

- [Linux 线程竞争范围及线程调度](#)
  - [1. 实验目的](#)
  - [2. 实验内容](#)
  - [3. 例子代码](#)
    - [3.1. 线程竞争范围代码 \(thread\\_scope.c\)](#)
    - [3.2. 线程调度确认源码 \(thread\\_sched.c\)](#)

## Linux 线程竞争范围及线程调度

### 1. 实验目的

1. 了解和掌握 Linux 系统的线程竞争范围
2. 了解和掌握 Linux 系统提供的线程调度算法

### 2. 实验内容

1. 编译运行 thread\_scope.c 程序，确认 Linux 系统提供的线程竞争范围，利用代码加以分析
2. 编译运行 thread\_sched.c 程序，确认 Linux 系统提供的线程调度算法，利用代码加以分析

### 3. 例子代码

#### 3.1. 线程竞争范围代码 (thread\_scope.c)

```
1  #include <pthread.h>
2  #include <stdio.h>
3  #define NUM_THREADS 5
4  void * runner(void * param);
5
6  int main (int argc, char * argv[]){
7      int i, scope;
8      pthread_t tid[NUM_THREADS];
9      pthread_attr_t attr;
10     pthread_attr_init(&attr);
11     if (pthread_attr_getscope(&attr, &scope) != 0)
12         fprintf(stderr, "unable to get scheduling scope\n");
13     else {
14         if (scope == PTHREAD_SCOPE_PROCESS)
15             printf("PTHREAD_SCOPE_PROCESS\n");
16         else if (scope == PTHREAD_SCOPE_SYSTEM)
17             printf("PTHREAD_SCOPE_SYSTEM\n");
18         else
19             fprintf(stderr, "Illegal scope value.\n");
20     }
21     pthread_attr_setscope(&attr, PTHREAD_SCOPE_PROCESS);
22
23     if (pthread_attr_getscope(&attr, &scope) != 0)
24         fprintf(stderr, "unable to get scheduling scope\n");
25     else {
26         if (scope == PTHREAD_SCOPE_PROCESS)
27             printf("PTHREAD_SCOPE_PROCESS\n");
28         else if (scope == PTHREAD_SCOPE_SYSTEM)
29             printf("PTHREAD_SCOPE_SYSTEM\n");
30         else
31             fprintf(stderr, "Illegal scope value.\n");
32     }
33
34     for (i = 0; i < NUM_THREADS; i++)
35         pthread_create(&tid[i], &attr, runner, NULL);
36     for (i = 0; i < NUM_THREADS; i++)
37         pthread_join(tid[i], NULL);
38 }
39
40 void * runner(void * param)
41 {
42     pthread_exit(0);
43 }
```

#### 3.2. 线程调度确认源码 (thread\_sched.c)

```

1  #include <unistd.h>
2  #include <pthread.h>
3  #include <sched.h>
4  #include <stdio.h>
5
6  static int get_thread_policy(pthread_attr_t attr)
7  {
8      int policy;
9      pthread_attr_getschedpolicy(&attr, &policy);
10     switch(policy)
11     {
12         case SCHED_FIFO:
13             printf(" policy = SCHED_FIFO\n");
14             break;
15         case SCHED_RR:
16             printf(" policy = SCHED_RR\n");
17             break;
18         case SCHED_OTHER:
19             printf(" policy = SCHED_OTHER\n");
20             break;
21         default:
22             printf(" policy = UNKOWN\n");
23             break;
24     }
25     return policy;
26 }
27
28
29 static void set_thread_policy(pthread_attr_t attr, int policy)
30 {
31     pthread_attr_setschedpolicy(&attr, policy);
32     get_thread_policy(attr);
33 }
34
35 static void show_thread_priority(pthread_attr_t attr, int policy)
36 {
37     int priority = sched_get_priority_max(policy);
38     printf(" max priority = %d, ", priority);
39     priority = sched_get_priority_min(policy);
40     printf(" min priority = %d\n", priority);
41 }
42
43 static int get_thread_priority(pthread_attr_t attr)
44 {
45     struct sched_param param;
46     pthread_attr_getschedparam(&attr, &param);
47     printf(" priority = %d\n", param.sched_priority);
48     return param.sched_priority;
49 }
50
51 int main (void)
52 {
53     pthread_attr_t attr;
54     struct sched_param sched;
55     pthread_attr_init(&attr);
56
57     printf("- Show Current Policy:");
58     int policy = get_thread_policy(attr);
59     printf("- Show current configuration of priority:");
60     show_thread_priority(attr, policy);
61     printf("- Show SCHED_FIFO of priority:");
62     show_thread_priority(attr, SCHED_FIFO);
63     printf("- Show SCHED_RR of priority:");
64     show_thread_priority(attr, SCHED_RR);
65
66     printf("- Show priority of current thread:");
67     int priority = get_thread_priority(attr);
68     printf("\n -SET THREAD POLICY\n");
69     printf("- SET SCHED_FIFO policy:");
70     set_thread_policy(attr, SCHED_FIFO);
71     printf("- SET SCHED_RR policy:");
72     set_thread_policy(attr, SCHED_RR);
73     printf("- Restore current policy:");
74     set_thread_policy(attr, policy);
75     pthread_attr_destroy(&attr);
76     return 0;
77 }

```