Computer Operating Systems, Practice Session 4

Threads

Mustafa Ersen (ersenm@itu.edu.tr)

Istanbul Technical University
34469 Maslak, İstanbul

05 March 2014

Today

Computer Operating Systems, PS 4

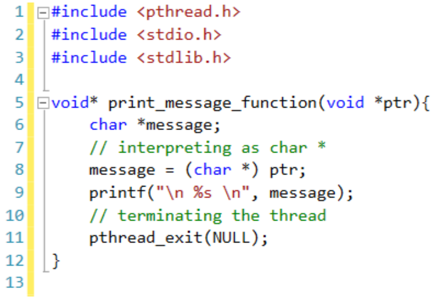
- Thread Creation and Termination
- Joining Threads
- Using Global Variables in Threads

Thread Creation

```
#include <pthread.h>

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

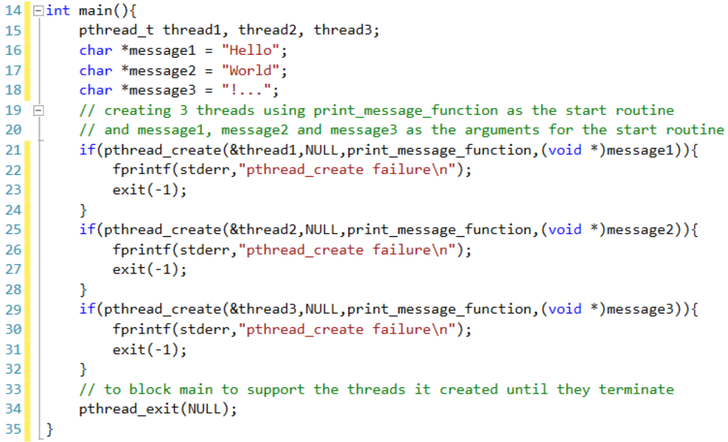
`pthread_t *thread`	: Pointer to the thread to be created
`const pthread_attr_t *attr`	: Pointer to attributes of the thread to be created
`void *(*start_routine)(void*)`	: Pointer to the routine that will start the thread
`void *arg`	: Pointer to the arguments for the start routine

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);
}

```

Example Program 1



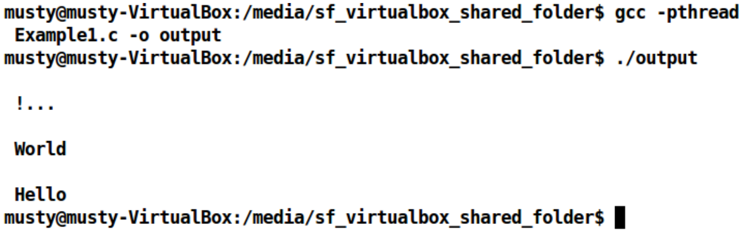
```c
int main(){
    pthread_t thread1, thread2, thread3;
    char *message1 = "Hello";
    char *message2 = "World";
    char *message3 = "!...";
    // creating 3 threads using print_message_function as the start routine
    // and message1, message2 and message3 as the arguments for the start routine
    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 the threads it created 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:

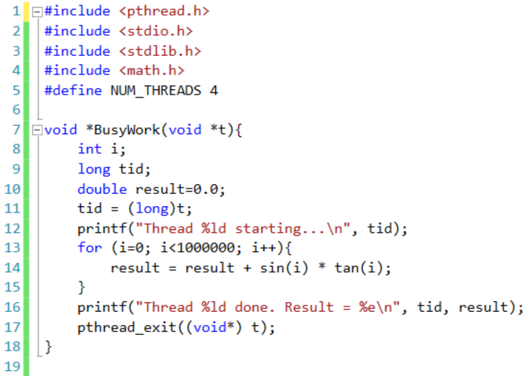
```
gcc -pthread source.c -o output
```

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$ 
```

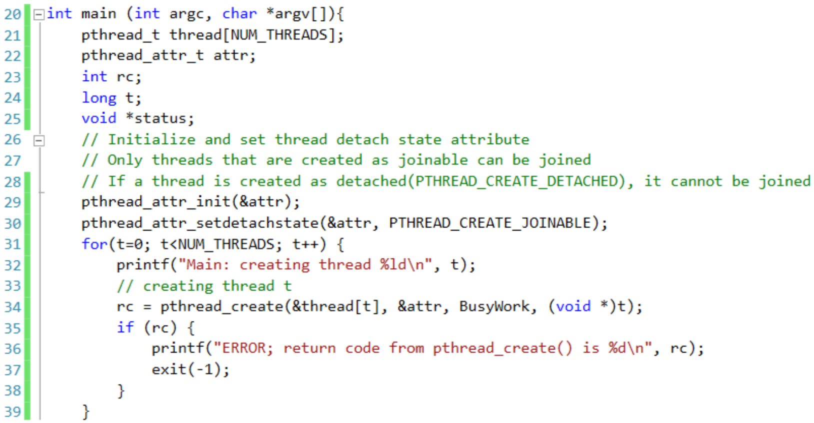
Example Program 2



```
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#define NUM_THREADS 4

void *BusyWork(void *t){
    int i;
    long tid;
    double result=0.0;
    tid = (long)t;
    printf("Thread %ld starting...\n", tid);
    for (i=0; i<1000000; i++){
        result = result + sin(i) * tan(i);
    }
    printf("Thread %ld done. Result = %e\n", tid, result);
    pthread_exit((void*) t);
}

```

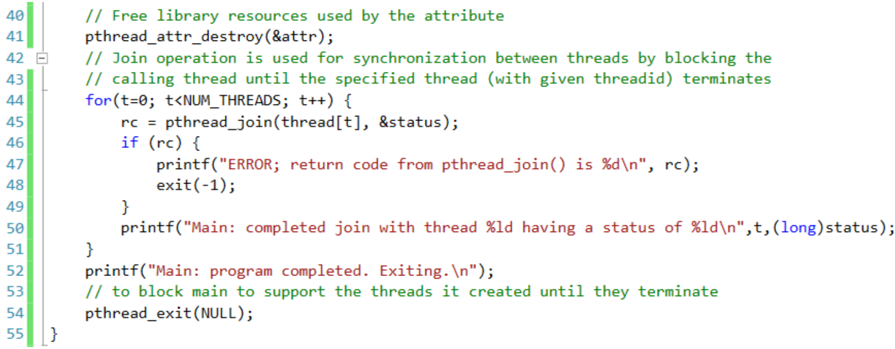
Barney B. (2013). POSIX Threads Programming. Retrieved March 03, 2014, from https://computing.llnl.gov/tutorials/pthreads/

Example Program 2



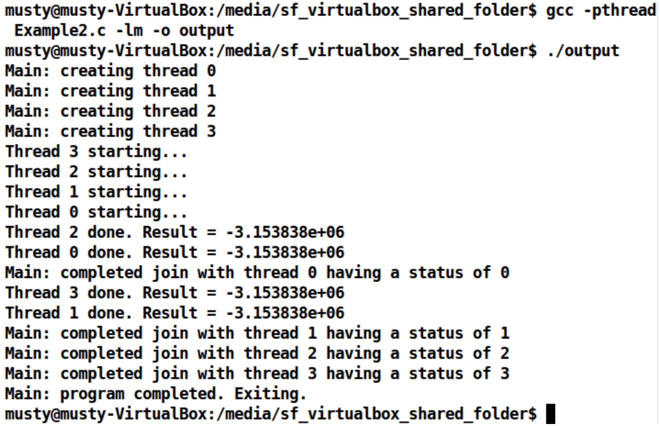
```
int main (int argc, char *argv[]){
    pthread_t thread[NUM_THREADS];
    pthread_attr_t attr;
    int rc;
    long t;
    void *status;
    // Initialize and set thread detach state attribute
    // Only threads that are created as joinable can be joined
    // If a thread is created as detached(PTHREAD_CREATE_DETACHED), it cannot be joined
    pthread_attr_init(&attr);
    pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);
    for(t=0; t<NUM_THREADS; t++) {
        printf("Main: creating thread %ld\n", t);
        // creating thread t
        rc = pthread_create(&thread[t], &attr, BusyWork, (void *)t);
        if (rc) {
            printf("ERROR; return code from pthread_create() is %d\n", rc);
            exit(-1);
        }
    }
```

Example Program 2



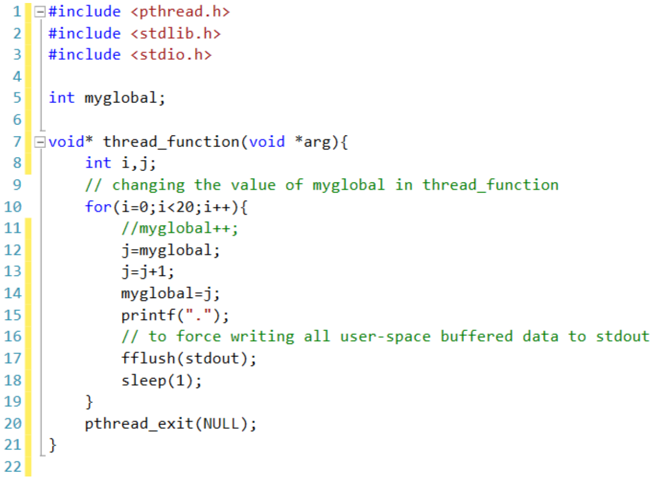
```c
    // Free library resources used by the attribute
    pthread_attr_destroy(&attr);
    // Join operation is used for synchronization between threads by blocking the
    // calling thread until the specified thread (with given threadid) terminates
    for(t=0; t<NUM_THREADS; t++) {
        rc = pthread_join(thread[t], &status);
        if (rc) {
            printf("ERROR; return code from pthread_join() is %d\n", rc);
            exit(-1);
        }
        printf("Main: completed join with thread %ld having a status of %ld\n",t,(long)status);
    }
    printf("Main: program completed. Exiting.\n");
    // to block main to support the threads it created until they terminate
    pthread_exit(NULL);
}
```

Output of the Example Program 2



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

Example Program 3

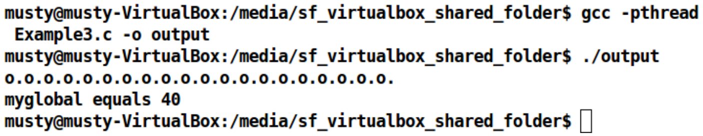


```c
#include <pthread.h>
#include <stdlib.h>
#include <stdio.h>

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 the threads it created until they terminate
    pthread_exit(NULL);
}
```

Output of the Example Program 3



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