ADB Shell Commands

Based on:
https://developer.android.com/studio/command-line/shell.html

==
Intro
==

The Android Debug Bridge (adb) provides a Unix shell that you can use to run a variety of commands on an emulator or connected device. The command binaries are stored in the file system of the emulator or device, at /system/bin/...
Issuing Shell Commands

You can use the shell command to issue commands, with or without entering the adb remote shell on the emulator/device. To issue a single command without entering a remote shell, use the shell command like this:

```
adb [-d|-e|-s <serialNumber>] shell <shell_command>
```

Or enter a remote shell on an emulator/device like this:

```
adb [-d|-e|-s <serialNumber>] shell
```

When you are ready to exit the remote shell, press CTRL+D or type exit.
Using activity manager (am)

Within an adb shell, you can issue commands with the activity manager (am) tool to perform various system actions, such as start an activity, force-stop a process, broadcast an intent, modify the device screen properties, and more. While in a shell, the syntax is:

```
am <command>
```

You can also issue an activity manager command directly from adb without entering a remote shell. For example:

```
adb shell am start -a android.intent.action.VIEW
```

For a list of all the available shell programs, use the following command:

```
adb shell ls /system/bin
```

Help is available for most of the commands.

==
Available activity manager commands
==

Command   Description
start [options] <INTENT>   Start an Activity specified by <INTENT>.

See the Specification for <INTENT> arguments.

Options are:

    -D: Enable debugging.
    -W: Wait for launch to complete.
    --start-profiler <FILE>: Start profiler and send results to <FILE>.

```
    -P <FILE>: Like --start-profiler, but profiling stops when the app
goes idle.
    -R: Repeat the activity launch <COUNT> times. Prior to each repeat,
the top activity will be finished.
    -S: Force stop the target app before starting the activity.
    --opengl-trace: Enable tracing of OpenGL functions.
    --user <USER_ID> | current: Specify which user to run as; if not
specified, then run as the current user.

startservice [options] <INTENT>   Start the Service specified by
<INTENT>.

See the Specification for <INTENT> arguments.

Options are:

    --user <USER_ID> | current: Specify which user to run as; if not
specified, then run as the current user.

force-stop <PACKAGE>   Force stop everything associated with <PACKAGE>
(the app's package name).
kill [options] <PACKAGE>    Kill all processes associated with <PACKAGE>
(the app's package name). This command kills only processes that are safe
to kill and that will not impact the user experience.

Options are:

    --user <USER_ID> | all | current: Specify user whose processes to
kill; all users if not specified.

kill-all   Kill all background processes.
broadcast [options] <INTENT>      Issue a broadcast intent.

See the Specification for <INTENT> arguments.

Options are:

    [--user <USER_ID> | all | current]: Specify which user to send to; if
not specified then send to all users.

instrument [options] <COMPONENT>  Start monitoring with an
Instrumentation instance. Typically the target <COMPONENT> is the form
<TEST_PACKAGE>/<RUNNER_CLASS>.

Options are:

    -r: Print raw results (otherwise decode <REPORT_KEY_STREAMRESULT>).
Use with [-e perf true] to generate raw output for performance
measurements.
    -e <NAME> <VALUE>: Set argument <NAME> to <VALUE>. For test runners a
common form is -e <testrunner_flag> <value>[,<value>...].
    -p <FILE>: Write profiling data to <FILE>.
    -w: Wait for instrumentation to finish before returning. Required for
test runners.
    --no-window-animation: Turn off window animations while running.
    --user <USER_ID> | current: Specify which user instrumentation runs
in; current user if not specified.

profile start <PROCESS> <FILE>    Start profiler on <PROCESS>, write
results to <FILE>.
```

profile stop <PROCESS>   Stop profiler on <PROCESS>.
dumpheap [options] <PROCESS> <FILE>   Dump the heap of <PROCESS>, write to <FILE>.

Options are:

    --user [<USER_ID>|current]: When supplying a process name, specify user of process to dump; uses current user if not specified.
    -n: Dump native heap instead of managed heap.

set-debug-app [options] <PACKAGE>   Set application <PACKAGE> to debug.

Options are:

    -w: Wait for debugger when application starts.
    --persistent: Retain this value.

clear-debug-app Clear the package previous set for debugging with set-debug-app.
monitor [options]   Start monitoring for crashes or ANRs.

Options are:

    --gdb: Start gdbserv on the given port at crash/ANR.

screen-compat [on|off] <PACKAGE> Control screen compatibility mode of <PACKAGE>.

display-size [reset|<WxH>] Override emulator/device display size. This command is helpful for testing your app across different screen sizes by mimicking a small screen resolution using a device with a large screen, and vice versa.

Example:
am display-size 1280x800
display-density <dpi> Override emulator/device display density. This command is helpful for testing your app across different screen densities on high-density screen environment using a low density screen, and vice versa.

Example:
am display-density 480
to-uri <INTENT> Print the given intent specification as a URI.

See the Specification for <INTENT> arguments.
to-intent-uri <INTENT>   Print the given intent specification as an intent: URI.

See the Specification for <INTENT> arguments.
Specification for <INTENT> arguments
Using package manager (pm)

Within an adb shell, you can issue commands with the package manager (pm) tool to perform actions and queries on application packages installed on the device. While in a shell, the syntax is:

```
pm <command>
```

You can also issue a package manager command directly from adb without entering a remote shell. For example:

```
adb shell pm uninstall com.example.MyApp
```

==
Available package manager commands
==

```
Command    Description
list packages [options] <FILTER>  Prints all packages, optionally only
those whose package name contains the text in <FILTER>.

Options:

    -f: See their associated file.
    -d: Filter to only show disabled packages.
    -e: Filter to only show enabled packages.
    -s: Filter to only show system packages.
    -3: Filter to only show third party packages.
    -i: See the installer for the packages.
    -u: Also include uninstalled packages.
    --user <USER_ID>: The user space to query.

list permission-groups      Prints all known permission groups.
list permissions [options] <GROUP>     Prints all known permissions,
optionally only those in <GROUP>.

Options:

    -g: Organize by group.
    -f: Print all information.
    -s: Short summary.
    -d: Only list dangerous permissions.
    -u: List only the permissions users will see.

list instrumentation   List all test packages.

Options:

    -f: List the APK file for the test package.
    <TARGET_PACKAGE>: List test packages for only this app.

list features    Prints all features of the system.
list libraries   Prints all the libraries supported by the current
device.
list users Prints all users on the system.
path <PACKAGE>   Print the path to the APK of the given <PACKAGE>.
install [options] <PATH>    Installs a package (specified by <PATH>) to
the system.

Options:

    -l: Install the package with forward lock.
    -r: Reinstall an exisiting app, keeping its data.
    -t: Allow test APKs to be installed.
    -i <INSTALLER_PACKAGE_NAME>: Specify the installer package name.
    -s: Install package on the shared mass storage (such as sdcard).
    -f: Install package on the internal system memory.
    -d: Allow version code downgrade.
    -g: Grant all permissions listed in the app manifest.
```

```
uninstall [options] <PACKAGE>    Removes a package from the system.

Options:

    -k: Keep the data and cache directories around after package removal.

clear <PACKAGE>  Deletes all data associated with a package.
enable <PACKAGE_OR_COMPONENT>    Enable the given package or component
(written as "package/class").
disable <PACKAGE_OR_COMPONENT>   Disable the given package or component
(written as "package/class").
disable-user [options] <PACKAGE_OR_COMPONENT>

Options:

    --user <USER_ID>: The user to disable.

grant <PACKAGE_NAME> <PERMISSION>      Grant a permission to an app. On
devices running Android 6.0 (API level 23) and higher, may be any
permission declared in the app manifest. On devices running Android 5.1
(API level 22) and lower, must be an optional permission defined by the
app.
revoke <PACKAGE_NAME> <PERMISSION>     Revoke a permission from an app.
On devices running Android 6.0 (API level 23) and higher, may be any
permission declared in the app manifest. On devices running Android 5.1
(API level 22) and lower, must be an optional permission defined by the
app.
set-install-location <LOCATION>   Changes the default install location.
Location values:

    0: Auto—Let system decide the best location.
    1: Internal—install on internal device storage.
    2: External—install on external media.

Note: This is only intended for debugging; using this can cause
applications to break and other undesireable behavior.
get-install-location   Returns the current install location. Return
values:

    0 [auto]: Lets system decide the best location
    1 [internal]: Installs on internal device storage
    2 [external]: Installs on external media

set-permission-enforced <PERMISSION> [true|false]  Specifies whether the
given permission should be enforced.
trim-caches <DESIRED_FREE_SPACE>  Trim cache files to reach the given
free space.
create-user <USER_NAME>     Create a new user with the given <USER_NAME>,
printing the new user identifier of the user.
remove-user <USER_ID>  Remove the user with the given <USER_IDENTIFIER>,
deleting all data associated with that user
get-max-users    Prints the maximum number of users supported by the
device.
Taking a device screenshot

============================================
The screen cap command
============================================
```

The screencap command is a shell utility for taking a screenshot of a device display. While in a shell, the syntax is:

```
screencap <filename>
```

To use the screencap from the command line, type the following:

```
$ adb shell screencap /sdcard/screen.png
```

Here's an example screenshot session, using the adb shell to capture the screenshot and the pull command to download the file from the device:

```
$ adb shell
shell@ $ screencap /sdcard/screen.png
shell@ $ exit
$ adb pull /sdcard/screen.png
```

==
Recording a device screen
==
The screenrecord command is a shell utility for recording the display of devices running Android 4.4 (API level 19) and higher. The utility records screen activity to an MPEG-4 file.

Note: Audio is not recorded with the video file.

A developer can use this file to create promotional or training videos. While in a shell, the syntax is:

```
screenrecord [options] <filename>
```

To use screenrecord from the command line, type the following:

```
$ adb shell screenrecord /sdcard/demo.mp4
```

Stop the screen recording by pressing Ctrl-C, otherwise the recording stops automatically at three minutes or the time limit set by --time-limit.

To begin recording your device screen, run the screenrecord command to record the video. Then, run the pull command to download the video from the device to the host computer. Here's an example recording session:

```
$ adb shell
shell@ $ screenrecord --verbose /sdcard/demo.mp4
(press Ctrl-C to stop)
shell@ $ exit
$ adb pull /sdcard/demo.mp4
```

The screenrecord utility can record at any supported resolution and bit rate you request, while retaining the aspect ratio of the device display. The utility records at the native display resolution and orientation by default, with a maximum length of three minutes.

There are some known limitations of the screenrecord utility that you should be aware of when using it:

    Some devices may not be able to record at their native display resolution. If you encounter problems with screen recording, try using a lower screen resolution.

Rotation of the screen during recording is not supported. If the screen does rotate during recording, some of the screen is cut off in the recording.

screenrecord options

Options   Description
--help   Displays command syntax and options
--size <WIDTHxHEIGHT> Sets the video size: 1280x720. The default value is the device's native display resolution (if supported), 1280x720 if not. For best results, use a size supported by your device's Advanced Video Coding (AVC) encoder.
--bit-rate <RATE>   Sets the video bit rate for the video, in megabits per second. The default value is 4Mbps. You can increase the bit rate to improve video quality, but doing so results in larger movie files. The following example sets the recording bit rate to 6Mbps:

```
screenrecord --bit-rate 6000000 /sdcard/demo.mp4
```

--time-limit <TIME>   Sets the maximum recording time, in seconds. The default and maximum value is 180 (3 minutes).
--rotate   Rotates the output 90 degrees. This feature is experimental.
--verbose Displays log information on the command-line screen. If you do not set this option, the utility does not display any information while running.
Other shell commands

```
=======================================
Some other adb shell commands
=======================================
```

Shell Command   Description   Comments
dumpsys   Dumps system data to the screen. The Dalvik Debug Monitor Server (DDMS) tool offers an integrated debug environment that you may find easier to use.
dumpstate Dumps state to a file.
logcat [option]... [filter-spec]...   Enables system and app logging and prints output to the screen.
dmesg   Prints kernel debugging messages to the screen.
start   Starts (restarts) an emulator/device instance.
stop Stops execution of an emulator/device instance.