CSCI315 – Operating Systems Design

Department of Computer Science
Bucknell University

Pthread Construct

Ch 4.4-4.6

This set of notes is based on notes from the textbook authors, as well as L. Felipe Perrone, Joshua Stough, and other instructors.
Pthread programming information is also from the tutorial by Blaise Barney from Lawrence Livermore National Lab. Xiannong Meng, Fall 2021.

An Example of Shared Data

A global variable

```
/* COMPILE WITH: gcc trd-share.c -lpthread -o trd-share */
#include <stdio.h>
#include <pthread.h>
#define NUM_THREADS  5
void *work(void *);   /* thread routine */
int v = 0;                    /* global variable, shared */
int  main(int argc, char *argv[]) {
      int i;
      pthread_t tid[NUM_THREADS];      /* array of thread IDs */
      for ( i = 0; i < NUM_THREADS; i++)
                  pthread_create(&tid[i], NULL, work, NULL);
      for ( i = 0; i < NUM_THREADS; i++)
                  pthread_join(tid[i], NULL);
      printf("main() reporting that all %d threads have terminated\n", i);
      printf("v should be %d, it is %d\n", NUM_THREADS, v);
      return (0);
}  /* main */
```

Call a function without parameters

http://www.eg.bucknell.edu/~cs315/F2021/meng/code/thread/trd-share.c

The Worker Function and Result

```
void * work(void *arg)   {
    v ++;    // 'v' is a global variable
    return (NULL);
}
```

```
[xmeng@polaris thread]$ ./trd-share
main() reporting that all 5 threads have terminated
v should be 5, it is 5
[xmeng@polaris thread]$
```

Everything seems working fine. However if one increases the number of threads to a larger value, e.g., 5000, we may see something incorrect.

http://www.eg.bucknell.edu/~cs315/F2021/meng/code/thread/trd-share.c

There May Be A Problem …

```
#define  NUM_THREADS  5000   // everything else is the same
```

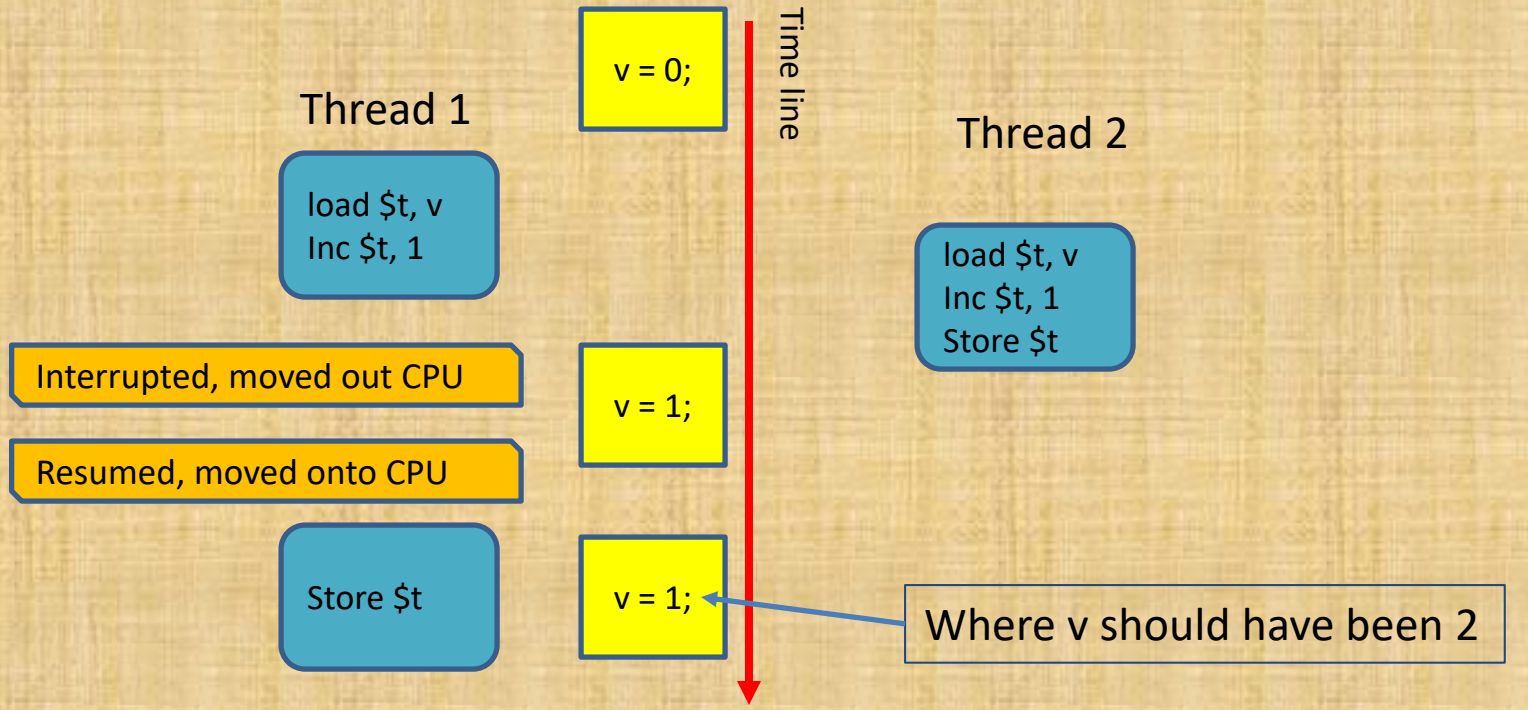
```
[xmeng@polaris thread]$ ./trd-share
main() reporting that all 5000 threads have terminated
v should be 5000, it is 4998
[xmeng@polaris thread]$
```

Who stole the two counts from me?!!

Why?

- How do we update the value of a variable?
- We learned that in CSCI 206
  - lw t0, 0(s1) # load memory content at s1 to t0
  - addi t0, t0, 1 # increment t0 by 1
  - sw t0, 0(s1) # store content in t0 to memory at s1
- In a multi-thread and multi-process environment, before finishing all three steps, a thread/process may be interrupted and moved out of the CPU and memory, leaving a inconsistent value for a shared variable

Two Threads/Processes Update the Same Variable at the Same Time



How To Prevent Problems of This Kind?

- The phenomenon in the previous slide is called “race condition,” --- *the value of a variable depends on the order of execution*.
- Threads and processes need coordination. We will discuss the topic in greater detail Chapter 5.

One More Example

```
/* COMPILE WITH: gcc trd-sleep.c -lpthread -o trd-sleep */
#include <stdio.h>
#include <pthread.h>
#define NUM_THREADS 5
int SLEEP_TIME = 3;

void *sleeping(void *); /* forward declaration to thread routine */

int main(int argc, char *argv[]) {
     int i;
     pthread_t tid[NUM_THREADS]; /* array of thread IDs */
     for ( i = 0; i < NUM_THREADS; i++)
       pthread_create(&tid[i], NULL, sleeping,(void *)&SLEEP_TIME);

     for ( i = 0; i < NUM_THREADS; i++)
       pthread_join(tid[i], NULL);

     printf("main() reporting that all %d threads have terminated\n", i);
     return (0);
} /* main */
```

http://www.eg.bucknell.edu/~cs315/F2021/meng/code/thread/trd-sleep.c

The sleeping() Function

```
void * sleeping(void *arg) {
  int sleep_time = *((int*)arg);  // cast, then dereference
  printf("thread %ld sleeping %d seconds ...\n",
           pthread_self(), sleep_time);
  sleep(sleep_time);
  printf("\nthread %ld awakening\n", pthread_self());
  return (NULL);
}
```

http://www.eg.bucknell.edu/~cs315/F2021/meng/code/thread/trd-sleep.c

How To Pass Parameter(s) to Worker

- In the two examples we have, one doesn't have any parameters to the worker function (*work()* where v is incremented by 1); the other has one parameter (*sleeping()*) to indicate the number of seconds to sleep.
- In general, the one parameter to a thread worker function is the address where the parameters should reside.
- What if we need multiple parameters?

Building Multi-Parameter Block

- What to use? C structures!
- Steps to take
  - Define a C structure that can hold multiple pieces of information
  - Fill in the parameters
  - Pass the address of the structure to the worker function
  - Extract return parameters, if any, from the pointer to the structure

http://www.eg.bucknell.edu/~cs315/F2021/meng/code/thread/trd-param.c

Example of Parameters

```
struct param_t {    /* a sample parameter structure */
    int id;         /* id and name are in params */
    char * name;
    int result;     /* result is out param */
};
```

Define parameter structures

```
for (k = 0; k < NUM_THREADS; k ++) {
    param[k].id = k;
    param[k].name = (char*)malloc(32);
    strcpy(param[k].name, “hello”);
};
```

Preparing parameter in structure

```
for (k = 0; k < NUM_THREADS; k ++) {
    pthread_create(&tid[k], NULL, work, &(param[k]));
};
```

Create thread with parameters

Compared to *pthread_create(&tid[k], NULL, work, NULL);*

Access and Return Parameters

```
void * work(void * arg) {
    v ++;   // v is a global variable
    ((struct param_t *)arg)->result = v; // set output parameter
    return NULL;
}
```

The *work()* function has access to the parameters, so is the calling function.

```
/* in main() after thread execution */
for (k = 0; k < NUM_THREADS; k ++) {
    printf("thread %d output value %d\n",
                param[k].id, param[k].result);
}
```

Access parameters from calling function.

Execution Results

```
[xmeng@polaris thread]$ ./trd-param
main() reporting that all 5 threads have terminated
v should be 5, it is 5
output parameters in each thread
thread 0 output 1
thread 1 output 3
thread 2 output 2
thread 3 output 4
thread 4 output 5
[xmeng@polaris thread]$
```

Program output

Note that the values and IDs in this example are out of order, not by design. Why?