Computer Operating Systems, Practice Session 4
Threads

Resul Tugay (tugayr@itu.edu.tr)

March 01, 2017

İTÜ

Today

Computer Operating Systems, PS 4
Thread Creation and Termination
Joining Threads
Using Global Variables in Threads

Thread Creation

```
#include <pthread.h>

int pthread_create(pthread_t *thread, const pthread_attr_t *attr, void
*(*start_routine)(void*), void *arg);
```

```
pthread_t *thread            : Pointer to the thread to be created
const pthread_attr_t *attr   : Pointer to attributes of the thread to be created
void *(*start_routine)(void*) : Pointer to the routine that will start the thread
void *arg                    : Pointer to the arguments for the start routine
```

returns 0 on success and an error number on failure

Example Program 1

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

void* print_message_function(void *ptr){
  char *message;
  // interpreting as char *
  message = (char *) ptr;
  printf("\n %s \n", message);
  // terminating the thread
  pthread_exit(NULL);
}

int main(){
  pthread_t thread1, thread2, thread3;
  char *message1 = "Hello";
  char *message2 = "World";
  char *message3 = "!...";
```

Example Program 1

```c
// creating 3 threads with start routine as print_message_function
// and start routine arguments as message1, message2 and message3
if(pthread_create(&thread1,NULL,print_message_function,(void *)
    message1)){
        fprintf(stderr,"pthread_create failure\n");
        exit(-1);
}
if(pthread_create(&thread2,NULL,print_message_function,(void *)
    message2)){
        fprintf(stderr,"pthread_create failure\n");
        exit(-1);
}
if(pthread_create(&thread3,NULL,print_message_function,(void *)
    message3)){
        fprintf(stderr,"pthread_create failure\n");
        exit(-1);
}
// to block main to support its threads until they terminate
pthread_exit(NULL);
}
```

Compiling a Program Including Thread/s

- Source File: `source.c`
- Executable File: `output`
- These applications should be linked with thread library. Sample, proper compilation:
  `gcc -pthread source.c -o output`

Output of the Example Program 1

```
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ gcc -pthread
 Example1.c -o output
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ ./output

 !...

 World

 Hello
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ █
```

Example Program 2

```
1  #include <pthread.h>
2  #include <stdio.h>
3  #include <stdlib.h>
4  #include <math.h>
5  #define NUM_THREADS 4
6
7  void *BusyWork(void *t){
8      int i;
9      long tid;
10     double result=0.0;
11     tid = (long)t;
12     long exit_status = 10*tid;
13     printf("Thread %ld starting...\n", tid);
14     for (i=0; i<1000000; i++){
15         result = result + sin(i) * tan(i);
16     }
17     printf("Thread %ld done. Result = %e\n", tid, result);
18     pthread_exit((void*) exit_status);
19 }
```

Barney B. (2013). POSIX Threads Programming. Retrieved March 03, 2014 (and modified) from https://computing.llnl.gov/tutorials/pthreads/

Example Program 2

```
1  int main (int argc, char *argv[]){
2    pthread_t thread[NUM_THREADS];
3    pthread_attr_t attr;
4    int rc;
5    long t;
6    void *status;
7    // Initialize and set thread detach state attribute
8    // Only threads that are created as joinable can be joined
9    // Threads created as PTHREAD_CREATE_DETACHED, cannot be joined
10   pthread_attr_init(&attr);
11   pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);
12   for(t=0; t<NUM_THREADS; t++) {
13     printf("Main: creating thread %ld\n", t);
14     // creating thread t
15     rc = pthread_create(&thread[t], &attr, BusyWork, (void *)t);
16     if (rc) {
17       printf("ERROR; return code from pthread_create() is %d\n", rc);
18       exit(-1);
19     }
20   }
```

Example Program 2

```
1   // Free library resources used by the attribute
2   pthread_attr_destroy(&attr);
3   // Join operation is used for synchronization between threads by
4   // blocking the calling thread until the specified thread (with
5   // given threadid) terminates. status is the exit status of the
6   // target thread (from pthread_exit) or PTHREAD_CANCELED if the
7   // target thread was canceled.
8   for(t=0; t<NUM_THREADS; t++) {
9       rc = pthread_join(thread[t], &status);
10      if (rc) {
11          printf("ERROR; return code from pthread_join() is %d\n", rc);
12          exit(-1);
13      }
14      printf("Main: completed join with thread %ld having a status of
        %ld\n",t,(long)status);
15  }
16  printf("Main: program completed. Exiting.\n");
17  // to block main to support its threads until they terminate
18  pthread_exit(NULL);
19 }
```

Output of the Example Program 2

```
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ gcc Example2.c -lm
  -pthread -o Example2
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ ./Example2
Main: creating thread 0
Main: creating thread 1
Main: creating thread 2
Main: creating thread 3
Thread 3 starting...
Thread 2 starting...
Thread 1 starting...
Thread 0 starting...
Thread 3 done. Result = -3.153838e+06
Thread 0 done. Result = -3.153838e+06
Main: completed join with thread 0 having a status of 0
Thread 2 done. Result = -3.153838e+06
Thread 1 done. Result = -3.153838e+06
Main: completed join with thread 1 having a status of 10
Main: completed join with thread 2 having a status of 20
Main: completed join with thread 3 having a status of 30
Main: program completed. Exiting.
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ ▮
```

Example Program 3

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

int myglobal;

void* thread_function(void *arg){
    int i,j;
    // changing the value of myglobal in thread_function
    for(i=0;i<20;i++){
        //myglobal++;
        j=myglobal;
        j=j+1;
        myglobal=j;
        printf(".");
        // to force writing all user-space buffered data to stdout
        fflush(stdout);
        sleep(1);
    }
    pthread_exit(NULL);
}
```

Example Program 3

```c
int main(void){
  pthread_t mythread;
  int i;
  myglobal=0;
  // creating a thread using thread_function as the start routine
  if(pthread_create(&mythread,NULL,thread_function,NULL)){
    printf("error creating thread");
    abort();
  }
  // changing the value of myglobal in main()
  for(i=0;i<20;i++){
    myglobal = myglobal+1;
    printf("o");
    // to force writing all user-space buffered data to stdout
    fflush(stdout);
    sleep(1);
  }
  printf("\nmyglobal equals %d\n",myglobal);
  // to block main to support its threads until they terminate
  pthread_exit(NULL);
}
```

Output of the Example Program 3

```
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ gcc Example3.c -pthread
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ ./a.out
o.o.o.o.o.o.o.o.o.o.o..oo.o..oo..o.oo..oo.
myglobal equals 40
musty@musty-VirtualBox:/media/sf_virtualbox_shared_folder$ 
```