NAME

runtests.pl – run one or more test cases

SYNOPSIS

runtests.pl [options] [test number]

DESCRIPTION

runtests.pl runs one, several or all the existing test cases in curl's test suite. It is often called from the root Makefile of the curl package with 'make test'.

TEST NUMBER

If no test case number is given, all existing tests that the script can find will be considered for running. You can specify single test cases to run, space-separated, like "1 3 5 7 11", and you can specify a range like "45 to 67".

OPTIONS

-a Continue running the rest of the test cases even if one test fails. By default, the test script stops as soon as an error is detected.

-c <curl>
Provide a custom curl binary to run the tests with. Default is the curl executable in the build tree.

-d Enable protocol debug: have the servers display protocol output.

-g Run the given test(s) with gdb. This is best used on a single test case and curl built --disable-shared. This then fires up gdb with command line set to run the specified test case. Simply (set a break-point and) type 'run' to start.

-h Displays a help text about this program's command line options.

-k Keep output and log files in log/ after a test run, even if no error was detected. Useful for debugging.

-l Lists all test case names.

-n Disable the check for and use of valgrind.

-p Prints out all files in "log/" to stdout when a test case fails. Very practical when used in the automated and distributed tests since then the people checking the failures and the reasons for them might not have physical access to the machine and logs.

-s Shorter output. Speaks less than default.

-t[num] Selects a **torture** test for the given tests. This makes runtests.pl first run the tests once and count the number of memory allocations made. It then reruns the test that number of times, each time forcing one of the allocations to fail until all allocs have been tested. By setting *num* you can force the allocation with that number to be set to fail at once instead of looping through everyone, which is very handy when debugging and then often in combination with *-g*.

-v Enable verbose output. Speaks more than default.

RUNNING TESTS

Many tests have conditions that must be met before the test case can run fine. They could depend on built-in features in libcurl or features present in the operating system or even in 3rd party libraries that curl may or may not use.

The test script checks this by itself, why it is safe to attempt to run all tests. They who cannot be run due to failed requirements, will simply be skipped and you'll get a report about it when all test cases have completed.

WRITING TESTS

The simplest way to write test cases is to start with a similar existing test, save it with a new number and then adjust it to fit. There's an attempt to document the test case file format in the tests/FILEFORMAT.