMs, effectuer une réinitialisation d'usine, modifier des paramètres système, etc ...

2. Configuration de l'ADB

Pour configurer et tester la connexion d'un émulateur Android avec ADB (Android Debug Bridge), il est essentiel de suivre quelques étapes clés. Voici un aperçu sur les procédures à suivre.

2.1 Vérification de l'ADB sur le système

Si nous avons installé Android Studio avec le SDK sur le PC, l'ADB sera inclus dans ce process, d'où il faut juste tester si cet outil est configuré sous les variables d'environnement ou pas. Pour cela on lance la commande `adb version` dans le cmd.

```
C:\adb>adb version
Android Debug Bridge version 1.0.41
Version 35.0.2-12147458
Installed as C:\Users\kalle\AppData\Local\Android\Sdk\platform-tools\adb.exe
Running on Windows 10.0.22631
```

FIG. 1 : Vérification de l'adb sous le système

Si le résultat décrit que cette commande n'est pas reconnu, il faut ajouter ce path dans les variables d'environnement ;

C : \Users\<user_name> \AppData \Local \Android \Sdk \platform- tools

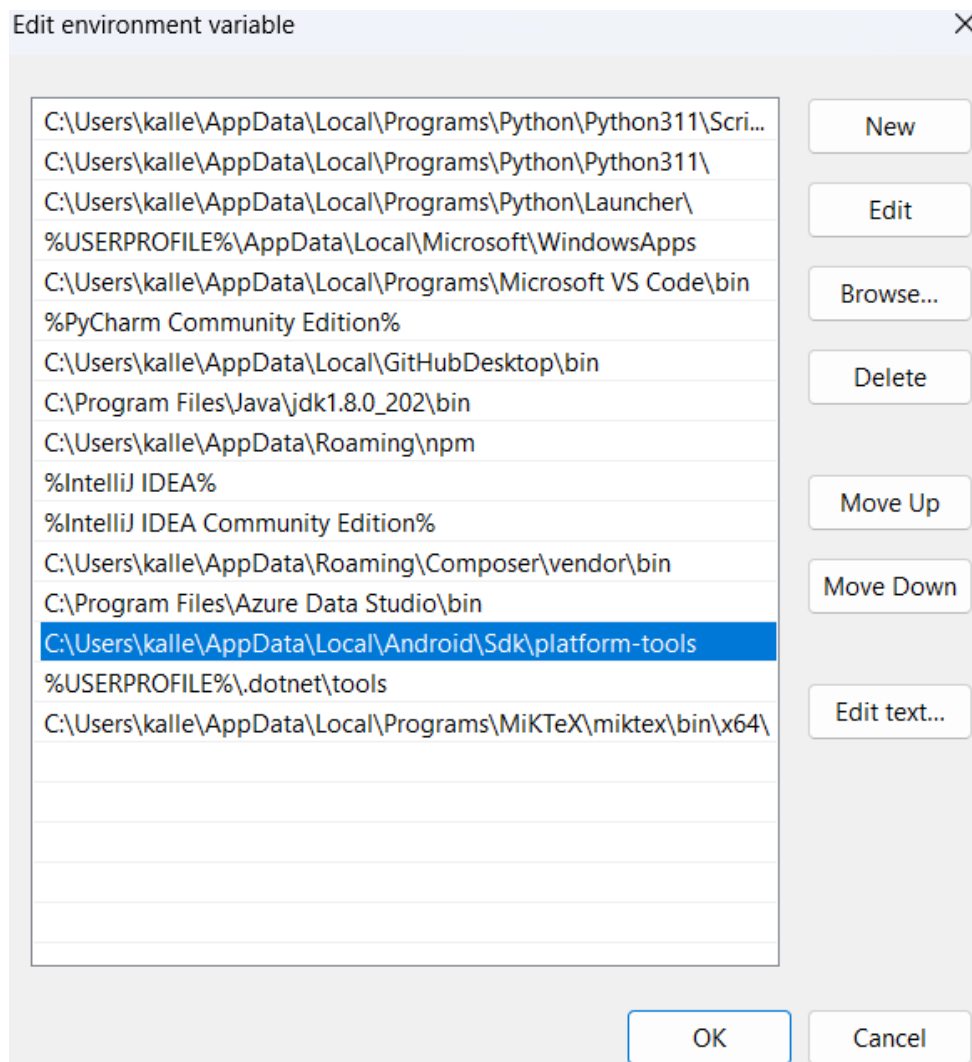


FIG. 2 : Variables d'environnement

2.2 Connection de l'ADB avec les appareils

Pour se connecter avec les appareils avec succès, nous allons parcourir 2 cas possibles ; appareil connecté directement ou connection à l'appareil avec des commandes adb.

2.2.1 Appareil connecté

Après avoir lancé un émulateur Android ou Si on a un smartphone réel, nous vérifions s'il est correctement connecté à l'ADB en utilisant la commande `adb devices` dans le terminal :

```
PS C:\Users\kalle> adb devices
List of devices attached
emulator-5554    device
```

FIG. 3 : Liste des appareils Connectés

Comme le nous voyons ci-dessus, il y a un appareil connecté avec succès qui est évidemment l'émulateur lancé.

Si on connecte avec un smartphone réel, le résultat sera une chaîne de caractères semblable à `1e778e25` qui est l'ID du smartphone.

2.2.2 Aucune appareil connecté

Si la commande `adb devices` n'a rien affiché et nous voulons connecter à notre smartphone via WiFi, il faut Suivre les étapes suivantes :

1. Ouvrir un port pour connecter au téléphone : `adb tcpip 5555`.
2. Connecter le téléphone à ce port là : `adb connect <device-ip>:5555` , avec <device-ip> est l'adresse IP de l'appareil.
3. Tester la connection avec `adb devices`.

3. La pratique des commandes adb

Dans cette partie, nous allons explorer la pratique des commandes `adb` pour un meilleur apprentissage de l'outil à travers les questions du TP ADB.

3.1 Transfert d'image du PC à l'appareil

Pour copier un fichier ou un dossier de l'ordinateur à l'appareil, nous utilisons la commande `push <local> <remote>`, avec <local> pour l'emplacement du fichier local et <remote> pour l'emplacement où nous allons déplacer le fichier.

```
PS C:\Users\kalle> adb push "C:\Users\kalle\OneDrive\Bureau\Karim\Photos\Karim Photo ID.jpg" /sdcard/Download/
C:\Users\kalle\OneDrive\Bureau\Karim\Photos\Karim Photo ID...file pushed, 0 skipped. 59.3 MB/s (313892 bytes in 0.005s)
```

FIG. 4 : Copie de l'image sur le PC vers l'appareil Android

La commande dans la figure est

```
adb push "C:\Users\kalle\OneDrive\Bureau\Karim\Photos\Karim Photo ID.jpg"
/sdcard/Download/
```

3.1.1 Vérification du transfert d'image

Pour vérifier la présence et la taille du fichier dans le smartphone, nous utilisons la commande `adb shell ls -l <remote>`, avec `<remote>` est le path du fichier.

```
PS C:\Users\kalle> adb shell ls -l /sdcard/Download/Karim\ Photo\ ID.jpg
-rw-rw---- 1 root sdcard_rw 313892 2024-11-26 12:14 /sdcard/Download/Karim Photo ID.jpg
```

FIG. 5 : Vérification de l'existence et la taille de l'image sous Android

313892 est la taille du fichier en octets.

Attention

→ Si le nom du fichier contient d'espaces, on ajoute \ avant l'espace.

Maintenant, nous voyons que la taille est la même sur le PC d'après la figure ci-dessous.

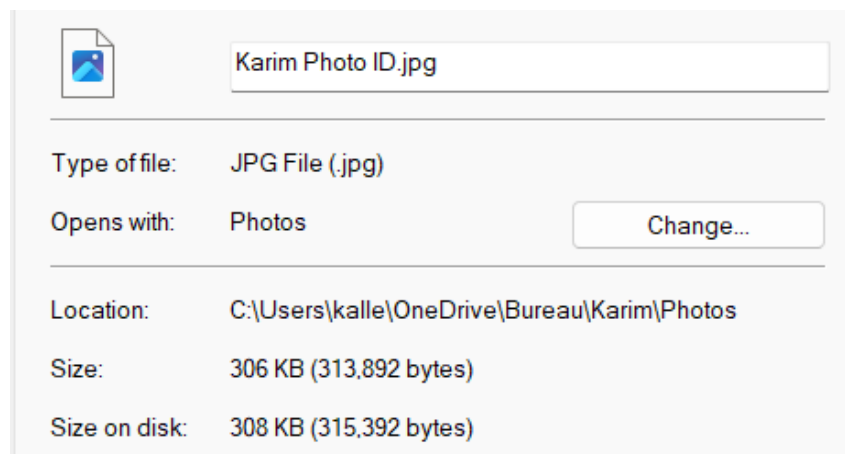


FIG. 6 : Taille de l'image sur PC

3.2 Création des fichiers sous le terminal mobile et transfert vers PC

Pour se connecter à l'émulateur afin d'exécuter des commandes root comme création et listement des fichiers, nous mettons à exécution la commande `adb root`.

```
PS C:\Users\kalle> adb root
restarting adbd as root
```

FIG. 7 : Affectation des permissions root au adb

3.2.1 Connection au terminal de l'appareil

nous lançons la commande `adb shell` pour se connecter sur le terminal mobile.

```
PS C:\Users\kalle> adb shell
generic_x86:/ #
```

FIG. 8 : Connection au terminal de l'émulateur

3.2.2 Exécution des commandes Shell et création des fichiers

Maintenant, nous exécutons les commandes `ps -ef` et `ls -l` et rediriger leurs sorties vers deux fichiers ; `ps_output.txt` et `ls_output.txt` sous le dossier `sdcard`.

```
generic_x86:/ # ps -ef > /sdcard/ps_output.txt
1|generic_x86:/ # ls -l > /sdcard/ls_output.txt
generic_x86:/ # |
```

FIG. 9 : Exécution des commandes `ps -ef` & `ls -l`

Nous devons vérifier l'existence des fichiers avant de passer à la prochaine étape.

```
generic_x86:/ # ls /sdcard
Alarms Android DCIM Download Movies Music Notifications Pictures Podcasts Ringtones ls_output.txt ps_output.txt
generic_x86:/ # |
```

FIG. 10 : Vérification de l'existence des fichiers

La commande dans la figure est

```
ls /sdcard
```

3.2.3 Récupération des fichiers sur le PC

Enfin, nous allons récupérer les fichiers sur le PC. Pour cela, nous suivons les étapes suivantes :

1. Sortir du terminal de l'émulateur avec la commande `exit`.
2. Lancer la commande `adb pull <remote> <local>`, avec `<remote>` est l'emplacement des fichiers textes et `<local>` spécifie le chemin du fichier sur le PC.

```
PS C:\Users\kalle> adb pull /sdcard/ps_output.txt C:\adb\ps_output.txt
/sdcard/ps_output.txt: 1 file pulled, 0 skipped. 0.0 MB/s (14 bytes in 0.007s)
PS C:\Users\kalle> adb pull /sdcard/ls_output.txt C:\adb\ls_output.txt
/sdcard/ls_output.txt: 1 file pulled, 0 skipped. 0.7 MB/s (2981 bytes in 0.004s)
```

FIG. 11 : Récupération des fichiers textes du smartphone au PC

3. Vérifier l'existence des fichiers en local.

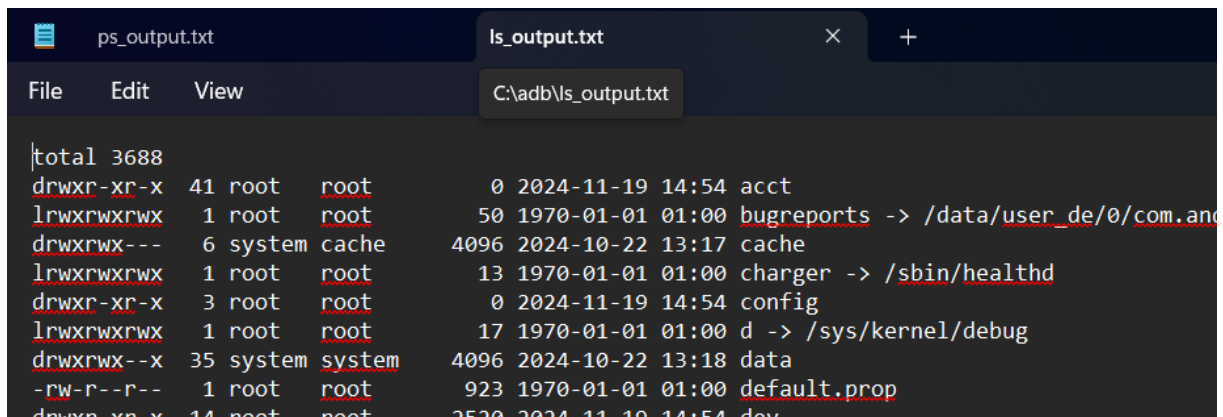
```
PS C:\Users\kalle> cd C:\adb
PS C:\adb> ls

Directory: C:\adb

Mode                LastWriteTime         Length Name
----                -
-a----           26/11/2024    21:14         2981 ls_output.txt
-a----           26/11/2024    21:14          14 ps_output.txt
```

FIG. 12 : Vérification de l'existence des fichiers en local

4. Vérifier le contenu des fichiers en local.



```
total 3688
drwxr-xr-x 41 root root 0 2024-11-19 14:54 acct
lrwxrwxrwx 1 root root 50 1970-01-01 01:00 bugreports -> /data/user_de/0/com.anc
drwxrwx--- 6 system cache 4096 2024-10-22 13:17 cache
lrwxrwxrwx 1 root root 13 1970-01-01 01:00 charger -> /sbin/healthd
drwxr-xr-x 3 root root 0 2024-11-19 14:54 config
lrwxrwxrwx 1 root root 17 1970-01-01 01:00 d -> /sys/kernel/debug
drwxrwx--- 35 system system 4096 2024-10-22 13:18 data
-rw-r--r-- 1 root root 923 1970-01-01 01:00 default.prop
drwxr-xr-x 14 root root 2520 2024-11-19 14:54 dev
```

FIG. 13 : Contenu du fichier ls_output.txt en local

3.3 Récupération des contacts du mobile au PC

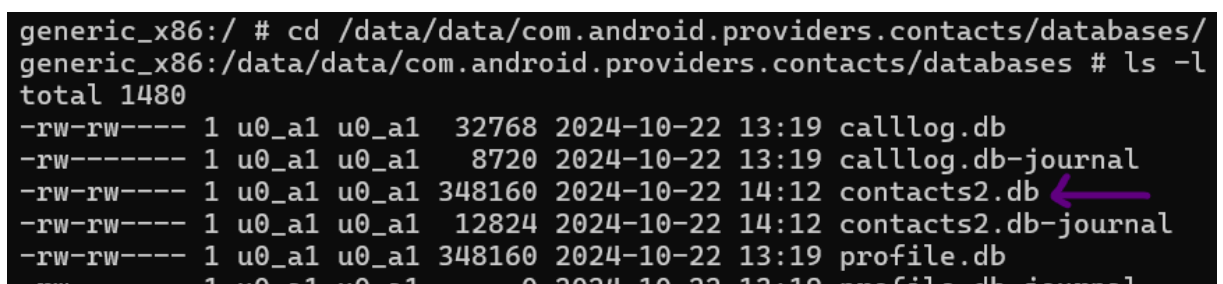
Dans cette section, nous allons récupérer la base de données des contacts de l'émulateur vers le PC.

3.3.1 Connection au terminal de l'appareil

Pour se connecter sur le terminal mobile avec ADB, nous exécutons la commande `adb shell` comme celle dans la partie "3.2.1".

3.3.2 Recherche du dossier des contacts sous l'appareil

La base de données des contacts est généralement stockée dans un fichier SQLite sous `/data/data/com.android.providers.contacts/databases/`. Donc, nous accédons à ce dossier pour vérifier son contenu.

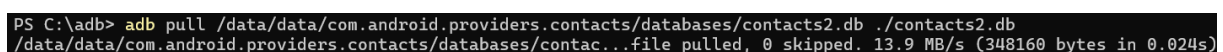


```
generic_x86:/ # cd /data/data/com.android.providers.contacts/databases/
generic_x86:/data/data/com.android.providers.contacts/databases # ls -l
total 1480
-rw-rw---- 1 u0_a1 u0_a1 32768 2024-10-22 13:19 calllog.db
-rw-rw---- 1 u0_a1 u0_a1 8720 2024-10-22 13:19 calllog.db-journal
-rw-rw---- 1 u0_a1 u0_a1 348160 2024-10-22 14:12 contacts2.db
-rw-rw---- 1 u0_a1 u0_a1 12824 2024-10-22 14:12 contacts2.db-journal
-rw-rw---- 1 u0_a1 u0_a1 348160 2024-10-22 13:19 profile.db
-rw-rw---- 1 u0_a1 u0_a1 0 2024-10-22 13:19 profile.db-journal
```

FIG. 14 : Vérification de l'existence de la base de données des contacts

3.3.3 Récupération des contacts en local

Maintenant, nous récupérons la base de données des contacts avec la commande `adb pull <remote> <local>`.



```
PS C:\adb> adb pull /data/data/com.android.providers.contacts/databases/contacts2.db ./contacts2.db
/data/data/com.android.providers.contacts/databases/contacts2.db: file pulled, 0 skipped. 13.9 MB/s (348160 bytes in 0.024s)
```

FIG. 15 : Récupération de la base des contacts sur le PC

La commande dans la figure est

```
adb pull /data/data/com.android.providers.contacts/databases/contacts2.db
./contacts2.db
```

Nous pouvons vérifier maintenant que le fichier .db existe dans sous C :/adb.

3.4 Installation manuelle d'un APK

Pour installer un APK d'une façon manuelle sur les téléphones, nous allons exploiter quelques commandes ADB et suivre les étapes suivantes.

3.4.1 Vérification préalable de l'inexistence d'APK

Nous allons vérifier l'inexistence de l'apk avant de continuer à travers la commande `adb shell pm list packages | findstr <package_name>`. Il faut que cette commande affiche rien.

```
C:\adb>adb shell pm list packages | findstr com.example.testadb
C:\adb>|
```

FIG. 16 : Vérification de l'inexistence d'apk

3.4.2 Positionnement sur le dossier et création d'archive

À ce moment, nous allons se positionner sur *C : \Users\kalle\AndroidStudioProjects* et nous créons un fichier zip par la commande suivante :

`powershell Compress-Archive -Path <file_name> -DestinationPath <file_name>.zip`

```
C:\adb>cd "C:\Users\kalle\AndroidStudioProjects"
C:\Users\kalle\AndroidStudioProjects>powershell Compress-Archive -Path testadb -DestinationPath testadb.zip
```

FIG. 17 : Navigation et compression de l'application

3.4.3 Connection au shell mobile

Pour connecter au shell du mobile, nous suivons la même étape mentionnée dans la partie "3.2.1".

3.4.4 Vérification du signature

Quand nous avons créé l'application sur Android, l'apk ne sera pas généré automatiquement qu'au moment du lancement. Pour ceci, nous allons naviger sur Android Studio à travers la barre d'outils vers *Build > Build Bundle(s)/APK(s) > Build APK(s)* pour générer l'apk. Puis, nous allons vérifier sa signature par la commande suivante :

```
jarsigner verify --verbose --print-certs
C:\Users\kalle\AndroidStudioProjects\testadb\app\build\outputs\
apk\debug\app-debug.apk
```

```
C:\Users\kalle\AndroidStudioProjects>jarsigner -verify -verbose -certs C:\Users\kalle\AndroidStudioProjects\testadb\app\build\outputs\apk\debug\app-debug.apk

s = signature was verified
m = entry is listed in manifest
k = at least one certificate was found in keystore

no manifest.

jar is unsigned.
```

FIG. 18 : Vérification du signature

3.4.5 Installation du apk

Maintenant on va installer l'apk généré dans l'émulateur en suivant ces étapes.

1. Copier l'apk du PC vers le mobile

```
adb push
C:\Users\kalle\AndroidStudioProjects\testadb\app\build\outputs\
apk\debug\app-debug.apk /data/local/tmp/app-debug.apk
```

2. Installer l'apk sur le mobile en utilisant la commande
`adb shell pm install /data/local/tmp/app-debug.apk`

```
C:\adb>adb push C:\Users\kalle\AndroidStudioProjects\testadb\app\build\outputs\apk\de
bug\app-debug.apk /data/local/tmp/app-debug.apk
C:\Users\kalle\AndroidStudioProjects\test...ped. 113.6 MB/s (5883078 bytes in 0.049s)

C:\adb>adb shell pm install /data/local/tmp/app-debug.apk
Success
```

FIG. 19 : Installation du apk dans le mobile

3.4.6 Vérification de l'installation du apk

- Pour vérifier l'existence du apk.

```
127|generic_x86:/ # pm list packages | grep com.example.testadb
package:com.example.testadb
```

FIG. 20 : Vérification de l'installation du apk

- Pour vérifier le fonctionnement du apk.

```
generic_x86:/ # monkey -p com.example.testadb -c android.intent.category.LAUNCHER 1
Events injected: 1
## Network stats: elapsed time=41ms (0ms mobile, 0ms wifi, 41ms not connected)
```

FIG. 21 : Vérification du fonctionnement du apk

3.4.7 Lancement de l'activité

Pour lancer l'activité, nous utilisons la commande
`adb shell am start -n <package_name>/<activity_name>`

```
1|generic_x86:/ # am start -n com.example.testadb/com.example.testadb.MainActivity
Starting: Intent { cmp=com.example.testadb/.MainActivity }
```

FIG. 22 : Lancement de l'activité

3.4.8 Lancement exécutive de l'activité et activation du profilage

Maintenant, nous allons suivre les étapes suivantes :

- Débuter le profilage

```
generic_x86:/ # am start -n com.example.testadb/com.example.testadb.MainActivity -P
/data/local/tmp/profiling_output.txt
Starting: Intent { cmp=com.example.testadb/.MainActivity }
```

FIG. 23 : Profilage

- Exécuter 20 fois avec cette commande.

```
FOR /L %i IN (1,1,20) DO adb shell am start -n
com.example.testadb/com.example.testadb.MainActivity
-P /data/local/tmp/profiling_output.txt
```

```
C:\adb>FOR /L %i IN (1,1,20) DO adb shell am start -n com.example.testadb/com.example.testadb.MainActivity
y -P /data/local/tmp/profiling_output.txt

C:\adb>adb shell am start -n com.example.testadb/com.example.testadb.MainActivity -P /data/local/tmp/prof
iling_output.txt
Starting: Intent { cmp=com.example.testadb/.MainActivity }

C:\adb>adb shell am start -n com.example.testadb/com.example.testadb.MainActivity -P /data/local/tmp/prof
iling_output.txt
Starting: Intent { cmp=com.example.testadb/.MainActivity }

C:\adb>adb shell am start -n com.example.testadb/com.example.testadb.MainActivity -P /data/local/tmp/prof
iling_output.txt
Starting: Intent { cmp=com.example.testadb/.MainActivity }

C:\adb>adb shell am start -n com.example.testadb/com.example.testadb.MainActivity -P /data/local/tmp/prof
iling_output.txt
Starting: Intent { cmp=com.example.testadb/.MainActivity }

C:\adb>adb shell am start -n com.example.testadb/com.example.testadb.MainActivity -P /data/local/tmp/prof
```

FIG. 24 : Exécution 20 fois

- Copier le fichier de profilage vers le PC avec la fameuse commande `adb pull <remote> <local>`

```
C:\adb>adb pull /data/local/tmp/profiling_output.txt .\profiling_output.txt
/data/local/tmp/profiling_output.txt: 1 file pulled, 0 skipped. 43.1 MB/s (1296333 bytes in 0.029s)
```

FIG. 25 : Copie du fichier de profilage vers PC

3.5 Désinstallation manuelle d'un APK

Dans cette partie, nous allons désinstaller un APK à partir du terminal.

3.5.1 Connection au shell mobile

Pour connecter au shell du mobile, nous suivons la même étape mentionnée dans la partie "3.2.1".

3.5.2 Listement des packages mobiles

Après accès au shell mobile, nous allons lancer la commande `pm list packages`.

```
C:\adb>adb shell
generic_x86:/ # pm list packages
package:com.android.smoketest
package:com.android.cts.priv.ctsshim
package:com.google.android.youtube
package:com.google.android.ext.services
package:com.example.android.livecubes
package:com.example.servicedemo
package:com.android.providers.telephony
package:com.google.android.googlequicksearchbo
package:com.android.providers.calendar
package:com.android.providers.media
package:com.google.android.onetimeinitializer
```

FIG. 26 : Liste des packages

3.5.3 Récupération de la liste dans un fichier en local

En utilisant la commande précédente, nous pouvons lui concaténer ceci `> /data/local/tmp/pm_output.txt` pour récupérer la liste des packages dans un fichier texte.

```
1|generic_x86:/ # pm list packages > /data/local/tmp/pm_output.txt
generic_x86:/ # cat /data/local/tmp/pm_output.txt
```

FIG. 27 : Création du fichier contenant les packages

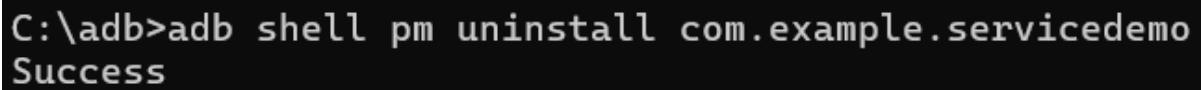
Maintenant, nous allons copier ce fichier sur le PC à travers la commande `adb pull <remote> <local>`

```
C:\adb>adb pull /data/local/tmp/pm_output.txt ./pm_output.txt
/data/local/tmp/pm_output.txt: 1 file pulled, 0 skipped. 0.4 MB/s (3644 bytes in 0.010s)
```

FIG. 28 : Copie du fichier des packages sur PC

3.5.4 Désinstallation d'un package

Nous allons lancer la commande `adb shell pm uninstall <package_name>`



```
C:\adb>adb shell pm uninstall com.example.servicedemo
Success
```

FIG. 29 : Désinstallation du package

Conclusion

Ce rapport met en lumière l'importance de l'Android Debug Bridge (ADB) comme outil incontournable pour les développeurs d'applications Android. En détaillant sa configuration et ses commandes essentielles, il souligne comment ADB facilite le débogage et la gestion des appareils. La maîtrise de cet outil permet d'optimiser le processus de développement et d'améliorer la gestion des appareils Android.