OpenJDK Testing Pitfalls

Stuart W. Marks
Oracle JDK Core Libraries
@stuartmarks

```
/*
 * @test
 * @summary This is an example test.
 */


public class ExampleTest {
    public static void main(String[] args) {
        Object obj = new Object();
        System.out.println("Success!");
    }
}
```

```
/*
 * @test
 * @summary This is an example test.
 * @build AnotherClass
 */

public class ExampleTest {
    public static void main(String[] args) {
        AnotherClass obj = new AnotherClass();
        System.out.println("Success!");
    }
}
```

A JTReg Tag Pitfall

- If there is no @run action, "@run main TestClass" is assumed
- @build is shorthand for @run build
- This counts as a @run action, eliminating the default behavior!
- My style rule: all tests must have an @run tag.
- New policy: the last @run action must not be a @build action
- (this might be fixed in the jtreg source, but not in a posted build)

```
// java/net/Authenticator/B4933582.sh, copyrights elided

# @test
# @bug 4933582

OS=`uname -s`
case "$OS" in
  SunOS | Linux | Darwin )
    PS=":"
    FS="/"
    ;;
  CYGWIN* )
    PS=";"
    FS="/"
    ;;
```

```
  Windows* )
    PS=";"
    FS="\\"
    ;;
  * )
    echo "Unrecognized system!"
    exit 1;
    ;;
esac
```

```
${COMPILEJAVA}${FS}bin${FS}javac ${TESTJAVACOPTS} \
    ${TESTTOOLVMOPTS} -d . \
    -classpath "${TESTSRC}${FS}..${FS}..${FS}..${FS}sun$
{FS}net${FS}www${FS}httptest" \
    ${TESTSRC}${FS}B4933582.java

rm -f cache.ser auth.save

${TESTJAVA}${FS}bin${FS}java ${TESTVMOPTS} -classpath
"${TESTSRC}${FS}..${FS}..${FS}..${FS}sun${FS}net${FS}www
${FS}httptest${PS}." B4933582 first

${TESTJAVA}${FS}bin${FS}java ${TESTVMOPTS} -classpath
"${TESTSRC}${FS}..${FS}..${FS}..${FS}sun${FS}net${FS}www
${FS}httptest${PS}." B4933582 second
```

Shell Test Pitfalls

- I'm not actually sure if the cross-platform shell stuff is necessary.
- It certainly bloats the script and makes it unreadable.
- Doesn't handle errors from javac or first test case.
- Relies on subtle shell behavior: script exit status is exit status of last command
- Adding another command, e.g. "echo Status is $?" destroys the test!

```
// test/com/sun/jmx/remote/NotificationMarshalVersions/
// TestSerializationMismatch.sh, draft early Jan 2013

$TESTJAVA/bin/javac -d $CLIENT_TESTCLASSES ...
$TESTJAVA/bin/javac -d $SERVER_TESTCLASSES ...

($TESTJAVA/bin/java -classpath $SERVER_TESTCLASSES \
    Server) > $URL_PATH 2>&1 &
SERVER_PID=$!

while true ; do
    [ -f $URL_PATH ] && break
    sleep 2
done
read JMXURL < $URL_PATH
```

```
HAS_ERRORS=`($TESTJAVA/bin/java -classpath \
    $CLIENT_TESTCLASSES Client $JMXURL) 2>&1 |
    grep -i "SEVERE: Failed to fetch notification,
stopping thread. Error is: java.rmi.UnmarshalException"`

kill "$SERVER_PID"
if [ -z "$HAS_ERRORS" ] ; then
  echo "Test PASSED"
  exit 0
fi

echo "Test FAILED"
echo $HAS_ERRORS 1>&2
exit 1
```

More Shell Test Pitfalls

- This test needs to compile two different (incompatible) versions of the same class.
- This is hard to do with jtreg @compile or @build tags.
- The easiest way is to write a shell script that calls javac, right?
- What if compilations fail?
- Compiles with $TESTJAVA instead of $COMPILEJAVA?

More Shell Test Pitfalls

- Client has to wait for server to start, hence the sleep loop.
- Still a race condition: $URL_PATH created by shell, written by newly started JVM (also, what if it's not a URL?)
- If server throws exception and exits, garbage ends up in $URL_PATH

More Shell Test Pitfalls

- If client throws an exception and exits, messages will be filtered away by grep, and test will pass!
  - The worse kind of failure: false negative, information destroyed
- Exit status of client not checked.
- Fragile dependence on a specific error message.

Recommendations

- Avoid shell tests! Is it really easier to manage subprocesses in a shell than in Java? Consider how to wait-with-timeout.
- Need better test library support for managing subprocesses.
  - See RMI test library: jdk/test/java/rmi/testlibrary
- Choose random ports (this test does that). But this requires passing information from server to client.
  - Use pipes or command-line args for this.

Recommendations

- Develop better test idioms and test library support for calling the compiler from Java programs.
  - e.g. JSR-199 can (potentially) compile a source string to bytecodes
- When testing frameworks (e.g. JMX, RMI, ...) develop test whitebox-style infrastructure for testing internal state, not error messages
- Rewriting shell tests into Java is hard and requires investment, but will probably result in better and more reliable tests.

Issues for Discussion

- Should tests clean up after themselves?
- (Related.) Should tests necessarily be runnable standalone, or is it ok for a test to rely on being run by jtreg?
  - Consider impact of test libraries on this issues.
- How verbose should tests be, particularly if they're successful?
- Tests give boolean output: PASS or FAIL. No one ever sees warnings.

Disclaimer

The preceding is intended to outline our general product direction. It is intended for information purposes only, and may not be incorporated into any contract. It is not a commitment to deliver any material, code, or functionality, and should not be relied upon in making purchasing decisions. The development, release, and timing of any features or functionality described for Oracle's products remains at the sole discretion of Oracle.