Using OpenSSL Tests

After a successful build, and before installing, the libraries should be tested. Run:

```
$ make test                                     # Unix
$ mms test                                      ! OpenVMS
$ nmake test                                    # Windows
```

Warning: you MUST run the tests from an unprivileged account (or disable your privileges temporarily if your platform allows it).

If some tests fail, take a look at the section Test Failures below.

Test Failures

If some tests fail, look at the output. There may be reasons for the failure that isn't a problem in OpenSSL itself (like an OS malfunction or a Perl issue). You may want increased verbosity, that can be accomplished like this:

Full verbosity, showing full output of all successful and failed test cases (`make` macro `VERBOSE` or `V`):

```
$ make V=1 test                                 # Unix
$ mms /macro=(V=1) test                         ! OpenVMS
$ nmake V=1 test                                # Windows
```

Verbosity on failed (sub-)tests only (`VERBOSE_FAILURE` or `VF` or `REPORT_FAILURES`):

```
$ make test VF=1
```

Verbosity on failed (sub-)tests, in addition progress on succeeded (sub-)tests (`VERBOSE_FAILURE_PROGRESS` or `VFP` or `REPORT_FAILURES_PROGRESS`):

```
$ make test VFP=1
```

If you want to run just one or a few specific tests, you can use the make variable TESTS to specify them, like this:

```
$ make TESTS='test_rsa test_dsa' test           # Unix
$ mms/macro="TESTS=test_rsa test_dsa" test      ! OpenVMS
$ nmake TESTS='test_rsa test_dsa' test          # Windows
```

And of course, you can combine (Unix examples shown):

```
$ make test TESTS='test_rsa test_dsa' VF=1
$ make test TESTS="test_cmp_*" VFP=1
```

You can find the list of available tests like this:

```
$ make list-tests                               # Unix
$ mms list-tests                                ! OpenVMS
$ nmake list-tests                              # Windows
```

Have a look at the manual for the perl module Test::Harness to see what other HARNESS_* variables there are.

To report a bug please open an issue on GitHub, at https://github.com/openssl/openssl/issues.

For more details on how the `make` variables `TESTS` can be used, see section Running Selected Tests below.

Running Selected Tests

The `make` variable `TESTS` supports a versatile set of space separated tokens with which you can specify a set of tests to be performed. With a "current set of tests" in mind, initially being empty, here are the possible tokens:

```
 alltests      The current set of tests becomes the whole set of available
               tests (as listed when you do 'make list-tests' or similar).

 xxx           Adds the test 'xxx' to the current set of tests.

-xxx           Removes 'xxx' from the current set of tests.  If this is the
               first token in the list, the current set of tests is first
               assigned the whole set of available tests, effectively making
               this token equivalent to TESTS="alltests -xxx".

 nn            Adds the test group 'nn' (which is a number) to the current
               set of tests.

-nn            Removes the test group 'nn' from the current set of tests.
               If this is the first token in the list, the current set of
               tests is first assigned the whole set of available tests,
               effectively making this token equivalent to
               TESTS="alltests -xxx".
```

Also, all tokens except for "alltests" may have wildcards, such as *. (on Unix and Windows, BSD style wildcards are supported, while on VMS, it's VMS style wildcards)

Examples

Run all tests except for the fuzz tests:

```
$ make TESTS='-test_fuzz*' test
```

or, if you want to be explicit:

```
$ make TESTS='alltests -test_fuzz*' test
```

Run all tests that have a name starting with "test_ssl" but not those starting with "test_ssl_":

```
$ make TESTS='test_ssl* -test_ssl_*' test
```

Run only test group 10:

```
$ make TESTS='10' test
```

Run all tests except the slow group (group 99):

```
$ make TESTS='-99' test
```

Run all tests in test groups 80 to 99 except for tests in group 90:

```
$ make TESTS='[89]? -90' test
```

To run specific fuzz tests you can use for instance:

```
$ make test TESTS='test_fuzz_cmp test_fuzz_cms'
```

To stochastically verify that the algorithm that produces uniformly distributed random numbers is operating correctly (with a false positive rate of 0.01%):

```
$ ./util/wrap.sh test/bntest -stochastic
```

Running Tests in Parallel

By default the test harness will execute the selected tests sequentially. Depending on the platform characteristics, running more than one test job in parallel may speed up test execution. This can be requested by setting the `HARNESS_JOBS` environment variable to a positive integer value. This specifies the maximum number of test jobs to run in parallel.

Depending on the Perl version different strategies could be adopted to select which test recipes can be run in parallel. In recent versions of Perl, unless specified otherwise, any task can be run in parallel. Consult the documentation for `TAP::Harness` to know more.

To run up to four tests in parallel at any given time:

```
$ make HARNESS_JOBS=4 test
```

Randomisation of Test Ordering

By default, the test harness will execute tests in the order they were added. By setting the `OPENSSL_TEST_RAND_ORDER` environment variable to zero, the test ordering will be randomised. If a randomly ordered test fails, the seed value used will be reported. Setting the `OPENSSL_TEST_RAND_ORDER` environment variable to this value will rerun the tests in the same order. This assures repeatability of randomly ordered test runs. This repeatability is independent of the operating system, processor or platform used.

To randomise the test ordering:

```
$ make OPENSSL_TEST_RAND_ORDER=0 test
```

To run the tests using the order defined by the random seed `42`:

```
$ make OPENSSL_TEST_RAND_ORDER=42 test
```