Android

ADB(Android debug bridge) Basic commands :

1. The adb devices command : Show all conected devices.

2. The adb push command : If you want to move a file onto your Android device programmatically, you want to use the adb push command. You'll need to know a few parameters, namely the full path of the file you're pushing, and the full path to where you want to put it. In the picture above I'm pushing a song from my Music folder on my desktop to the music folder on my phone.

3. The adb pull command: If adb push sends files to your Android device, it stands to reason the adb pull command would pull them out.

That's exactly what it does, and it works the same way as the adb push command did. You need to know both the path of the file you want to pull off, as well as the path you want it placed into. You can leave the destination path blank and it will drop the file into your tools folder to make things easy.

4. The adb reboot command: This is exactly what you think it is — a way to reboot your device from the command line. Running it is simple, just type *adb reboot* and enter.

Before you say "I can just push the button!" you have to understand that these commands can be scripted, and your device can reboot in the middle of a script if you need it to. And that's a good segue to number five.

5. The adb reboot-bootloader and adb reboot recovery commands :

Not only can you reboot your device, you can specify that it reboots to the bootloader. This is awfully handy, as sometimes those button combos are touchy, and if you have a lot of devices you can never remember them all. Some devices don't even a way to boot to the bootloader without this command. And once again, being able to use this command in a script is priceless.

Doing it is easy, just type *adb reboot-bootloader* and hit the enter key.

Most devices can also boot into the recovery directly with the *adb reboot recovery* (note there is no hyphen in this one) and some can't. It won't hurt anything to try, and if yours can't nothing will happen

6. The fastboot devices command: When you're working inside the bootloader, adb no longer works. You're not yet booted into Android, and the debugging tools aren't active to communicate with. You'll need to use the fastboot command in its place.

Fastboot is probably the most powerful Android debug tool available, and many devices don't have it enabled. If your's does, you need to be sure things are communicating. That's where the fastboot devices command comes into play. At the prompt, just type in *fastboot devices* and you should see a serial number, just like the adb devices command we looked at earlier.

If things aren't working and you are using Windows, you likely have a driver issue. Fastboot uses a different driver than adb and you'll need to source it from the manufacturer.

7. The fastboot unlock command: The fastboot unlock process will erase *everything* on your phone and reset it.

The holy grail of Android commands, fastboot flashing unlock does one thing, and one thing only -- unlocks your bootloader. It's not enabled on every phone, even phones that support fastboot but we're including it because even if you don't need it, it's an important part of Android's openness. Google doesn't care what we do with phones or tablets that we've bought, and it includes this easy way to crack them open as part of Android even if the company who made your phone doesn't support it.

Using it is easy enough. Once you've used fastboot devices to make sure everything is communicating, just type *fastboot flashing unlock* at the prompt and hit enter. Look at your device, read carefully, and choose wisely.

8. The adb shell command:

The **adb shell** command confuses a lot of folks. There are two ways to use it, one where you send a command to the device to run in its own command line shell, and one where you actually enter the device's command shell from your terminal.

The other method of using the adb shell command is using it to tell your phone to run a sh command without going into the shell. Using it is easy; type *adb shell* An example would be changing permissions on a file like so: *adb shell chmod666 /sdcard/somefile*.

Be very careful running direct commands using these methods.

9. The adb install command :

While adb push can copy files to our Android devices, adb install can actually install apps. You'll need to supply the path where you have the .apk file saved, then run it like this: *adb install TheAppName.apk*.

If you're updating an app, you use the -r switch: *adb install -r TheAppName.apk*. There is also a -s switch which tries to install on the SD card as well as other commands you probably won't ever need.

And finally, you can uninstall apps by their package name with *adb uninstall package-name-here*. Uninstall has a switch, too. The -k switch will uninstall the app but leave all the app data and cache in place.

10. The adb sideload command : An OTS (over-the-air) update is downloaded as a .zip file. You can also download that zip file manually and install it without having to wait for your phone to have the update pushed to it. The end result is the same as if you wait. But we hate waiting.

All you have to do is download the update to your computer. Plug your phone into the computer. Reboot into recovery on your phone and using the up and down volume buttons choose *Apply update from ADB*. Then hop into your favorite terminal/command line and type *adb sideload Full-Path-to-the-file.zip* and hit enter. Let things run their course, and you're golden.

And there you have it. There are plenty more commands to learn if you 're the type who likes to learn commands, but these 10 are the ones you really need to know if you if you want to start digging around at the command prompt.

Defference between Android and Linux:
Summary:

1. Android is an open source operating system developed by Android, Inc. which is now

owned by Google, Inc. whereas Linux is developed as an open source operating system under the GNU project by Linus Torvalds and many others.

2. Android is developed for Mobile Internet Devices and mobile phones whereas Linux is developed for desktops/laptops/servers.

3. The Android operating system has its own C library called Bionic whereas Linux systems use GNU C library.

4. The Android systems use flash memory instead of hard drives while the standard Linux systems use magnetic drives.

5. The Android systems have their own power manager whereas the Linux systems use APM(Advanced power management) and ACPI (Advanced Configaration and power Interface)to manage the power.

Read more: Difference Between Android and Linux | Difference Between
http://www.differencebetween.net/technology/software-technology/difference-between-android-and-linux/#ixzz57qX1qVwm