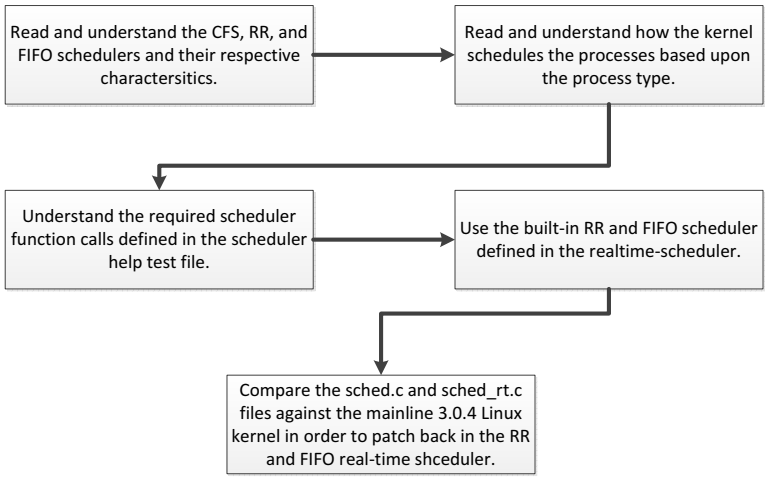
1 Design

To implement the Round Robin (RR) and First In First Out (FIFO) schedulers in the Linux kernel, the group decided to not "reinvent the wheel." Since the real-time scheduler already had a RR and FIFO completed. However, with the code provided for the project, these algorithms have been removed. to remedy this, we diffed between the "vanilla" 3.0.4 kernel obtained from kernel.org against the provided 3.0.4 kernel files. The difference was in kernelsched.c and kernelsched_rt.c. We created a patch that would add the missing functions and functionality back into the kernel. The basic flowchart is provided below.



2 Code

```
Index: kernel/sched.c
===================================================================
--- kernel/sched.c        (revision 5)
+++ kernel/sched.c        (revision 6)
@@ -122,9 +122,10 @@
  */
 #define RUNTIME_INF        ((u64)~0ULL)
```

```
-
 static inline int rt_policy(int policy)
 {
+        if (unlikely(policy == SCHED_FIFO || policy == SCHED_RR))
+                return 1;
         return 0;
 }

@@ -2048,8 +2049,8 @@
                  * much confusion -- but then, stop work should not
                  * rely on PI working anyway.
                  */
-
-
+                sched_setscheduler_nocheck(stop, SCHED_FIFO, &param);
+
                 stop->sched_class = &stop_sched_class;
         }

@@ -2838,7 +2839,10 @@
          * Revert to default priority/policy on fork if requested.
          */
         if (unlikely(p->sched_reset_on_fork)) {
-
+                if (p->policy == SCHED_FIFO || p->policy == SCHED_RR) {
+                        p->policy = SCHED_NORMAL;
+                        p->normal_prio = p->static_prio;
+                }

                 if (PRIO_TO_NICE(p->static_prio) < 0) {
                         p->static_prio = NICE_TO_PRIO(0);
@@ -5081,7 +5085,8 @@
                 reset_on_fork = !!(policy & SCHED_RESET_ON_FORK);
                 policy &= ~SCHED_RESET_ON_FORK;

-        if (policy != policy != SCHED_NORMAL && policy != SCHED_BATCH &&
+                if (policy != SCHED_FIFO && policy != SCHED_RR &&
+                                policy != SCHED_NORMAL && policy != SCHED_BATCH &&
                                 policy != SCHED_IDLE)
                         return -EINVAL;
         }
@@ -5730,9 +5735,10 @@
         int ret = -EINVAL;

         switch (policy) {
-
-
-
+        case SCHED_FIFO:
+        case SCHED_RR:
```

```
+                ret = MAX_USER_RT_PRIO-1;
+                break;
         case SCHED_NORMAL:
         case SCHED_BATCH:
         case SCHED_IDLE:
@@ -5754,9 +5760,10 @@
         int ret = -EINVAL;

         switch (policy) {
-
-
-
+        case SCHED_FIFO:
+        case SCHED_RR:
+                ret = 1;
+                break;
         case SCHED_NORMAL:
         case SCHED_BATCH:
         case SCHED_IDLE:
@@ -5786,7 +5793,6 @@
         if (pid < 0)
                 return -EINVAL;

-
         retval = -ESRCH;
         rcu_read_lock();
         p = find_process_by_pid(pid);
Index: kernel/sched_rt.c
===================================================================
--- kernel/sched_rt.c        (revision 5)
+++ kernel/sched_rt.c        (revision 6)
@@ -1766,8 +1766,14 @@
          * RR tasks need a special form of timeslice management.
          * FIFO tasks have no timeslices.
          */
+        if (p->policy != SCHED_RR)
+                return;

-
+        if (--p->rt.time_slice)
+                return;
+
+        p->rt.time_slice = DEF_TIMESLICE;
+
         /*
          * Requeue to the end of queue if we are not the only element
          * on the queue:
@@ -1793,7 +1799,9 @@
         /*
          * Time slice is 0 for SCHED_FIFO tasks
          */
```

```
-
+       if (task->policy == SCHED_RR)
+               return DEF_TIMESLICE;
+       else
                return 0;
 }
```