Outline

- General principles
- Testing AOSP UI
- Testing Lower-Level System features
- Linaro Test Suites refactoring effort
- Discussion
General Principles

● Do Not Lock Yourself Out of Your Own Tests™
  ○ Automated tests should not depend on the testing executor
  ○ Execution should be possible standalone (local test bench) and remotely (testing farm)

● Open Closed principle
  ○ Software artifacts should be open for extension, but closed to modification
  ○ I.e. it should be possible to add new tests by adding code without modifying the code for existing tests (except of course with a good reason; refactorings for example)
  ○ E.g. using frameworks xUnit-like is very helpful

● Tests code is code just like any other
  ○ Test code must be Good™
  ○ Must be properly versioned
  ○ Can be properly released
Testing AOSP UI

- **Best practices**
  - AOSP provides developers with very flexible UIAutomator framework
  - UIObjects (controls) can be named (use unique names)
  - Open closed principle
    - Make code modular
    - Use name strings only in one place

- **Useful tools**
  - UIAutomator
    - Code written in Java
    - Requires recompilation when app code or test code changes
    - Supports JUnit results
  - AndroidViewClient (https://github.com/dtmilano/AndroidViewClient)
    - Python interface to UIAutomator
    - Uses XML output of `uiautomator dump`
    - Tests written as python scripts (can be pre-recorded from the tool)
Testing lower level system features

- **Best practices for shell scripting**
  - Write portable scripts - don’t assume bash features!
  - Run or test your script with POSIX dash shell
  - Don’t assume /bin/sh points to any specific shell (e.g. Bash)
  - Shell is a very permissive programming language and will let you shoot yourself on the foot
  - Be consistent and avoid clever code - just like you should in any other language

- **Useful tools for checking scripts for common mistakes**
  - shellcheck
  - checkbashisms
    - [http://mywiki.wooledge.org/Bashism](http://mywiki.wooledge.org/Bashism)

- **Useful tools for organizing tests**
  - shunit2 (xUnit for shell)
  - clitest (allows defining shell-based tests as "plain text" and even to embed them in documentation)
Linaro Test Suites refactoring effort

- Allow local test execution
  - Linux tests are executed on the target system
  - Handle package installation in test case
  - Support Debian/Ubuntu, Fedora/CentOS in the same script
  - Support SKIP_INSTALL for test run on OE
  - Test results saved to temporary directory
  - Parsing results is identical for local execution and in LAVA
  - AOSP tests executed using adb

- Easy integration to LAVA
  - Helper script `send-to-lava.sh`
  - Local test definition run using `test-runner.py`
  - Run single test or a set of tests
  - Convert test-case.yaml to run.sh, and run it
  - Collect results in json and csv format
Open discussion

What testing techniques and tools are you using that could be helpful for everyone?

Do you want help/suggestions/opinions from the group?
Thank You

#LAS16

For further information: www.linaro.org
LAS16 keynotes and videos on: connect.linaro.org