RCU Concepts

The basic idea behind RCU (read-copy update) is to split destructive operations into two parts, one that prevents anyone from seeing the data item being destroyed, and one that actually carries out the destruction. A "grace period" must elapse between the two parts, and this grace period must be long enough that any readers accessing the item being deleted have since dropped their references. For example, an RCU-protected deletion from a linked list would first remove the item from the list, wait for a grace period to elapse, then free the element. See the :ref:`Documentation/RCU/listRCU.rst <list_rcu_doc>` for more information on using RCU with linked lists.

> **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\RCU\(linux-master) (Documentation) (RCU)rcu.rst`, line 6);** *backlink*
>
> Unknown interpreted text role "ref".

Frequently Asked Questions

- Why would anyone want to use RCU?

  The advantage of RCU's two-part approach is that RCU readers need not acquire any locks, perform any atomic instructions, write to shared memory, or (on CPUs other than Alpha) execute any memory barriers. The fact that these operations are quite expensive on modern CPUs is what gives RCU its performance advantages in read-mostly situations. The fact that RCU readers need not acquire locks can also greatly simplify deadlock-avoidance code.

- How can the updater tell when a grace period has completed if the RCU readers give no indication when they are done?

  Just as with spinlocks, RCU readers are not permitted to block, switch to user-mode execution, or enter the idle loop. Therefore, as soon as a CPU is seen passing through any of these three states, we know that that CPU has exited any previous RCU read-side critical sections. So, if we remove an item from a linked list, and then wait until all CPUs have switched context, executed in user mode, or executed in the idle loop, we can safely free up that item.

  Preemptible variants of RCU (CONFIG_PREEMPT_RCU) get the same effect, but require that the readers manipulate CPU-local counters. These counters allow limited types of blocking within RCU read-side critical sections. SRCU also uses CPU-local counters, and permits general blocking within RCU read-side critical sections. These variants of RCU detect grace periods by sampling these counters.

- If I am running on a uniprocessor kernel, which can only do one thing at a time, why should I wait for a grace period?

  See :ref:`Documentation/RCU/UP.rst <up_doc>` for more information.

  > **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\RCU\(linux-master) (Documentation) (RCU)rcu.rst`, line 53);** *backlink*
  >
  > Unknown interpreted text role "ref".

- How can I see where RCU is currently used in the Linux kernel?

  Search for "rcu_read_lock", "rcu_read_unlock", "call_rcu", "rcu_read_lock_bh", "rcu_read_unlock_bh", "srcu_read_lock", "srcu_read_unlock", "synchronize_rcu", "synchronize_net", "synchronize_srcu", and the other RCU primitives. Or grab one of the cscope databases from:

  (http://www.rdrop.com/users/paulmck/RCU/linuxusage/rculocktab.html).

- What guidelines should I follow when writing code that uses RCU?

  See the checklist.txt file in this directory.

- Why the name "RCU"?

  "RCU" stands for "read-copy update". :ref:`Documentation/RCU/listRCU.rst <list_rcu_doc>` has more information on where this name came from, search for "read-copy update" to find it.

  > **System Message: ERROR/3 (`D:\onboarding-resources\sample-onboarding-resources\linux-master\Documentation\RCU\(linux-master) (Documentation) (RCU)rcu.rst`, line 71);** *backlink*
  >
  > Unknown interpreted text role "ref".

- I hear that RCU is patented? What is with that?

  Yes, it is. There are several known patents related to RCU, search for the string "Patent" in Documentation/RCU/RTFP.txt to

find them. Of these, one was allowed to lapse by the assignee, and the others have been contributed to the Linux kernel under GPL. There are now also LGPL implementations of user-level RCU available (https://liburcu.org/).

- I hear that RCU needs work in order to support realtime kernels?

  Realtime-friendly RCU can be enabled via the CONFIG_PREEMPT_RCU kernel configuration parameter.

- Where can I find more information on RCU?

  See the Documentation/RCU/RTFP.txt file. Or point your browser at (http://www.rdrop.com/users/paulmck/RCU/).