CSCI-3150: Introduction to Operating Systems

Lab Six: Condition Variable

Deadline: Dec. 12, 2021 23:59

1 Background

In this lab, you will learn how to use condition variables in pthread library to implement a wakeup-enabled multi-thread program. You will learn the following functions:

1. `pthread_cond_init()`: function to create a condition variable for wakeup-enabled access to shared resources.
2. `pthread_cond_wait()`: function to wait for a condition variable.
3. `pthread_cond_signal()`: function to wake up a waiting thread.

2 Condition Variable Exercise

In this exercise, you are required to create some child threads and use condition variables to achieve the same behaviour as `pthread_join` and `pthread_exit`. The following program creates 5 (`N_THREADS`) and each of them prints a message before they call `thr_exit`. The main function calls `thr_join` to wait for the threads to terminate before it returns. You should fill in all `/*YOUR CODE HERE */` with your own code to complete the program.

```c
#include <stdio.h>
#include <pthread.h>

#define N_THREADS 5

static int exited;

static pthread_mutex_t mutex = PTHREAD_MUTEX_INITIALIZER;
static pthread_cond_t cond = PTHREAD_COND_INITIALIZER;

void thr_exit() {
  pthread_mutex_lock(&mutex);

  /* YOUR CODE HERE */

  pthread_mutex_unlock(&mutex);
}

void thr_join() {
  pthread_mutex_lock(&mutex);
```

```
21
22    while (exited < N_THREADS) /* YOUR CODE HERE */
23
24    pthread_mutex_unlock(&mutex);
25  }
26
27  void *child_func(void *arg) {
28    int thr_id = /* YOUR CODE HERE */;
29    printf("child %d created and exited\n", thr_id);
30    thr_exit();
31    return NULL;
32  }
33
34  int main() {
35    pthread_t p[N_THREADS];
36    int thr_idx[N_THREADS];
37    void *arg;
38    int i;
39
40    exited = /* YOUR CODE HERE */;
41
42    puts("parent: begin");
43
44    for (i = 0; i < N_THREADS; i++) {
45      thr_idx[i] = i;
46      arg = &thr_idx[i];
47
48      pthread_create(/* YOUR CODE HERE */);
49    }
50
51    thr_join();
52
53    puts("parent: end");
54
55    return 0;
56  }
```

You can find the program code in `exercise.c`. To compile and run the program, use:

```
1  $ gcc -o exercise exercise.c -lpthread
2  $ ./exercise
```

Sample output:

```
parent: begin
child 0 created and exited
```

```
child 3 created and exited
child 1 created and exited
child 4 created and exited
child 2 created and exited
parent: end
```

Note: Because the threads are scheduled to run in arbitrary order, the order of child output can be different from the sample output. **If you correctly implement `thr_join` and `thr_exit`, you should see "`parent:  end`" at the end of the output, after all child outputs**. Also, it is possible that all child threads run and finish before the main thread call `thr_join` so you may get the sample output even if the two functions are not correctly implemented. You should run the program multiple times to detect potential bugs.

3 Submission

You ONLY need to submit the completed exercise.c to Blackboard. If you have any questions about this assignment, please send a email to jinxue@cse.cuhk.edu.hk.