4.2 实验 4-2：内核线程

1．实验目的

了解和熟悉 Linux 内核中是如何创建内核线程的。

2．实验要求

1）写一个内核模块，创建一组内核线程，每个 CPU 一个内核线程。
2）在每个内核线程中，打印当前 CPU 的状态，比如 ARM64 通用寄存器的值。
3）在每个内核线程中，打印当前进程的优先级等信息。

3．实验步骤

下面是本实验的实验步骤。
进入本实验的参考代码目录进行交叉编译。

```
cd
/home/lab466/runninglinuxkernel_4.0/rlk_lab/rlk_basic/chapter_8/lab2_kthread
export ARCH=arm
export CROSS_COMPILE=arm-linux-gnueabi-
    make BASEINCLUDE=/home/lab466/runninglinuxkernel_4.0
```

然后把 ko 内核模块拷贝到 runninglinuxkernel_4.0/kmodules 目录下面。

```
    cp kthread_test.ko /home/lab466/runninglinuxkernel_4.0/kmodules
```

启动 QEMU+runninglinuxkernel。最好另外开一个窗口，运行：

```
    sudo su
cd /home/lab466/runninglinuxkernel_4.0
    sh run.sh arm32
```

进入本实验的参考代码。

```
    # cd /mnt
```

安装本实验的内核模块。

```
benshushu:lab2_kthread# insmod kthread_test.ko
[13427.799305] Loading module cpu=0.
[13427.817282] About to wake up and run the thread for cpu=0
[13427.820939] Staring thread for cpu 0
[13427.821134] on cpu=2.
[13427.824743] SLEEP in Thread Function  cpu=0.
[13427.831980] About to wake up and run the thread for cpu=1
[13427.834020] Staring thread for cpu 1
```

```
[13427.834056] on cpu=2.
[13427.838384] SLEEP in Thread Function  cpu=1.
[13427.848019] About to wake up and run the thread for cpu=2
[13427.848956] Staring thread for cpu 2
[13427.848988] on cpu=3.
[13427.849403] SLEEP in Thread Function  cpu=2.
[13427.856697] About to wake up and run the thread for cpu=3
[13427.858520] Staring thread for cpu 3
[13427.858551] on cpu=3.
[13427.867324] SLEEP in Thread Function  cpu=3.
benshushu:lab2_kthread# [13429.851394] msleep over in Thread Function cpu=1.
[13429.851434] msleep over in Thread Function cpu=0.
[13429.851520] running cpu=0.
[13429.851823] running cpu=1.
[13429.864449] spsr:0x20000005, sp:0x20b07d40, el=1
[13429.864563] kdemo/0 pid:1820, nice:0 prio:120 static_prio:120
normal_prio:120
[13429.864613] SLEEP in Thread Function  cpu=0.
[13429.869668] spsr:0x60000005, sp:0x250d7d40, el=1
```

后续，删除设备和模块 rmmod kthread_test

4．实验代码

```
#include <linux/module.h>
#include <linux/init.h>
#include <linux/module.h>
#include <linux/kthread.h>
#include <linux/delay.h>

static struct task_struct *tsk[NR_CPUS];

static void show_reg(void)
{
      unsigned int spsr, sp, el;

      asm("mrs %0, spsr_el1" : "=r" (spsr) : : "cc");
      asm("mov %0, sp" : "=r" (sp) : : "cc");
      asm("mrs %0, CurrentEL" : "=r" (el) : : "cc");

      printk("spsr:0x%x, sp:0x%x, el=%d\n", spsr, sp, el >> 2);
}

static void show_prio(void)
{
      struct task_struct *task = current;

      printk("%s pid:%d, nice:%d prio:%d static_prio:%d normal_prio:%d\n",
               task->comm, task->pid,
               PRIO_TO_NICE(task->static_prio),
               task->prio, task->static_prio,
               task->normal_prio);
}

static void print_cpu(char *s)
{
      preempt_disable();
      pr_info("%s cpu=%d.\n", s, smp_processor_id());
      preempt_enable();
}
```

```
static int thread_fun(void *t)
{
      do {
           print_cpu("SLEEP in Thread Function ");
           msleep_interruptible(2000);
           print_cpu("msleep over in Thread Function");
           print_cpu("running");
           show_reg();
           show_prio();
      } while (!kthread_should_stop());
      return 0;
}

static int __init my_init(void)
{
      int i;
      print_cpu("Loading module");
      for_each_online_cpu(i) {
           tsk[i] = kthread_create(thread_fun, NULL, "kdemo/%d", i);
           if (!tsk[i]) {
                pr_info("Failed to generate a kernel thread\n");
                return -1;
           }
           kthread_bind(tsk[i], i);
           pr_info("About to wake up and run the thread for cpu=%d\n", i);
           wake_up_process(tsk[i]);
           pr_info("Staring thread for cpu %d", i);
           print_cpu("on");
      }
      return 0;
}

static void __exit my_exit(void)
{
      int i;
      for_each_online_cpu(i) {
           pr_info(" Kill Thread %d", i);
           kthread_stop(tsk[i]);
           print_cpu("Kill was done on ");
      }
}

module_init(my_init);
module_exit(my_exit);

MODULE_AUTHOR("Ben ShuShu");
MODULE_LICENSE("GPL v2");
```