Android Development Environment & Emulator

ICT2105 Mobile Application Development Spring 2020

Overview

This lab provides instructions to run the Android emulator, gain familiarity with the Android Virtual Device (AVD), use the adb tool, and create a simple application.

Outcomes

Upon completion of the session, you should be able to:

- Create a virtual device and use the Android Emulator
- Connect and get a shell on the emulator and on a real device
- Execute some simple shell commands
- Become familiar with the adb tool
- Create a simple Android application

Using the Android Emulator & AVD Manager

This section briefly describes creating a device definition and running the Android Emulator.

1. In Android Studio, select **Tools > Android > AVD Manager**, or click the AVD Manager icon in the toolbar.
   (You might have to create a project first to get to the main toolbar.)
2. Follow the instructions in the following link to launch the AVD manager, create a virtual device, and create a device definition:
   https://developer.android.com/studio/run/managing-avds
3. Once you have created a device definition, you can also start the AVD using the command line by changing the directory to **`%USERPROFILE%\.android\avd`**

Issue the following command:

```
$ emulator -avd <name of the avd> -no-boot-anim
```

An example could be:

```
$ emulator -avd Pixel_2 -no-boot-anim
```

Question: What do the arguments of the emulator command mean?

Question: How long did the emulator take to boot? Why? Will HAXM help? What is the benefit of HAXM?

Unlock and explore the emulator and navigate the installed applications. Become familiar with the Android launcher and interface and scrolling and launching applications.

More information and instructions are located here:

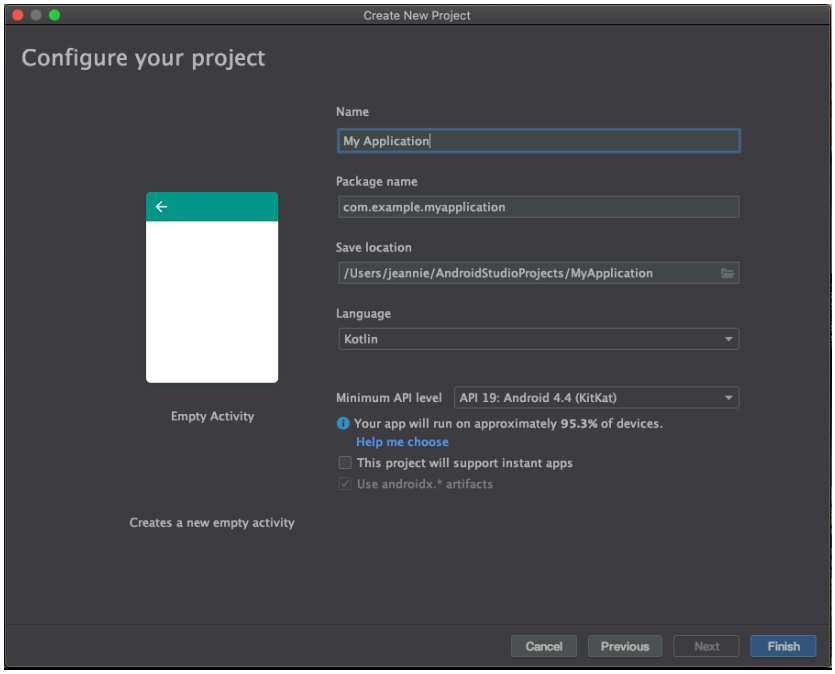
- https://developer.android.com/training/basics/firstapp/running-app
- https://developer.android.com/studio/run/managing-avds

Create an Android Application

This section briefly describes creating a new Android Application and project. More information and instructions are located here:

https://developer.android.com/training/basics/firstapp/creating-project

1. In Android Studio, create a new project:
    - If you don't have a project opened, in the Welcome screen, click **New Project**.
    - If you have a project opened, from the File menu, select **New Project**.
2. Under Choose your project, select “Empty Activity”.
3. Under Configure your new project, fill in the fields as shown below and click **Next**.



Application Name is the app name that appears to users. For this project, use "HelloWorld”

Company domain provides a qualifier that will be appended to the package name; Android Studio will remember this qualifier for each new project you create.

Package name is the fully qualified name for the project (following the same rules as those for naming packages in the Java programming language). Your package name must be unique across all packages installed on the Android system. You can edit this value independently from the application name or the company domain.

Project location is the directory on your system that holds the project files. Configure this to what you like.

4. Under Select the form factors your app will run on, check the box for Phone and Tablet.
5. For Minimum SDK, select API 19 Android 4.4 KitKat or higher

The Minimum Required SDK is the earliest version of Android that your app supports, indicated using the API level.

6. Click **Finish**. Your Android project is now a basic "Hello World" app that contains some default files.
7. Run the app from Android Studio
8. Select one of your project's files and click Run (the Play button) from the toolbar.
9. In the Choose Device window that appears, select the Choose a running device radio button, select your device, and click OK.
10. Android Studio installs the app on your connected device and starts it.

Using the ADB Tool and Getting a Shell

ADB is the Android Debug Bridge. More information on ADB can be found here: https://developer.android.com/studio/command-line/adb

Connect to the emulator and get a shell

1. Open a command line window and navigate to your development folder
2. Before issuing adb commands, it is helpful to know what emulator/device instances are connected to the adb server. You can generate a list of attached emulators/devices using the devices command: **`adb devices`**
3. You should see the list of attached devices.
4. Within an adb shell, you can issue commands directly to the device via command line. This is similar to a unix shell.
5. Get shell access to the device with the command **`adb shell`**. You should see the shell prompt when you successfully get onto your device
6. Issue the command **`ps`**. What processes do you see on the emulator? (You need to issue the command **`ps -A`** to see all processes in Android Oreo)
7. Issue the command **`df`**. What do you see?

Question: What does the ps command do?

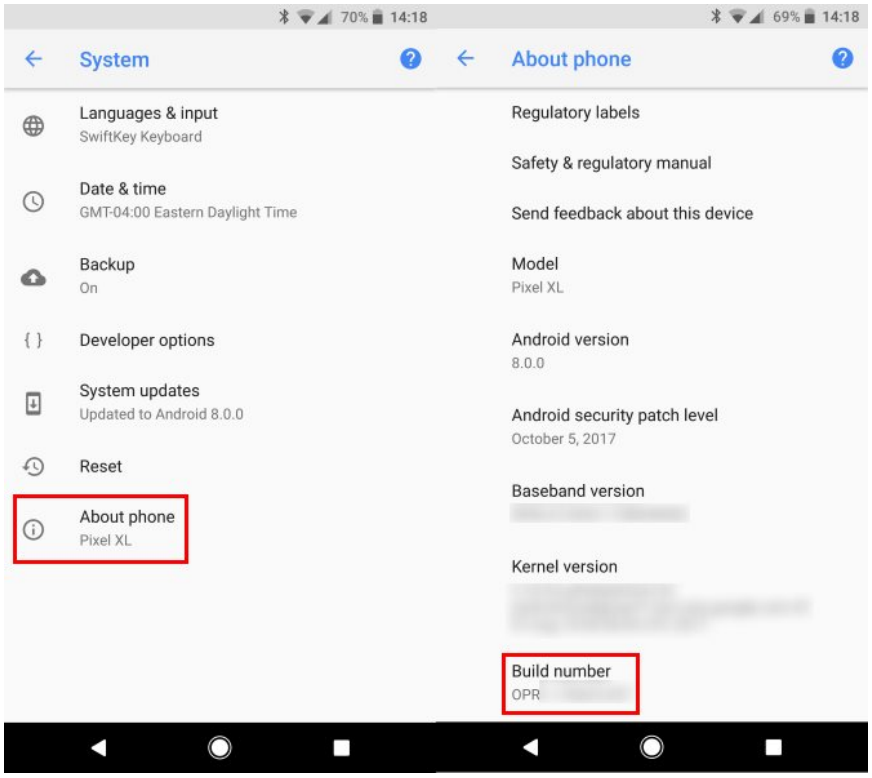
Question: What does the df command do?

Connect to a real device and get a shell

1. The appropriate USB driver for the device will first need to be installed. The USB drivers for the device's model should first be downloaded and installed. Check the manufacturer's page (e.g. Samsung or HTC) on how the drivers can be obtained.
2. Enable **USB debugging** on your device.
   - On most devices running Android 3.2 or older, you can find the option under: Settings > Applications > Development.
   - On Android 4.0 and newer, it's in: Settings > Developer options.

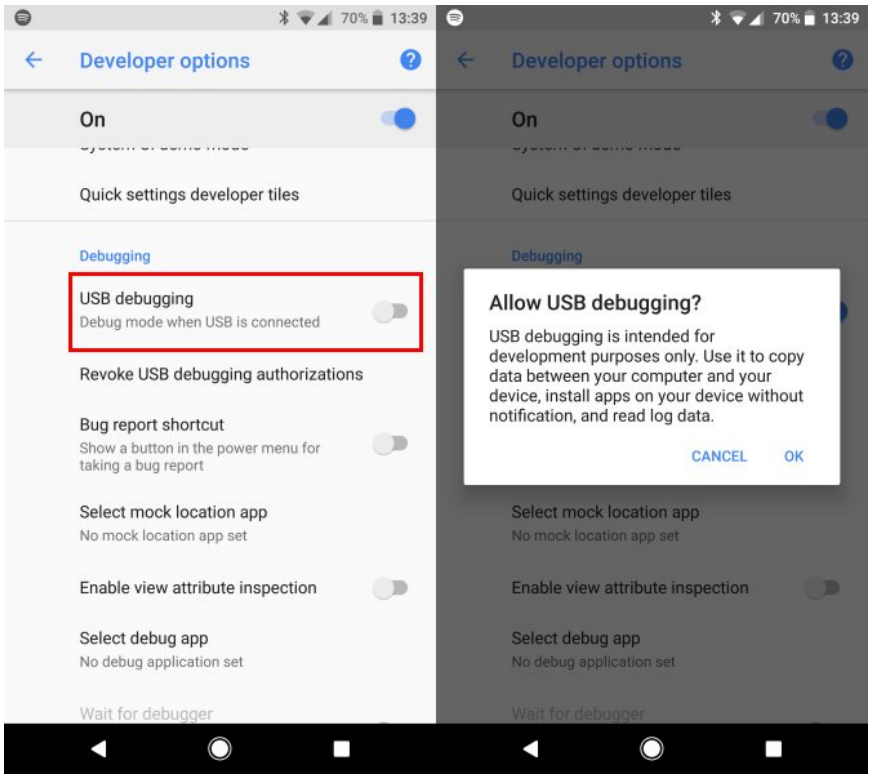
   On Android 4.2 and newer, Developer options are hidden by default.

   To make it available, go to Settings > About phone and **tap the build number seven times**.

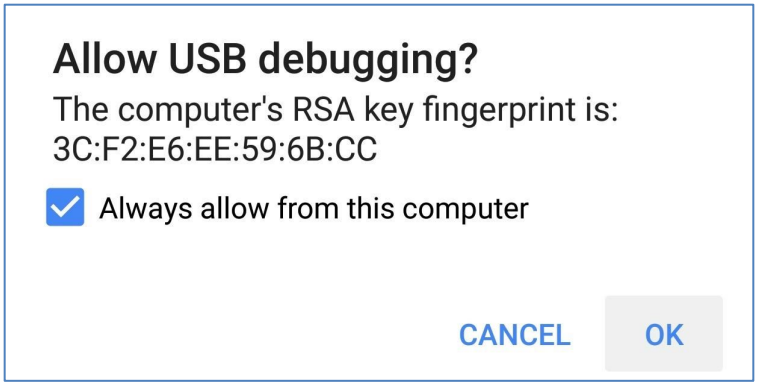


Return to the previous screen to find **Developer Options**.

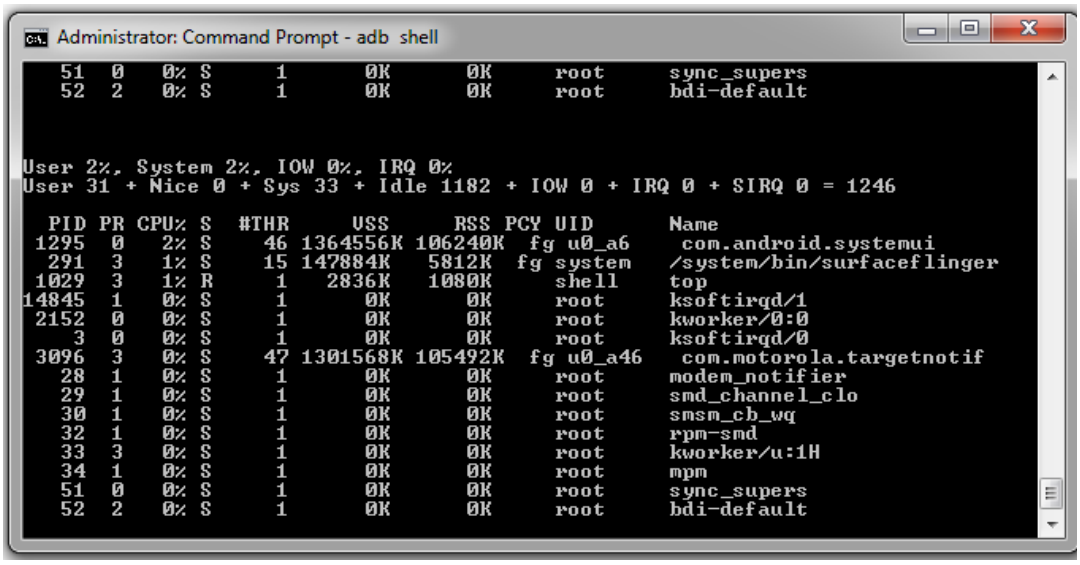
Select **Developer options**, and look for USB debugging under the Debugging header. Hit the slider to enable it, and confirm Android's warning that you understand what this feature is for.



3. If the device asks for permission, you may select "Always allow from this computer" and hit OK.



4. Issue the command **`ps`** again. How is it different from the emulator?
   (Issue the command **`ps -A`** to see all processes in Android Oreo)
5. Issue the command **`top -m 15`**.
   (Simply issue the command **`top`** in Android Oreo)
   Question: What does the top command do?



Android Profiler

The Android Profiler provide real-time data and analysis tools to help understand how the app uses CPU, memory, network, and battery resources. The following links provide information on running and exploring the further features of the Android Profiler:

https://developer.android.com/studio/profile/android-profiler

Debug an Android Application

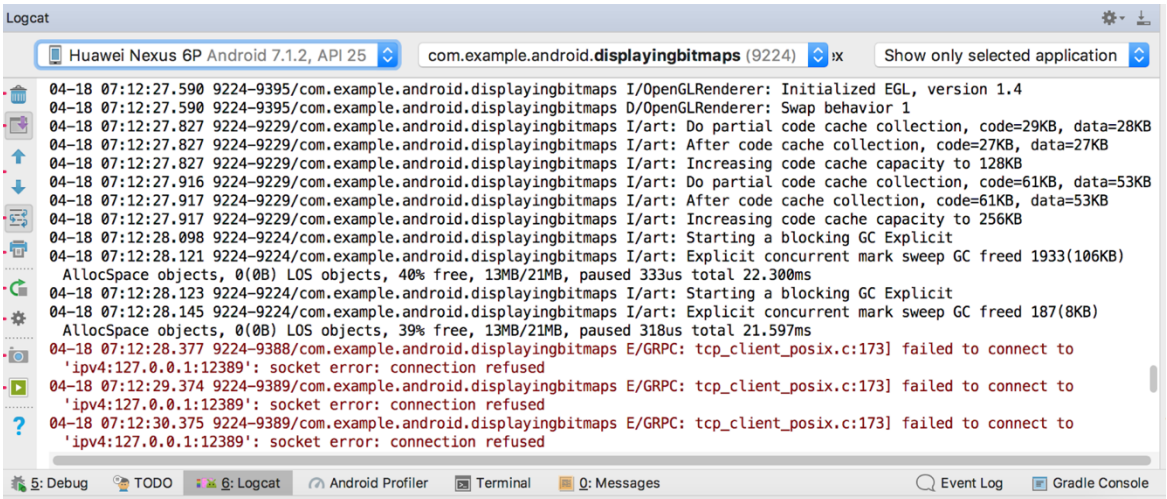
This section briefly describes debugging an app on the Android Emulator. More information and instructions are located here:

https://developer.android.com/studio/debug/

Explore and learn to set breakpoints, step into and step over, and create logging messages in the code with the Android Log class. More info on writing a log message:

https://developer.android.com/studio/debug/am-logcat

Question: How do you print a log? Where do you see the output of the log?



Use version control (Git and GitHub)

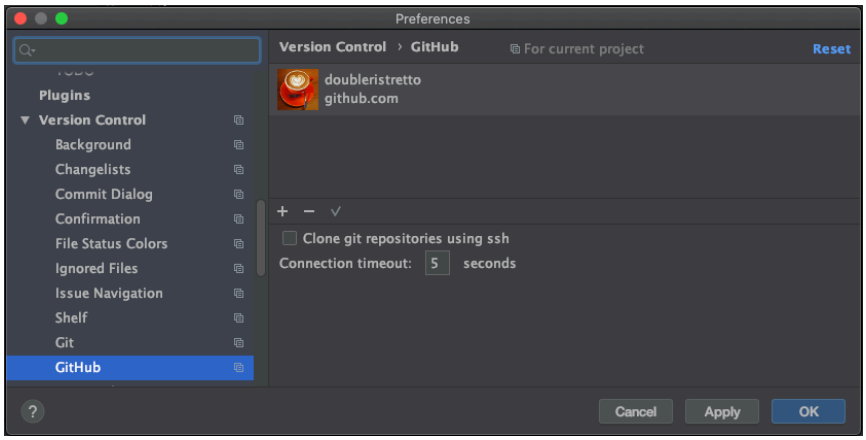
Git is a popular version control system used in industry. This section briefly covers usage of Git within Android Studio. More information on Git can be found here:

http://git-scm.com/book/en/v2/Getting-Started-Git-Basics

Share repository on GitHub

Configure Android Studio to use GitHub and Git for version control.

1. Go to **Settings or Preferences** and enter your GitHub credentials and configuration. This will set up GitHub within Android Studio.



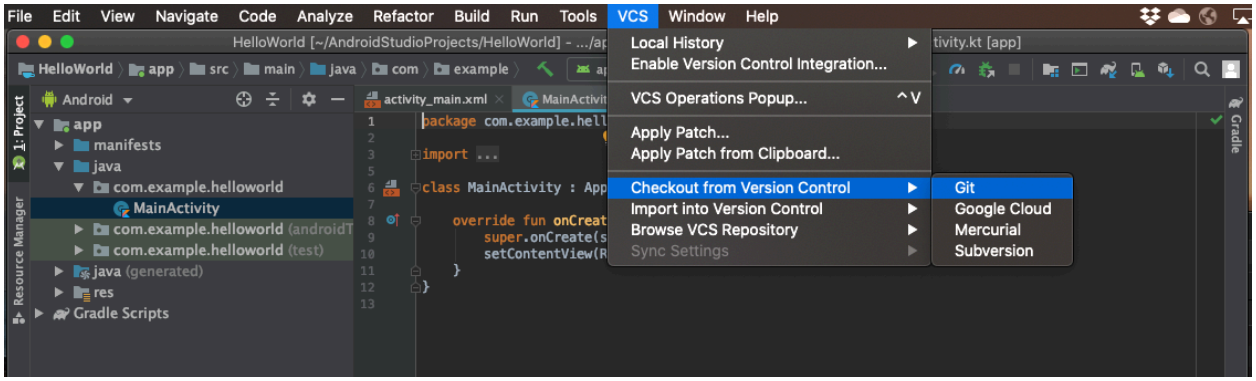
2. With the project open in Android Studio, on the top menu, select ***VCS*** > ***Import into Version Control*** > **Share Project on GitHub**
3. Give a name for your repository
4. Select all the files under the folder, write a commit message, then hit OK. You just committed all files.
5. Make some changes to your code. Commit all these changes into your GitHub repository. How do you do this?
6. Now push the changes to the repository.

Another tutorial on using GitHub with Android Studio is detailed here:

https://www.londonappdeveloper.com/how-to-use-git-hub-with-android-studio/

Get existing repository on GitHub

1. In Android Studio, On the top menu, select ***VCS*** > ***Checkout From Version Control*** > **Git**



2. In the clone repository window, enter the example project's URL:

   https://github.com/sgtech-ict2105/exampleproject.git

   The entire project will be checked out from GitHub. If prompted by Android Studio, create an Android Studio project from the checked-out sources.

Configure the team project repository for Continuous Integration (CI)

Each project team is assigned a code repository. The continuous integration service Travis CI https://travis-ci.org will be used for the purposes of the project.

The CI build status for each team can be viewed online, the URL differs depending if the repo and build are public or private:

Public builds: https://travis-ci.org

Private builds: https://travis-ci.com

The example URL for Team 1 is the following (It does not currently work until Team 1 has configured it.):

https://travis-ci.com/sgtech-ict2105/ict2105-team01-2020

Figure out your team's Travis CI URL to view the build status.

An example working team project repository with CI can be viewed here:

https://github.com/sgtech-ict2105/exampleteamproject

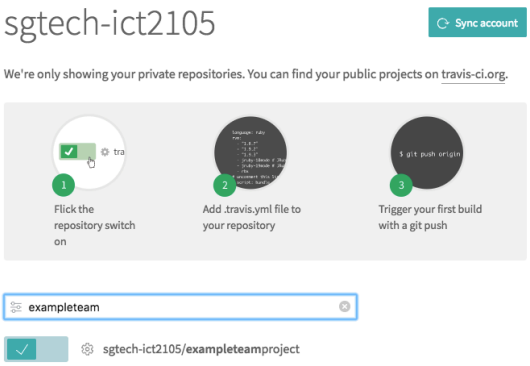
Set up continuous integration (CI) on TravisCI for your team's repository. What is the URL where you can view the build status?

Configure continuous integration (CI) build status results to post on Slack. How do you do this?

Setup Continuous Integration (CI) for the team project repository

Note: Should first attempt this exercise on your personal repository to understand what continuous integration does.

1. First, commit and push a working Android project and code to the repository.
2. Example **`.travis.yml`** and **`.gitignore`** files are provided on LMS in the Lab1 subheading. More about what the two files do in subsequent steps.
3. A **`.gitignore`** file tells git to ignore certain files that do not need to be committed to the repository. An example of files to ignore are local project configurations for Android Studio that are individual to each team member.
4. The **`.gitignore`** file will need to be created in the root directory. Commit and push the file to the repository. (You need the dot in front of the file name, in Unix, it means a hidden file)
5. In order to recognize the project for CI, a **`.travis.yml`** file will first need to be created in the root directory.
6. Configure the control panel on travis-ci.com to enable builds by flicking the repo switch on and adding the **`.travis.yml`** to the team project repository.



7. Commit and push the **`.travis.yml`** to the repository in the root directory, and this should kick off the build.

Additional details and instructions are at the following URLs:

https://docs.travis-ci.com/user/languages/android

https://github.com/codepath/android_guides/wiki/Setting-up-Travis-CI

Lab Exercise 1

Due Date: Sun Jan 19, 2020 2359 hrs

1. Connect your phone (or start the emulator). Issue the command **`ps`**. (Or **`ps -A`** if running Android Oreo and above) Which Android processes do you see on your device? Find a way to get the results of the ps command into a text file. Save the text file as **`q1_ps_result.txt`**.
2. Issue the command **`top -m 15`**. (Or **`top`** if running Android Oreo) What are the top 10 Android processes you see on your device? Find a way to get the results of the top command as **`q2_top_result.txt`** into a text file.
3. Play music using the Google Play Music application (or any other music player application e.g. Spotify). Issue the command **`top -m 10`**. (Or **`top`** if running Android Oreo) What does it do? Save the text file as **`q3_music_top_result.txt`**.
4. What is the process that plays your music? Are there one or more processes that play your music? Hint: Observe what is using the CPU with the top command. Put your answers in **`q4_music_answer.txt`**
5. Fork the code in: https://github.com/sgtech-ict2105/ict2105-lab01-2020 Import the code **from your forked branch** into Android Studio.
6. Write code to create a debug logging message within the **`onCreate()`** function of the MainActivity. This log message should print out on the console of the Android Monitor. The tag should be "MainActivity" and the message should be "Attack of the killer androids".
7. Commit all text file answers to the root directory of your forked repository **ict2105-lab01-2020**.
8. Commit and push all code changes to forked repository **ict2105-lab01-2020**.
9. Set up continuous integration (CI) on TravisCI for your forked repository. What is the URL where you can view the build status?

END OF DOCUMENT