# **Linux Test Project**

**What is LTP?**

The LTP (Linux Text Project) is a testsuite contains a collection of tools for testing the Linux Kernel and related features (including regression tests). The dominant programming language in LTP is ANSI-C (94%) followed by Bash (5%) and Perl (0.62%).

Our goal is to improve the Linux kernel and system libraries by bringing test automation to the testing effort.

**Scope of LTP**

The goal of the project has always been "to validate the reliability, robustness, and stability of Linux".

The LTP test suite is a collection of automated and semi-automated tests for testing various aspects of the Linux operating system. The goal of LTP is to deliver a suite of automated testing tools for Linux as well as publishing the results of tests we run.

**What kind of test LTP do?**

1. filesystem stress tests
2. disk I/O tests
3. memory management stress tests
4. ipc stress scheduler tests
5. commands functional varification tests
6. system call functional varification tests
7. Suspecting run time bug
8. Suspecting man pages
9. Memory Leak Analysis
10. Checking race condition
11. Avoiding Segmentations faults
12. Static Analysis
13. Non-ANSI defination
14. Splint Analysis
15. Experimenting with Fault Injection

# **Build and Installation Guide**

1) Go to the below link:

<https://linux-test-project.github.io/>

and click on Download the latest tarball icon

2) Create folder LTP and untar downloaded file using below command:

**tar xzf ltp-xxxxxxxx.tar.gz**

3) Install dependency file

Run the below command:

**sudo apt-get update**

This is a library for applications dealing with netlink sockets. The library provides an interface for raw netlink messaging and various netlink family specific interfaces.

**sudo apt-get install libnl-3-dev**

**sudo apt-get install libnl-3-genl**

CheckInstall keeps track of all the files created or modified by your installation script, builds a standard binary package (.deb, .rpm, .tgz) and installs it in your system giving you the ability to uninstall it with your distribution's standard package management utilities.

**sudo apt-get install checkinstall**

The ncurses library routines are a terminal-independent method of updating character screens with reasonable optimization. This package contains the header files, static libraries and symbolic links that developers using ncurses will need.

**sudo apt-get install libncurses-dev**

In Ubuntu, the splash screen is governed by the usplash. To make sure that it does not conflict with other splash screen, we will need to remove usplash.

sudo apt-get autoremove usplash

**GNU Automake** is a [programming tool](https://en.wikipedia.org/wiki/Programming_tool) to automate parts of the compilation process. It eases usual compilation problems

**sudo apt-get install automake**

4) Run the following commands:

4.1) **make autotools**

4.2) **./configure**

4.3) **make all**

4.4) **sudo make install**

This will install LTP to **/opt/ltp.**

* If you have a problem see doc/mini-howto-building-ltp-from-git.txt.
* If you still have a problem see INSTALL and./configure --help.
* Failing that, ask for help on the mailing list or Github.

Some tests will be disabled if the configure script can not find their build dependencies.

* If a test returns TCONF due to a missing component, check the ./configure output.
* If a tests fails due to a missing user or group, see the Quick Start section of INSTALL.

5) To run all the test suites

$ **cd /opt/ltp**

$ .**/runltp**

Note that many test cases have to be executed as root.

6) To run a particular test suite

$ **./runltp -f syscalls**

To run all tests with madvise in the name

$ **./runltp -f syscalls -s madvise**

Also see

**$ ./runltp --help**

Test suites (e.g. syscalls) are defined in the runtest directory. Each file contains a list of test cases in a simple format, see doc/ltp-run-files.txt.

Each test case has its own executable or script, these can be executed directly

**$ testcases/bin/abort01**

Some have arguments

**$ testcases/bin/fork13 -i 37**

The vast majority of test cases accept the -h (help) switch

**$ testcases/bin/ioctl01 -h**

Many require certain environment variables to be set

$ LTPROOT=/opt/ltp PATH="$PATH:$LTPROOT/testcases/bin" testcases/bin/wc01.sh

Most commonly, the path variable needs to be set and also LTPROOT, but there are a number of other variables, runltp usually sets these for you.

Note that all shell scripts need the PATH to be set. However this is not limited to shell scripts, many C based tests need environment variables as well.

**Compiling LTP for ARM 64 bit**

1. Install Ubuntu ARM64 packages

**sudo apt-get install gcc-4.8-aarch64-linux-gnu**

2. Configure build environment for ARM 64

**./configure AR=aarch64-linux-gnu-ar CC=aarch64-linux-gnu-gcc-4.8 RANLIB=aarch64-linux-gnu- ranlib --host=arm-linux –target=arm-buildroot**

3. Build all modules

**make all ARCH=arm64 CROSS\_COMPILE=aarch64-linux-gnu-**

The final step should compile source files into ARM 64 object files.

After this, copy ltp directory (/opt/ltp) to the /pi/home

To run all the test suites in /ltp

$ .**/runltp**

Note that many test cases have to be executed as root.

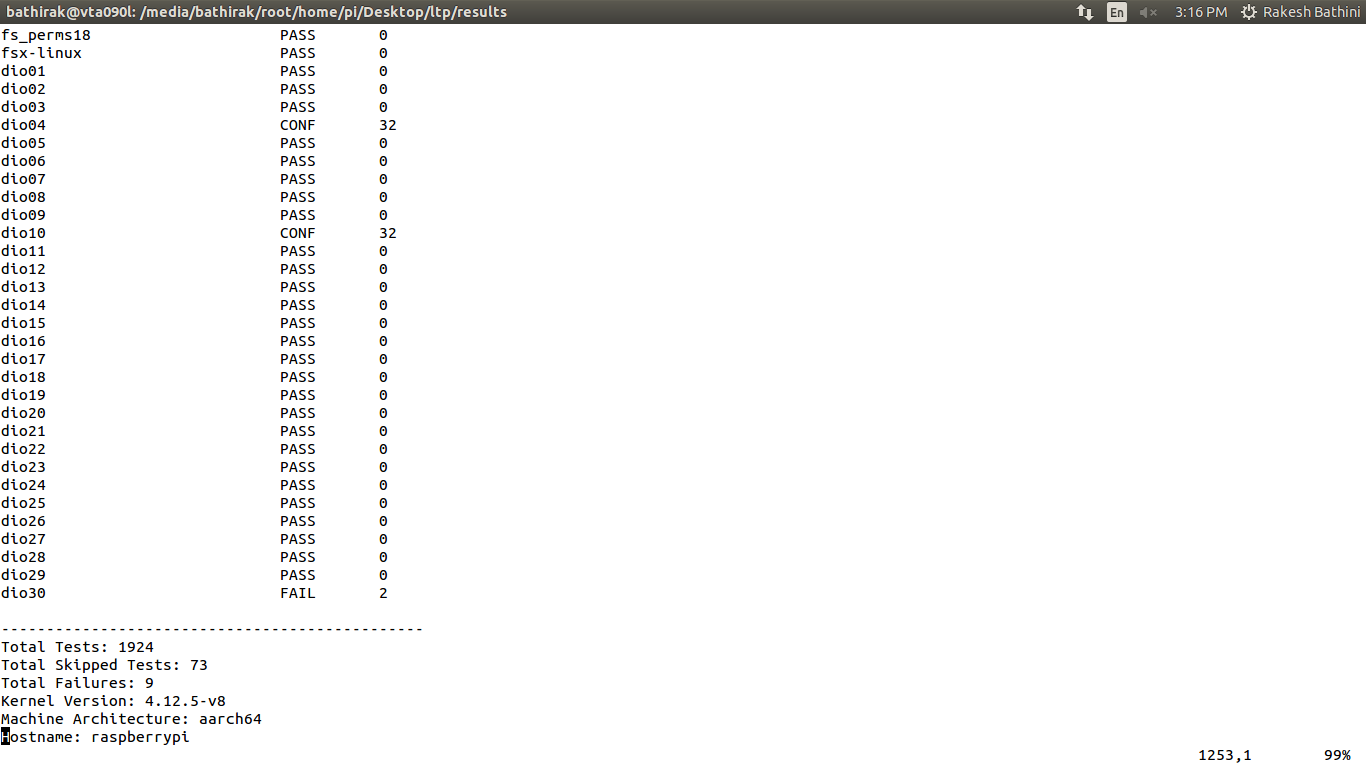
Following log you will find in /ltp/output directory

**Achieved:**

1. Performed Regression and conformance test**s** on arm based Linux Platform

**Learnt:**

1. Building kernel from source
2. The use of cross compilers for cross platform development
3. Flashing board with an Image

****

**References:**

<https://github.com/linux-test-project/ltp>

[Getting started with LTP](https://github.com/linux-test-project/ltp/wiki/GettingStarted)

* [Test Writing Guidelines](https://github.com/linux-test-project/ltp/wiki/Test-Writing-Guidelines)
* [Build system](https://github.com/linux-test-project/ltp/wiki/BuildSystem)
* [Style guide](https://github.com/linux-test-project/ltp/wiki/StyleGuide)

**https://github.com/linux-test-project/ltp/wiki/Developers**