Operating Systems (COMP 2101)

LAB 5 – Threads

Task 1. Compile and run a program with one thread

Download the “thread1.c” source code and print the content of this file.

```
$ wget raw.githubusercontent.com/mirektud/Operating-Systems/main/lab6/thread1.c
$ more thread1.c
```

You should see the following:

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

void * function(){
   printf("Hello world, this is child thread!\n");

   //terminate thread
   printf("Child thread is exiting\n");
   pthread_exit(NULL);
}
int main(){
   pthread_t t_id;
   printf("Hello world, this is master thread!\n");
   //create  thread
   int r = pthread_create(&t_id, NULL, function, NULL);
   if (r == 0) {
      printf("Child thread created, t_id = %lu\n", (unsigned long) t_id);
   }
   else {
      printf("Thread error nr: %d\n", r);
   }
 printf("Master thread is exiting\n");
 return 0;
}
```

Compile the source code and run the output program.

```
$ gcc thread1.c -o thread1 -lpthread
$ ./thread1
```

You should get similar output:

```
Hello world, this is master thread!
Child thread created, t_id = 123145487347712
Master thread is exiting
Hello world, this is child thread!
Hello world, this is child thread!
Child thread is exiting
```

or:

```
Hello world, this is master thread!
Child thread created, t_id = 123145487347712
Master thread is exiting
```

or:

```
Hello world, this is master thread!
Hello world, this is child thread!
Child thread is exiting
Child thread created, t_id = 123145487347712
Master thread is exiting
```

Task 2. Modify the *thread1.c* source code to block the master thread until the child thread terminates. Then save it as *thread2.c*

Open ***thread1.c*** using “pico”

```
$ pico thread1.c
```

Modify the source code…

Hint:

Add the following line:

```
pthread_join(t_id, NULL);
```

... in the ***main()*** function as described in Lecture 5 and save the modified source code as ***thread2.c .*** Then compile and run the output program.

```
$ gcc thread2.c -o thread2 -lpthread
$ ./thread2
```

You should get similar output:

```
Hello world, this is master thread!
Child thread created, t_id = 1171478272
Hello world, this is child thread!
Child thread is exiting
Master thread is exiting
```

Task 3. Compile and run a program with two threads

Download the “thread3.c” source code and print the content of this file.

```
$ wget raw.githubusercontent.com/mirektud/Operating-Systems/main/lab6/thread3.c
$ more thread3.c
```

You should see the following:

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

int i, j;

void * f1(){
   for (i=0; i<10000; i++){
      printf("1");
      fflush(stdout);
    }
   pthread_exit(NULL);
}

void * f2(){
   for (j=0; j<10000; j++){
      printf("2");
      fflush(stdout);
    }
   pthread_exit(NULL);
}

int main(){
   pthread_t t_id1, t_id2;
   int r1 = pthread_create(&t_id1, NULL, f1, NULL);
   int r2 = pthread_create(&t_id2, NULL, f2, NULL);
   pthread_join(t_id1, NULL);
   pthread_join(t_id2, NULL);
return 0;
}
```

Compile the source code and run the output program.

```
$ gcc thread3.c -o thread3 -lpthread
$ ./thread3
```

You should get similar output:

```
111122222222222122222222222111111222222221111111111111111111222211111222222
211111111111111111212222222222221111111112122222222222222221222222222222122
222222111111111111112222222211111111111111111111111111111111111111111111111
111111111111111111111111111111111111222222222222222111112222221111122221111
121111122222222221111111111111111121111111111111111111111111111111111211111
111111112111111111111111111111111111111111111111111111111111111111111112222
222111111111111111111111111111111111111111111111111111111111111222221111111
111111111122212222111111211111222222212222111111211111111111111111112222222
111111111111111111111111111111111111111111111112222211111121111122221111121
111111111111111111111111111122222222221111111111111111111111111121111111111
211111111111111111111111111111111111111111111111111111112122111111111121111
111111111111111111111112122211111111111111111111111111112122222222222222221
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111121222222222222222211
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111222
222222222222222222222222222222222222222222222222222222222222222222222222211

(...)
```

Task 4. Modify the *thread3.c* source code by adding one thread that prints “3” numbers 10000 times. Then save it as *thread4.c*

Open ***thread3.c*** using “pico”

```
$ pico thread3.c
```

HINT:
Add the following to the source code:

a) Add integer k

```
int i, j, k;
```

b) Add function f3() that prints “3” numbers 10000 times:

```
void * f3(){
   for (k=0; k<10000; k++){
      printf("3");
      fflush(stdout);
    }
   pthread_exit(NULL);
}
```

c) Add t_id3 to pthread_t.

```
pthread_t t_id1, t_id2, t_id3;
```

d) Create new thread that calls function f3

```
int r3 = pthread_create(&t_id3, NULL, f3, NULL);
```

e) Add pthread_join() function for this new thread.

```
pthread_join(t_id3, NULL);
```

f) Save the source code as thread4.c

Compile the source code and run the output program.

```
$ gcc thread4.c -o thread4 -lpthread
$ ./thread4
```

You should get similar output:

```
111122222222222122222222222111112222222221111111111111111111222211111222222
333333333332222222222222222222222222222222222222222222222222122222223222222
222222222222222222222222222222222222222222223222222222222222222222222223222
222222222222122222222222222222222222222222222222322222222222222122222222222
222222222222222222222222322222222222122222222222222222222222222222322212222
222222222222222222212222222222222222222222222222232222212222222222222222232
222222221222222222222222222222221222222222222222222212222222212222222222222
222222222222222222223222222222222222122222222222222222222222212222222222222
222222222222222122222222222222221222221111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111111111111
111111111111111111111111111111111111111111111111111111111111111111333232222
222222222212222222222221222222222222232222222222222222212133333333333333333
333333333333333333333332222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222233333333333
233333333333333333333333333333333333333333333333333333333333333333333333233
333222222222222222222222222222222222222222222222222222232222222222222222223
222222222222222222222222222222223222222222222222222222222222322222222222222
222212222222222222222322222222222222212222222122222222222222222122222222222
222222122222222222222322212222222222222222222223222222222222222222222222222
222222212222222222222222222222222223222222222222122222222222222222222222222
223222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222232222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
222222222222222222222222222222222222222222222222222222222222222222222222222
(...)
```

Tasks 5 & 6 – Compare execution time of a single-threaded program (thread5) and a multi-threaded program (thread6).

```
$ wget raw.githubusercontent.com/mirektud/Operating-Systems/main/lab6/thread5.c
$ more thread5.c
```

You should see the following:

```
#include <stdlib.h>
#include <stdio.h>
#include <unistd.h>

int i,j;
void f1(){
 for (i=0; i<10; i++){
 printf("1");
 fflush(stdout);
 sleep(1);
 }
}
void f2(){
 for (j=0; j<10; j++){
 printf("2");
 fflush(stdout);
 sleep(1);
 }
}
int main(){
 f1();
 f2();
return 0;
}
```

Compile the source code and measure the execution time of this program.

```
$ gcc thread5.c -o thread5
$ time ./thread5
```

You should get similar output:

```
11111111112222222222
real    0m20.004s
user    0m0.000s
sys     0m0.002s
```

```
$ wget raw.githubusercontent.com/mirektud/Operating-Systems/main/lab6/thread6.c
$ more thread6.c
```

You should see the following:

```
#include<pthread.h>
#include<stdio.h>
#include <unistd.h>
int i, j;

void * f1(){
   for (i=0; i<10; i++){
      printf("1");
      fflush(stdout);
      sleep(1);
    }
   pthread_exit(NULL);
}

void * f2(){
   for (j=0; j<10; j++){
      printf("2");
      fflush(stdout);
      sleep(1);
    }
   pthread_exit(NULL);
}
int main(){
   pthread_t t_id1, t_id2;
   int r1 = pthread_create(&t_id1, NULL, f1, NULL);
   int r2 = pthread_create(&t_id2, NULL, f2, NULL);
   pthread_join(t_id1, NULL);
   pthread_join(t_id2, NULL);
return 0;
}
```

Compile the source code and measure the execution time of this program.

```
$ gcc thread6.c -o thread6 -lpthread
$ time ./thread6
```

You should get similar output:

```
12212121212112121212
real    0m10.003s
user    0m0.000s
sys     0m0.002s
```

Task 7. Passing an integer to the thread function

```
$ wget raw.githubusercontent.com/mirektud/Operating-Systems/main/lab6/thread7.c
$ more thread7.c
```

You should see the following:

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

void * function(void * ptr){
       printf("Hello world, this is child thread!\n");
       int *x = (int*) ptr;
       printf("I received an integer: %d\n", *x);
       pthread_exit(NULL);
}

int main(){
       printf("Hello world, this is master thread!\n");
       pthread_t t_id;

       //a variable x stores the integer value
       int x = 10;

       // a pointer to the address of x will be passed
       int * ptr = &x;

       int r = pthread_create(&t_id, NULL, function, (void *)ptr);
       pthread_join(t_id, NULL);
       printf("Master thread is exiting\n");
}
```

Compile the source code and run this program.

```
$ gcc thread7.c -o thread7 -lpthread
$ ./thread7
```

You should get similar results:

```
$ ./thread7

Hello world, this is master thread!

Hello world, this is child thread!

I received an integer: 10

Master thread is exiting.
```

Task 8. Passing an array of characters (i.e. string) to the thread function

```
$ wget raw.githubusercontent.com/mirektud/Operating-Systems/main/lab6/thread8.c

$ more thread8.c
```

You should see the following:

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

void * function(void * ptr){
       printf("Hello world, this is child thread!\n");
       printf("I got a message: %s\n", (char*)ptr);
       pthread_exit(NULL);
}

int main(){
       printf("Hello world, this is master thread!\n");

       pthread_t t_id;

       //create a message "hello mr thread"
       char message[25] = {"hello mr thread"};

       //create a thread and pass the message to the thread function
       int r = pthread_create(&t_id, NULL, function, message);

       //wait for the thread function to exit
       pthread_join(t_id, NULL);

       printf("Master thread is exiting\n");
return 0;
}
```

Compile the source code and run this program.

```
$ gcc thread8.c -o thread8 -lpthread

$ ./thread8
```

You should get similar results:

```
$ ./thread8

Hello world, this is master thread!

Hello world, this is child thread!

I got a message: hello mr thread

Master thread is exiting
```

Task 9. A multi-threaded program with integer passing

```
$ wget raw.githubusercontent.com/mirektud/Operating-Systems/main/lab6/thread9.c
$ more thread9.c
```

You should see the following:

```
#include <stdlib.h>
#include <assert.h>
#include <pthread.h>
#include <unistd.h>
#include <stdio.h>
#define NUM_THREADS 5
void *perform_work(void *arguments){
  int index = *((int *)arguments);
  int sleep_time = index;
  printf("THREAD %d: Started.\n", index);
  printf("THREAD %d: Will be sleeping for %d seconds.\n", index, sleep_time);
  sleep(sleep_time);
  printf("THREAD %d: Ended.\n", index);
}
int main(void) {
  pthread_t threads[NUM_THREADS];
  int thread_args[NUM_THREADS];
  int i,j;
//create all threads one by one
  for (i = 0; i < NUM_THREADS; i++) {
    printf("IN MAIN: Creating thread %d.\n", i);
    threads[i] = i;
    thread_args[i] = i;
   int r = pthread_create(&threads[i], NULL, perform_work, &thread_args[i]);
          if (r == 0) {
             printf("Thread %d created succesfully\n", i);
           }
          else {
             printf("Thread %d error nr: %d\n", i, r);
           }
  }
  printf("IN MAIN: All threads are created.\n");

  //wait for each thread to complete
  for (j = 0; j < NUM_THREADS; j++) {
    pthread_join(threads[i], NULL);
    printf("IN MAIN: Thread %d has ended.\n", i);
  }

   printf("MAIN program has ended.\n");
   return 0;
}
```

Compile the source code and measure the execution time of this program.

```
$ gcc thread9.c -o thread9 -lpthread
$ ./thread9
```

You should get similar output:

```
IN MAIN: Creating thread 0.
IN MAIN: Creating thread 1.
IN MAIN: Creating thread 2.
IN MAIN: Creating thread 3.
IN MAIN: Creating thread 4.
IN MAIN: All threads are created.
THREAD 0: Started.
THREAD 0: Will be sleeping for 0 seconds.
THREAD 1: Started.
THREAD 1: Will be sleeping for 1 seconds.
THREAD 2: Started.
THREAD 2: Will be sleeping for 2 seconds.
THREAD 3: Started.
THREAD 3: Will be sleeping for 3 seconds.
THREAD 4: Started.
THREAD 4: Will be sleeping for 4 seconds.
THREAD 0: Ended.
THREAD 1: Ended.
THREAD 2: Ended.
THREAD 3: Ended.
THREAD 4: Ended.
IN MAIN: Thread 0 has ended.
IN MAIN: Thread 1 has ended.
IN MAIN: Thread 2 has ended.
IN MAIN: Thread 3 has ended.
IN MAIN: Thread 4 has ended.
MAIN program has ended.
```

Task 10. A multi-threaded program with string passing

Write a multi-threaded program that takes input arguments from the command line, creates a number of threads (i.e. one thread per one input argument), and passes input arguments to the threads.

Hint:

```
#include<stdio.h>

int main(int argc, char** argv){
  int i;
  printf("The number of input arguments are: %d\n",argc);
  printf("The arguments are:");
  for ( i = 0; i < argc; i++)
  {
     printf("%s\n", argv[i]);
  }
return 0;
}
```

Source:
raw.githubusercontent.com/mirektud/Operating-Systems/main/lab3/inputArg.c