1 Hands-on; Shared Memory I; Threads

1. **Creation and Termination Threads**, This example code13.c creates 5 threads with the **pthread_create()** routine. Each thread prints a "Hello World!" message, and then terminates with a call to **pthread_exit()**. Compile as

   ```
   gcc -o code13 code13.c -lpthread /* libpthread as a part of
   Unix/Linux operating systems */
   ./code13
   ```

```c
/*****************************************************************
 * FILE: hello.c
 * DESCRIPTION:
 *   A "hello world" Pthreads program.  Demonstrates thread creation
     and
 *   termination.
 * AUTHOR: Blaise Barney
 * LAST REVISED: 08/09/11
 *****************************************************************
     */
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define NUM_THREADS     5

void *PrintHello(void *threadid)
{
  sleep(10);
  long tid;
  tid = (long)threadid;
  printf("Hello World! It's me, thread #%ld!\n", tid);
  pthread_exit(NULL);
}

int main(int argc, char *argv[])
{
  pthread_t threads[NUM_THREADS];
  int rc;
  long t;
  for(t=0;t<NUM_THREADS;t++){
    printf("In main: creating thread %ld\n", t);
    rc = pthread_create(&threads[t], NULL, PrintHello, (void *)t);
    if (rc){
      printf("ERROR; return code from pthread_create() is %d\n", rc);
      exit(-1);
    }
  }

  /* Last thing that main() should do */
  pthread_exit(NULL);
}
```

2. **Passing Arguments to Threads 1**, This example code14.c demonstrates how to pass a simple integer to each thread.

```
gcc -o code14 code14.c -lpthread
./code14
```

```c
/******************************************************************
 * FILE: hello_arg1.c
 * DESCRIPTION:
 *   A "hello world" Pthreads program which demonstrates one safe way
 *   to pass arguments to threads during thread creation.
 * AUTHOR: Blaise Barney
 * LAST REVISED: 08/04/15
 ******************************************************************
    */
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define NUM_THREADS     8
char *messages[NUM_THREADS];

void *PrintHello(void *threadid)
{
  sleep(1);
  long taskid;
  taskid = (long) threadid;
  printf("Thread %ld: %s\n", taskid, messages[taskid]);
  pthread_exit(NULL);
}

int main(int argc, char *argv[])
{
  pthread_t threads[NUM_THREADS];
  long taskids[NUM_THREADS];
  int rc, t;

  messages[0] = "English: Hello World!";
  messages[1] = "French: Bonjour, le monde!";
  messages[2] = "Spanish: Hola al mundo";
  messages[3] = "Klingon: Nuq neH!";
  messages[4] = "German: Guten Tag, Welt!";
  messages[5] = "Russian: Zdravstvuyte, mir!";
  messages[6] = "Japan: Sekai e konnichiwa!";
  messages[7] = "Latin: Orbis, te saluto!";

  for(t=0;t<NUM_THREADS;t++) {
    taskids[t] = t;
    printf("Creating thread %d\n", t);
    rc = pthread_create(&threads[t], NULL, PrintHello, (void *)
    taskids[t]);
    if (rc) {
      printf("ERROR; return code from pthread_create() is %d\n", rc);
      exit(-1);
    }
  }

  pthread_exit(NULL);
}
```

3. **Passing Arguments to Threads 2**, This example code15.c shows how to setup/pass multiple arguments via a structure. Each thread receives a unique instance of the structure.

```
gcc -o code15 code15.c -lpthread
./code15
```

```
/******************************************************************
 * FILE: hello_arg2.c
 * DESCRIPTION:
 * A hello world Pthreads program which demonstrates another safe way
 * to pass arguments to threads during thread creation.  In this case,
 * a structure is used to pass multiple arguments.
 * AUTHOR: Blaise Barney
 * LAST REVISED: 01/29/09
 ******************************************************************
    */
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define NUM_THREADS     8

char *messages[NUM_THREADS];

struct thread_data
{
  int  thread_id;
  int  sum;
  char *message;
};

struct thread_data thread_data_array[NUM_THREADS];

void *PrintHello(void *threadarg)
{
  //    sleep(1);
  int taskid, sum;
  char *hello_msg;
  struct thread_data *my_data;

  sleep(1);
  my_data = (struct thread_data *) threadarg;
  taskid = my_data->thread_id;
 sum = my_data->sum;
  hello_msg = my_data->message;
  printf("Thread %d: %s  Sum=%d\n", taskid, hello_msg, sum);
  pthread_exit(NULL);
}

int main(int argc, char *argv[])
{
  pthread_t threads[NUM_THREADS];
  int *taskids[NUM_THREADS];
  int rc, t, sum;

 sum=0;
  messages[0] = "English: Hello World!";
  messages[1] = "French: Bonjour, le monde!";
  messages[2] = "Spanish: Hola al mundo";
  messages[3] = "Klingon: Nuq neH!";
  messages[4] = "German: Guten Tag, Welt!";
  messages[5] = "Russian: Zdravstvytye, mir!";
  messages[6] = "Japan: Sekai e konnichiwa!";
  messages[7] = "Latin: Orbis, te saluto!";

  for(t=0;t<NUM_THREADS;t++) {
    sum = sum + t;
    thread_data_array[t].thread_id = t;
    thread_data_array[t].sum = sum;
    thread_data_array[t].message = messages[t];
    printf("Creating thread %d\n", t);
    rc = pthread_create(&threads[t], NULL, PrintHello, (void *) &
    thread_data_array[t]);

    if (rc) {
      printf("ERROR; return code from pthread_create() is %d\n", rc);
      exit(-1);
    }
  }

  pthread_exit(NULL);
}
```

4. **Passing Arguments to Threads 3 - Incorrectly**, This example code16.c performs argument passing incorrectly.

   - It passes the address of variable t, *which is shared memory space and visible to all threads.*
   - The loop which creates threads modifies the contents of the address passed as an argument, *possibly before the created threads can access it.*

```
gcc -o code16 code16.c -lpthread
./code16
```

```c
/*****************************************************************************
 * FILE: bug3.c
 * DESCRIPTION:
 * This "hello world" Pthreads program demonstrates an unsafe (incorrect)
 * way to pass thread arguments at thread creation. Compare with hello_arg1.c.
 * In this case, the argument variable is changed by the main thread as it
 * creates new threads.
 * AUTHOR: Blaise Barney
 * LAST REVISED: 07/16/14
 ******************************************************************************/
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>

#define NUM_THREADS     8
void *PrintHello(void *threadid)
{
  //    sleep(1);
  long taskid;
  taskid = *(long *)threadid;
  printf("Hello from thread %ld\n", taskid);
  pthread_exit(NULL);
}

int main(int argc, char *argv[])
{
  pthread_t threads[NUM_THREADS];
  int rc;
  long t;

  for(t=0;t<NUM_THREADS;t++) {
    printf("Creating thread %ld\n", t);
    rc = pthread_create(&threads[t], NULL, PrintHello, (void *) &t);
    if (rc) {
      printf("ERROR; return code from pthread_create() is %d\n", rc);
      exit(-1);
    }
  }
  pthread_exit(NULL);
}
```

5. **Joining Threads**, This example code17.c demonstrates how to "wait" for thread completions by using the Pthread join routine. Since some implementations of Pthreads may not create threads in a joinable state, the threads in this example are explicitly created in a joinable state so that they can be joined later. Compile as

   `gcc -o code17 code17.c -lpthread -lm`

```c
/*****************************************************************
 * FILE: join.c
 * DESCRIPTION:
 * This example demonstrates how to "wait" for thread completions by
 * using the Pthread join routine.  Threads are explicitly created in
 * a joinable state for portability reasons. Use of the pthread_exit
 * status argument is also shown.
 * AUTHOR: 8/98 Blaise Barney
 * LAST REVISED:  01/30/09
 ***************************************************************** */
#include <pthread.h>
#include <stdio.h>
#include <stdlib.h>
#include <math.h>
#include <unistd.h>

#define NUM_THREADS     4
void *BusyWork(void *t)
{
  //      sleep(1);
  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);
}

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 detached attribute */
  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);
    rc = pthread_create(&thread[t], &attr, BusyWork, (void *)t);
    if (rc) {
      printf("ERROR; return code from pthread_create() is %d\n", rc);
      exit(-1);
    }
  }
```