Documentation

**Installation**

Get the latest sources from the [git repository](https://git.kernel.org/cgit/utils/rt-tests/rt-tests.git/), do a git clone <git://git.kernel.org/pub/scm/utils/rt-tests/rt-tests.git> or fetch a released [tarball from the archive](https://www.kernel.org/pub/linux/utils/rt-tests/), untar into a directory of your choice and run *make* in the source directory. If you want to cross compile, just run *make CROSS\_COMPILE=<your-compiler-prefix>* (for example *make CROSS\_COMPILE=arm-v4t-linux-gnueabi-*).

You can run the resulting binary from there or install it.

lgs@f11#> git clone git://git.kernel.org/pub/scm/utils/rt-tests/rt-tests.git

lgs@f11#> cd rt-tests

lgs@f11#> make all

lgs@f11#> cp ./cyclictest /usr/bin/

lgs@f11#> cyclictest --help

**NOTE!**

**libnuma** is required to build cyclictest. Usually, it's safe to have libnuma installed also in non-numa systems, but if you don't want to install the numa libs (e.g. in embedded environment) then compile with **make NUMA=0**.

**Run it**

Make sure to be root or use sudo to run cyclictest.

Without parameters cyclictest creates one thread with a 1ms interval timer.

cyclictest -h provides help text for the various options

[lgs@f11 rt-tests]#

[lgs@f11 rt-tests]#

[lgs@f11 rt-tests]# ./cyclictest --help

cyclictest V 0.42

Usage:

cyclictest <options>

-a [NUM] --affinity run thread #N on processor #N, if possible

with NUM pin all threads to the processor NUM

-b USEC --breaktrace=USEC send break trace command when latency > USEC

-B --preemptirqs both preempt and irqsoff tracing (used with -b)

-c CLOCK --clock=CLOCK select clock

0 = CLOCK\_MONOTONIC (default)

1 = CLOCK\_REALTIME

-C --context context switch tracing (used with -b)

-d DIST --distance=DIST distance of thread intervals in us default=500

-E --event event tracing (used with -b)

-f --ftrace function trace (when -b is active)

-i INTV --interval=INTV base interval of thread in us default=1000

-I --irqsoff Irqsoff tracing (used with -b)

-l LOOPS --loops=LOOPS number of loops: default=0(endless)

-m --mlockall lock current and future memory allocations

-n --nanosleep use clock\_nanosleep

-N --nsecs print results in ns instead of ms (default ms)

-o RED --oscope=RED oscilloscope mode, reduce verbose output by RED

-O TOPT --traceopt=TOPT trace option

-p PRIO --prio=PRIO priority of highest prio thread

-P --preemptoff Preempt off tracing (used with -b)

-q --quiet print only a summary on exit

-r --relative use relative timer instead of absolute

-s --system use sys\_nanosleep and sys\_setitimer

-T TRACE --tracer=TRACER set tracing function

configured tracers: unavailable (debugfs not mounted)

-t --threads one thread per available processor

-t [NUM] --threads=NUM number of threads:

without NUM, threads = max\_cpus

without -t default = 1

-v --verbose output values on stdout for statistics

format: n:c:v n=tasknum c=count v=value in us

-D --duration=t specify a length for the test run

default is in seconds, but 'm', 'h', or 'd' maybe add

ed

to modify value to minutes, hours or days

-h --histogram=US dump a latency histogram to stdout after the run

US is the max time to be be tracked in microseconds

-w --wakeup task wakeup tracing (used with -b)

-W --wakeuprt rt task wakeup tracing (used with -b)

**-b** is a debugging option to control the latency tracer in the realtime preemption patch.

It is useful to track down unexpected large latencies on a system. This option does only work with

* CONFIG\_PREEMPT\_RT=y
* CONFIG\_WAKEUP\_TIMING=y
* CONFIG\_LATENCY\_TRACE=y
* CONFIG\_CRITICAL\_PREEMPT\_TIMING=y
* CONFIG\_CRITICAL\_IRQSOFF\_TIMING=y

kernel configuration options enabled. The USEC parameter to the -b option defines a maximum latency value, which is compared against the actual latencies of the test. Once the measured latency is higher than the given maximum, the kernel tracer and cyclictest is stopped. The trace can be read from /proc/latency\_trace

mybox# cat /proc/latency\_trace >trace.log

Please be aware that the tracer adds significant overhead to the kernel, so the latencies will be much higher than on a kernel with latency tracing disabled.

**-c CLOCK** selects the clock, which is used

* 0 selects CLOCK\_MONOTONIC, which is the monotonic increasing system time. This is the default selection
* 1 selects CLOCK\_REALTIME, which is the time of day time.

CLOCK\_REALTIME can be set by settimeofday, while CLOCK\_MONOTONIC can not be modified by the user.

This option has no influence when the **-s** option is given.

**-d DIST** set the distance of thread intervals in microseconds (default is 500us)

When cylictest is called with the **-t** option and more than one thread is created, then this distance value is added to the interval of the threads.

Interval(thread N) = Interval(thread N-1) + DIST

**-i INTV** set the base interval of the thread(s) in microseconds (default is 1000us)

This sets the interval of the first thread. See also **-d**.

**-l LOOPS** set the number of loops (default = 0(endless))

This option is useful for automated tests with a given number of test cycles. cyclictest is stopped once the number of timer intervals has been reached.

**-n** use clock\_nanosleep instead of posix interval timers

Setting this option runs the tests with clock\_nanosleep instead of posix interval timers.

**-p PRIO** set the priority of the first thread

The given priority is set to the first test thread. Each further thread gets a lower priority:

Priority(Thread N) = Priority(Thread N-1)

**-q** run the tests quiet and print only a summary on exit

Useful for automated tests, where only the summary output needs to be captured

**-r** use relative timers instead of absolute

The default behaviour of the tests is to use absolute timers. This option is there for completeness and should not be used for reproducible tests.

**-s** use sys\_nanosleep and sys\_setitimer instead of posix timers

Note, that **-s** can only be used with one thread because itimers are per process and not per thread. **-s** in combination with **-n** uses the nanosleep syscall and is not restricted to one thread

**-t NUM** set the number of test threads (default is 1), -t without an argument makes the number of threads equal to the number of cpus

Create NUM test threads. See **-d**, **-i** and **-p** for further information.

**-v** output values on stdout for statistics

This option is used to gather statistical information about the latency distribution. The output is sent to stdout. The output format is

n:c:v

where n=task number c=count v=latency value in us

Use this option in combination with **-l**

The [OSADL Realtime LiveCD](http://www.osadl.org/projects-live-cd.0.html) project provides a script to plot the latency distribution.

Expected Results

**tglx's reference machine**

All tests have been run on a Pentium III 400MHz based PC.

The tables show comparisons of vanilla Linux 2.6.16, Linux-2.6.16-hrt5 and Linux-2.6.16-rt12. The tests for intervals less than the jiffy resolution have not been run on vanilla Linux 2.6.16. The test thread runs in all cases with SCHED\_FIFO and priority 80. All numbers are in microseconds.

* Test case: clock\_nanosleep(TIME\_ABSTIME), Interval 10000 microseconds,. 10000 loops, no load.

Commandline: *cyclictest -t1 -p 80 -n -i 10000 -l 10000*

|  |  |  |  |
| --- | --- | --- | --- |
| Kernel | min | max | avg |
| 2.6.16 | 24 | 4043 | 1989 |
| 2.6.16-hrt5 | 12 | 94 | 20 |
| 2.6.16-rt12 | 6 | 40 | 10 |

* Test case: clock\_nanosleep(TIME\_ABSTIME), Interval 10000 micro seconds,. 10000 loops, 100% load.

Commandline: *cyclictest -t1 -p 80 -n -i 10000 -l 10000*

|  |  |  |  |
| --- | --- | --- | --- |
| Kernel | min | max | avg |
| 2.6.16 | 55 | 4280 | 2198 |
| 2.6.16-hrt5 | 11 | 458 | 55 |
| 2.6.16-rt12 | 6 | 67 | 29 |

* Test case: POSIX interval timer, Interval 10000 micro seconds,. 10000 loops, no load.

Commandline: *cyclictest -t1 -p 80 -i 10000 -l 10000*

|  |  |  |  |
| --- | --- | --- | --- |
| Kernel | min | max | avg |
| 2.6.16 | 21 | 4073 | 2098 |
| 2.6.16-hrt5 | 22 | 120 | 35 |
| 2.6.16-rt12 | 20 | 60 | 31 |

* Test case: POSIX interval timer, Interval 10000 micro seconds,. 10000 loops, 100% load.

Commandline: *cyclictest -t1 -p 80 -i 10000 -l 10000*

|  |  |  |  |
| --- | --- | --- | --- |
| Kernel | min | max | avg |
| 2.6.16 | 82 | 4271 | 2089 |
| 2.6.16-hrt5 | 31 | 458 | 53 |
| 2.6.16-rt12 | 21 | 70 | 35 |

* Test case: clock\_nanosleep(TIME\_ABSTIME), Interval 500 micro seconds,. 100000 loops, no load.

Commandline: *cyclictest -t1 -p 80 -i 500 -n -l 100000*

|  |  |  |  |
| --- | --- | --- | --- |
| Kernel | min | max | avg |
| 2.6.16-hrt5 | 5 | 108 | 24 |
| 2.6.16-rt12 | 5 | 48 | 7 |

* Test case: clock\_nanosleep(TIME\_ABSTIME), Interval 500 micro seconds,. 100000 loops, 100% load.

Commandline: *cyclictest -t1 -p 80 -i 500 -n -l 100000*

|  |  |  |  |
| --- | --- | --- | --- |
| Kernel | min | max | avg |
| 2.6.16-hrt5 | 9 | 684 | 56 |
| 2.6.16-rt12 | 10 | 60 | 22 |

* Test case: POSIX interval timer, Interval 500 micro seconds,. 100000 loops, no load.

Commandline: *cyclictest -t1 -p 80 -i 500 -l 100000*

|  |  |  |  |
| --- | --- | --- | --- |
| Kernel | min | max | avg |
| 2.6.16-hrt5 | 8 | 119 | 22 |
| 2.6.16-rt12 | 12 | 78 | 16 |

* Test case: POSIX interval timer, Interval 500 micro seconds,. 100000 loops, 100% load.

Commandline: *cyclictest -t1 -p 80 -i 500 -l 100000*

|  |  |  |  |
| --- | --- | --- | --- |
| Kernel | min | max | avg |
| 2.6.16-hrt5 | 16 | 489 | 58 |
| 2.6.16-rt12 | 12 | 95 | 29 |

External Links

Current repo

Clone one of the following

* <git://git.kernel.org/pub/scm/utils/rt-tests/rt-tests.git>
* <https://git.kernel.org/pub/scm/utils/rt-tests/rt-tests.git>
* <https://kernel.googlesource.com/pub/scm/utils/rt-tests/rt-tests.git>

rt-tests tarballs

<https://www.kernel.org/pub/linux/utils/rt-tests/>

Mailing List

* [(linux-rt-users mailing list)](http://marc.info/?l=linux-rt-users)

FAQ

ps shows the wrong scheduling class SCHED\_OTHER

Each cyclictest-task consist of one or more threads. *ps -ce* shows only the main-process not the threads of the main-process. *ps -eLc | grep cyclic* shows the main-process an the containing threads with the correct scheduler class SCHED\_FIFO.

#>./cyclictest -t5 -p 80 -n -i 10000

#> ps -cLe | grep cyclic

4764 4764 TS 19 pts/1 00:00:01 cyclictest

4764 4765 FF 120 pts/1 00:00:00 cyclictest

4764 4766 FF 119 pts/1 00:00:00 cyclictest

4764 4767 FF 118 pts/1 00:00:00 cyclictest

4764 4768 FF 117 pts/1 00:00:00 cyclictest

4764 4769 FF 116 pts/1 00:00:00 cyclictest

chrt shows the wrong scheduling class SCHED\_OTHER

Don't use the PID of the main-process, but the pid of one of the threads from the main-process. The threads are shown with *ps -cLe | grep cyclic*.

#> chrt -p 4766

pid 4766's current scheduling policy: SCHED\_FIFO

pid 4766's current scheduling priority: 79

taskset for CPU affinity

taskset command is Written by Robert M. Love. SMP operating systems have choices when it comes to scheduling processes: a new or newly rescheduled process can run on any available cpu. However, while it shouldn't matter where a new process runs, an existing process should go back to the same cpu it was running on simply because the cpu may still be caching data that belongs to that process. This is particularly apt to be true if the process is a thread: the other threads in the same program are very likely to have cpu cache of interest to their brethren (though obviously this also diminishes the performance gain that might be seen from multithreading) . For these reasons, scheduling algorithms pay attention to cpu affinity and try to keep it constant.

It is possible to force a process to run only on a certain cpu. There are Linux system calls (sched\_setaffinity and sched\_getaffinity) and a command line "taskset".

lgs@f11#> taskset -c 3 top

lgs@f11#> taskset -p [pid]

Compile failure because numa.h can't be found

make

cc -D VERSION\_STRING=0.85 -c src/cyclictest/cyclictest.c -Wall -Wno-nonnull -O2 -DNUMA -D\_GNU\_SOURCE -Isrc/include

In file included from src/cyclictest/cyclictest.c:37:0:

src/cyclictest/rt\_numa.h:23:18: fatal error: numa.h: No such file or directory

compilation terminated.

make: \*\*\* [cyclictest.o] Error 1

Simply install your distribution's numa development package. On Fedora this is numactl-devel, so

su -c 'yum install numactl-devel'

This is only required for building. This will not affect the way the test runs on non-numa machines