PR 5297: Testing Facilities

Vassil Vassilev; Pratyush Das
TL;DR

- Allow gtest-based unit tests to check for ROOT diagnostics
Consider

extern void Error(const char *l, const char *msg, ...);

int doIt(int x) {
    if (x < 0) {
        Error("doIt", "X must be >= 1. X = %d\n", x);
        return 1;
    }
    /* ... */
    return 0;
}

void doSomething(int x) {
    /* Error deep in the stack */
    checkX_And_Complain(x);
    /* ... */
}

TEST(checks, checkDoIt) {
    // Check the error path:
    ASSERT_EQ(-1, doIt(-42));
    // Check error message?
    /* ??? */
    // Check the rest
    ASSERT_EQ(0, doIt(42));
}
How we can’t do it today

- Ignore the output in gtest and test the error code only
  - Does not work well if the function does not have error code or the error is deeper in the call stack
- Use reference-style tests
  - Loses the exact timing of when the diagnostic happened
  - Harder debugging (requires test reduction)
  - Can’t write a gtest-style test and reference files
After PR5297

```c
extern void Error(const char *l, const char *msg, ...);
int doIt(int x) {
    if (x < 0) {
        Error("doIt", "X must be >= 1. X = %d\n", x);
        return 1;
    }
    /* ... */
    return 0;
}

void doSomething(int x) {
    /* Error deep in the call stack */
    checkX_And_Complain(x);
    /* ... */
}

TEST(checks, checkDoIt) {
    // Check the error path and error message.
    ROOT_EXPECT_ERROR(ASSERT_EQ(-1, doIt(-42)), "doIt", "X must be >= 1");
    // Check the rest
    ASSERT_EQ(0, doIt(42));
}

TEST(checks, doSomething) {
    // Check the error path and error message.
    ROOT_EXPECT_ERROR(doSomething(-42), "X must be >= 1");
}
```
After PR5297

• We can check diagnostics such as errors, warnings and notices in gtest, root.git and roottest.git

• Easier debugging of regressions

• Easier handling of ROOT options (eg. -Dimt=Off) or platform-dependent code by adding #ifdefs in one place in the test
PR5297. Cost

- These facilities exist today in root.git and we are fairly happy with the way they work and their stability.

- In order to make them available to all testing we propose to wrap them in a header which lives in root.git (so that we can cover the in-line unit tests).

- We see this as complementary facility to roottest reference files which allows better granularity wherever is needed.
Discussion