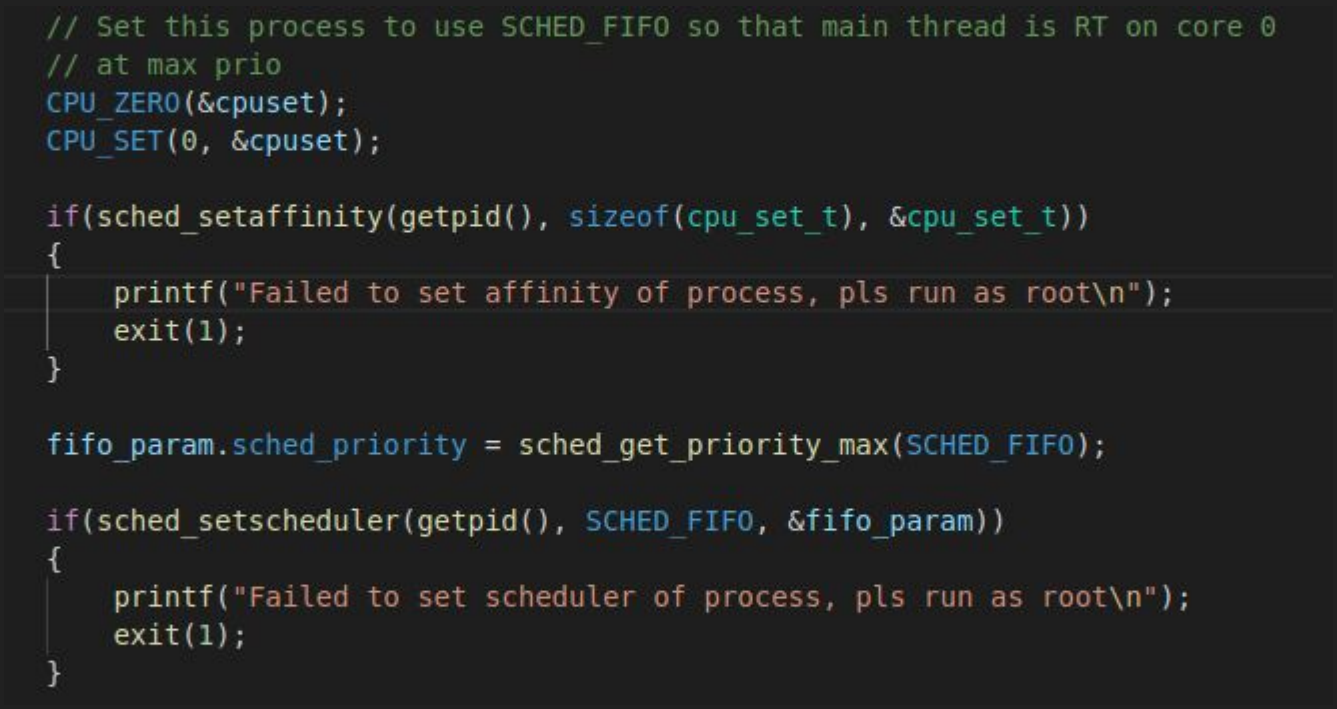
RT Threads Code Review

Tanner Johnson

Contents

- This powerpoint as substitute for video
  - Go over key sections of code
  - Runtime Behavior
  - Show output
- sum.c implementation of inc_dec
- Makefile to compile
- log.txt containing syslog output

Code Key Section 1) Set main thread to RT

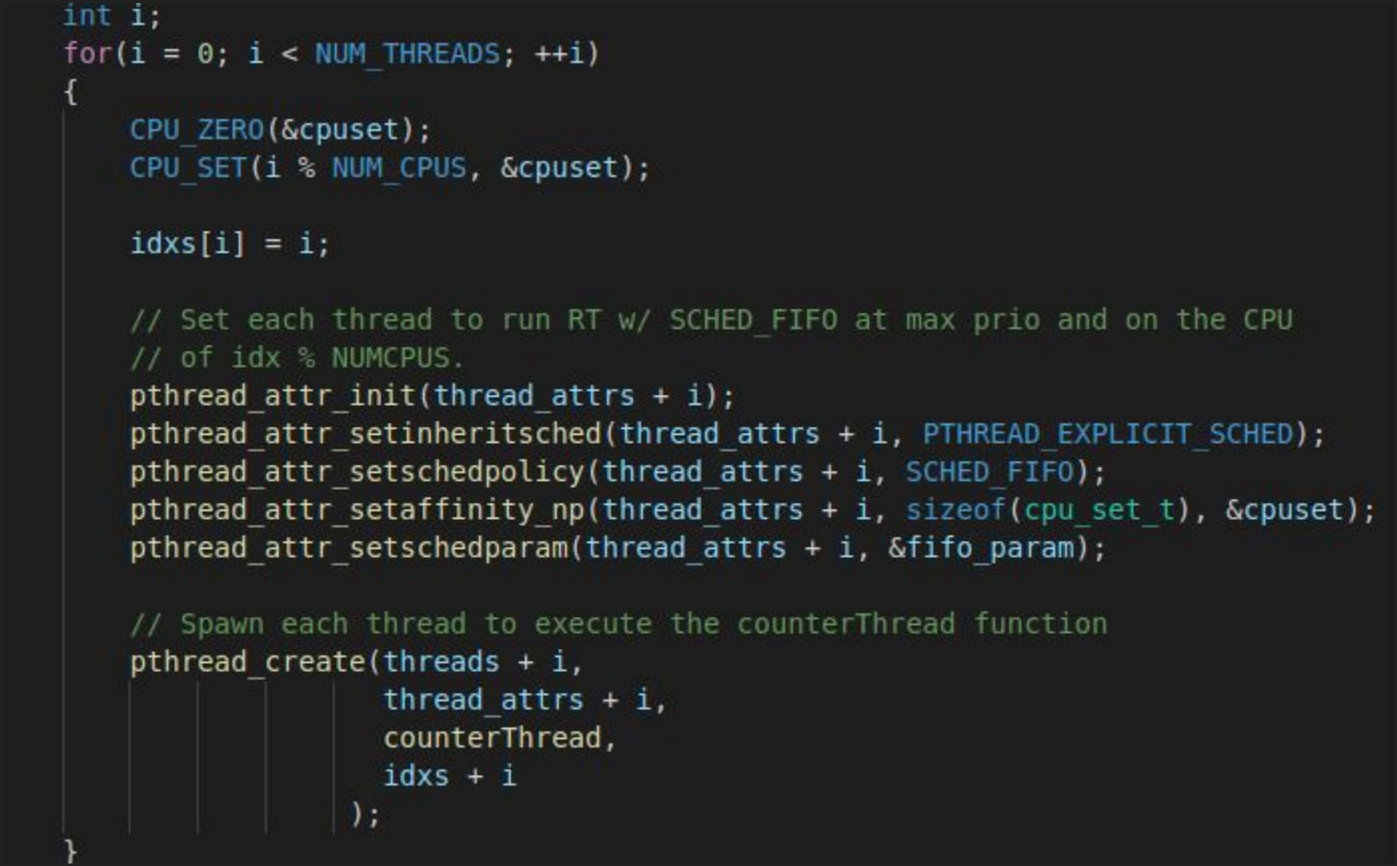


```c
// Set this process to use SCHED_FIFO so that main thread is RT on core 0
// at max prio
CPU_ZERO(&cpuset);
CPU_SET(0, &cpuset);

if(sched_setaffinity(getpid(), sizeof(cpu_set_t), &cpu_set_t))
{
    printf("Failed to set affinity of process, pls run as root\n");
    exit(1);
}

fifo_param.sched_priority = sched_get_priority_max(SCHED_FIFO);

if(sched_setscheduler(getpid(), SCHED_FIFO, &fifo_param))
{
    printf("Failed to set scheduler of process, pls run as root\n");
    exit(1);
}
```

Code Key Section 1) Set main thread to RT

- We use CPU_ macros to set a cpu bit mask to target CPU 0 only
- Use `sched_setaffinity()` to actually set the affinity of the main thread
- We set the scheduler priority to max with the following:
    - `fifo_param.sched_priority = sched_get_priority_max(SCHED_FIFO);`
- Then we actually use the `sched_setscheduler` system call to set the process to max priority using the real time SCHED_FIFO scheduler
- The two sched_* system calls require root access so we add error guards to catch that.

Code Key Section 2) Set Thread attributes and spawn



```c
int i;
for(i = 0; i < NUM_THREADS; ++i)
{
    CPU_ZERO(&cpuset);
    CPU_SET(i % NUM_CPUS, &cpuset);

    idxs[i] = i;

    // Set each thread to run RT w/ SCHED_FIFO at max prio and on the CPU
    // of idx % NUMCPUS.
    pthread_attr_init(thread_attrs + i);
    pthread_attr_setinheritsched(thread_attrs + i, PTHREAD_EXPLICIT_SCHED);
    pthread_attr_setschedpolicy(thread_attrs + i, SCHED_FIFO);
    pthread_attr_setaffinity_np(thread_attrs + i, sizeof(cpu_set_t), &cpuset);
    pthread_attr_setschedparam(thread_attrs + i, &fifo_param);

    // Spawn each thread to execute the counterThread function
    pthread_create(threads + i,
                   thread_attrs + i,
                   counterThread,
                   idxs + i
                  );
}
```

Code Key Section 2) Set Thread attributes and spawn

- For each thread we intent to spawn we set the CPU bit mask to the thread id mod 4 or the number of cpus so the threads get distribute as evenly as possible across the 4 cores.
- We use the pthread_attr_* setter calls to set each thread attribute to:
    - That thread id’s core assignment
    - Use SCHED_FIFO real time scheduling
    - Max priority
    - Set inherit sched to explicit so it uses are explicit sched attributes we set
- Finally we launch all the threads using pthread_create()

Other Code

- In previous assignments we went over the pthread_join and pthread_create system calls thus we omit that discussion here
- Similarly we went over the syslog system call in previous assignments
- Finally, the actually counter thread is pretty straight forward and was covered in peer graded assignment 1 and so we omit a discussion.

RT FIFO behavior

- The main thread and all 128 counter threads are of max prio
- Thus all threads occupy the same FIFO queue and are in the order: {main,0,1,...,127}.
- Since the threads target different CPUs, threads targeting the same CPU will be ran in order and preempted only when the previous thread finishes
- As an example, CPU 1 will be occupied by thread 1 until it finishes.
- Thread 5 will take over, followed by {5,9, …, 125}.
- This behavior is analogous for CPU 2 and 3.
- CPU 0 is slightly different because it shares execution with main that has a blocking system call.

RT FIFO behavior (CPU 0)

- The execution order for CPU 0 is as follows, {main, 0,4,8, …, 124, main}.
- The first instance of main executing is when main is calling pthread_create.
- It will hold the CPU until it finishes creating all threads and will yield when it hits the first pthread_join, waiting for thread 0 to die
- Thread 0 takes over the CPU, and finishes executing.
- At this point main because runnable and is placed at the end of the FIFO.
- Threads {4,8,...124} finish executing
- Main takes back CPU 0 to finish processing the pthread_join calls

Output

- See contained log.txt for output