

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
```

```
1  /*****
2  * FILE: hello.c
3  * DESCRIPTION:
4  *   A "hello world" Pthreads program. Demonstrates thread creation
5  *   and
6  *   termination.
7  * AUTHOR: Blaise Barney
8  * LAST REVISED: 08/09/11
9  *****/
10 #include <pthread.h>
11 #include <stdio.h>
12 #include <stdlib.h>
13 #include <unistd.h>
14 #define NUM_THREADS      5
15
16 void *PrintHello(void *threadid)
17 {
18     sleep(10);
19     long tid;
20     tid = (long)threadid;
21     printf("Hello World! It's me, thread #%ld!\n", tid);
22     pthread_exit(NULL);
23 }
24
25 int main(int argc, char *argv[])
26 {
27     pthread_t threads[NUM_THREADS];
28     int rc;
29     long t;
30     for (t=0;t<NUM_THREADS;t++){
31         printf("In main: creating thread %ld\n", t);
32         rc = pthread_create(&threads[t], NULL, PrintHello, (void *)t);
33         if (rc){
34             printf("ERROR; return code from pthread_create() is %d\n", rc);
35             exit(-1);
36         }
37     }
38
39     /* Last thing that main() should do */
40     pthread_exit(NULL);
41 }
```

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
```

```
1 /*****
2  * FILE: hello_arg1.c
3  * DESCRIPTION:
4  *   A "hello world" Pthreads program which demonstrates one safe way
5  *   to pass arguments to threads during thread creation.
6  * AUTHOR: Blaise Barney
7  * LAST REVISED: 08/04/15
8  *****/
9 #include <pthread.h>
10 #include <stdio.h>
11 #include <stdlib.h>
12 #include <unistd.h>
13
14 #define NUMTHREADS      8
15 char *messages [NUMTHREADS];
16
17 void *PrintHello(void *threadid)
18 {
19     sleep(1);
20     long taskid;
21     taskid = (long) threadid;
22     printf("Thread %ld: %s\n", taskid, messages[taskid]);
23     pthread_exit(NULL);
24 }
25
26 int main(int argc, char *argv[])
27 {
28     pthread_t threads [NUMTHREADS];
29     long taskids [NUMTHREADS];
30     int rc, t;
31
32     messages [0] = "English: Hello World!";
33     messages [1] = "French: Bonjour, le monde!";
34     messages [2] = "Spanish: Hola al mundo";
35     messages [3] = "Klingon: Nuq neH!";
36     messages [4] = "German: Guten Tag, Welt!";
37     messages [5] = "Russian: Zdravstvuyte, mir!";
38     messages [6] = "Japan: Sekai e konnichiwa!";
39     messages [7] = "Latin: Orbis, te saluto!";
40
41     for (t=0;t<NUMTHREADS;t++) {
42         taskids [t] = t;
43         printf("Creating thread %d\n", t);
44         rc = pthread_create(&threads[t], NULL, PrintHello, (void *)
45             taskids[t]);
46         if (rc) {
47             printf("ERROR: return code from pthread_create() is %d\n", rc);
48             exit(-1);
49         }
50     }
51     pthread_exit(NULL);
52 }
```

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
```

```
1  /*****
2  * FILE: hello_arg2.c
3  * DESCRIPTION:
4  * A hello world Pthreads program which demonstrates another safe way
5  * to pass arguments to threads during thread creation. In this case,
6  * a structure is used to pass multiple arguments.
7  * AUTHOR: Blaise Barney
8  * LAST REVISED: 01/29/09
9  *****/
10 #include <pthread.h>
11 #include <stdio.h>
12 #include <stdlib.h>
13 #include <unistd.h>
14
15 #define NUM_THREADS      8
16
17 char *messages [NUM_THREADS];
18
19 struct thread_data
20 {
21     int  thread_id;
22     int  sum;
23     char *message;
24 };
25
26 struct thread_data thread_data_array [NUM_THREADS];
27
28 void *PrintHello (void *threadarg)
29 {
30     //    sleep(1);
31     int taskid, sum;
32     char *hello_msg;
33     struct thread_data *my_data;
34
35     sleep(1);
36     my_data = (struct thread_data *) threadarg;
37     taskid = my_data->thread_id;
38     sum = my_data->sum;
39     hello_msg = my_data->message;
40     printf("Thread %d: %s Sum=%d\n", taskid, hello_msg, sum);
41     pthread_exit(NULL);
42 }
43
44 int main(int argc, char *argv[])
45 {
46     pthread_t threads [NUM_THREADS];
47     int *taskids [NUM_THREADS];
48     int rc, t, sum;
49
50     sum=0;
51     messages [0] = "English: Hello World!";
```

```

52 messages [1] = "French: Bonjour, le monde!";
53 messages [2] = "Spanish: Hola al mundo";
54 messages [3] = "Klingon: Nuq neH!";
55 messages [4] = "German: Guten Tag, Welt!";
56 messages [5] = "Russian: Zdravstvyte, mir!";
57 messages [6] = "Japan: Sekai e konnichiwa!";
58 messages [7] = "Latin: Orbis, te saluto!";
59
60 for (t=0; t<NUM_THREADS; t++) {
61     sum = sum + t;
62     thread_data_array[t].thread_id = t;
63     thread_data_array[t].sum = sum;
64     thread_data_array[t].message = messages[t];
65     printf("Creating thread %d\n", t);
66     rc = pthread_create(&threads[t], NULL, PrintHello, (void *) &
        thread_data_array[t]);
67
68     if (rc) {
69         printf("ERROR: return code from pthread_create() is %d\n", rc);
70         exit(-1);
71     }
72 }
73
74 pthread_exit(NULL);
75 }

```

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
```

```
1  /*****
2  * FILE: bug3.c
3  * DESCRIPTION:
4  * This "hello world" Pthreads program demonstrates an unsafe (
5  *   incorrect)
6  * way to pass thread arguments at thread creation. Compare with
7  *   hello_arg1.c.
8  * In this case, the argument variable is changed by the main thread
9  *   as it
10 *   creates new threads.
11 * AUTHOR: Blaise Barney
12 * LAST REVISED: 07/16/14
13 *****/
14 #include <pthread.h>
15 #include <stdio.h>
16 #include <stdlib.h>
17 #include <unistd.h>
18
19 #define NUMTHREADS      8
20 void *PrintHello(void *threadid)
21 {
22     //   sleep(1);
23     long taskid;
24     taskid = *(long *)threadid;
25     printf("Hello from thread %ld\n", taskid);
26     pthread_exit(NULL);
27 }
28
29 int main(int argc, char *argv[])
30 {
31     pthread_t threads[NUMTHREADS];
32     int rc;
33     long t;
34
35     for (t=0;t<NUMTHREADS;t++) {
36         printf("Creating thread %ld\n", t);
37         rc = pthread_create(&threads[t], NULL, PrintHello, (void *) &t);
38         if (rc) {
39             printf("ERROR; return code from pthread_create() is %d\n", rc);
40             exit(-1);
41         }
42     }
43     pthread_exit(NULL);
44 }
```

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
```

```

1 /*****
2 * FILE: join.c
3 * DESCRIPTION:
4 * This example demonstrates how to "wait" for thread completions by
5 * using the Pthread join routine. Threads are explicitly created in
6 * a joinable state for portability reasons. Use of the pthread_exit
7 * status argument is also shown.
8 * AUTHOR: 8/98 Blaise Barney
9 * LAST REVISED: 01/30/09
10 *****/
11 #include <pthread.h>
12 #include <stdio.h>
13 #include <stdlib.h>
14 #include <math.h>
15 #include <unistd.h>
16
17 #define NUMTHREADS 4
18 void *BusyWork(void *t)
19 {
20 // sleep(1);
21 int i;
22 long tid;
23 double result=0.0;
24 tid = (long)t;
25 printf("Thread %ld starting...\n",tid);
26 for (i=0; i<1000000; i++)
27 {
28 result = result + sin(i) * tan(i);
29 }
30 printf("Thread %ld done. Result = %e\n",tid, result);
31 pthread_exit((void*) t);
32 }
33
34 int main (int argc, char *argv[])
35 {
36 pthread_t thread[NUMTHREADS];
37 pthread_attr_t attr;
38 int rc;
39 long t;
40 void *status;
41 /* Initialize and set thread detached attribute */
42 pthread_attr_init(&attr);
43 pthread_attr_setdetachstate(&attr, PTHREAD_CREATE_JOINABLE);
44
45 for(t=0; t<NUMTHREADS; t++) {
46 printf("Main: creating thread %ld\n", t);
47 rc = pthread_create(&thread[t], &attr, BusyWork, (void *)t);
48 if (rc) {
49 printf("ERROR; return code from pthread_create() is %d\n", rc);
50 exit(-1);
51 }
52 }

```