Android platform profiling

Here are some tips for Android platform developers, who build and flash system images on rooted devices:

1. After running `adb root`, simpleperf can be used to profile any process or system wide.
2. It is recommended to use the latest simpleperf available in AOSP master, if you are not working on the current master branch. Scripts are in `system/extras/simpleperf/scripts`, binaries are in `system/extras/simpleperf/scripts/bin/android`.
3. It is recommended to use `app_profiler.py` for recording, and `report_html.py` for reporting. Below is an example.

```
# Record surfaceflinger process for 10 seconds with dwarf based call graph. More ex
# scripts reference in the doc.
$ python app_profiler.py -np surfaceflinger -r "-g --duration 10"

# Generate html report.
$ python report_html.py
```

4. Since Android >= O has symbols for system libraries on device, we don't need to use unstripped binaries in `$ANDROID_PRODUCT_OUT/symbols` to report call graphs. However, they are needed to add source code and disassembly (with line numbers) in the report. Below is an example.

```
# Doing recording with app_profiler.py or simpleperf on device, and generates perf.
$ python app_profiler.py -np surfaceflinger -r "--call-graph fp --duration 10"

# Collect unstripped binaries from $ANDROID_PRODUCT_OUT/symbols to binary_cache/.
$ python binary_cache_builder.py -lib $ANDROID_PRODUCT_OUT/symbols

# Report source code and disassembly. Disassembling all binaries is slow, so it's b
# --binary_filter option to only disassemble selected binaries.
$ python report_html.py --add_source_code --source_dirs $ANDROID_BUILD_TOP --add_di
  --binary_filter surfaceflinger.so
```