**Race Condition:**

#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#include <stdlib.h>

#define NITER 1000000

int cnt = 0;

void \* Count(void \* a)

{

int i, tmp;

for(i = 0; i < NITER; i++)

{

tmp = cnt; /\* copy the global cnt locally \*/

tmp = tmp+1; /\* increment the local copy \*/

cnt = tmp; /\* store the local value into the global cnt \*/

}

}

int main(int argc, char \* argv[])

{

pthread\_t tid1, tid2;

if(pthread\_create(&tid1, NULL, Count, NULL))

{

printf("\n ERROR creating thread 1");

exit(1);

}

if(pthread\_create(&tid2, NULL, Count, NULL))

{

printf("\n ERROR creating thread 2");

exit(1);

}

if(pthread\_join(tid1, NULL)) /\* wait for the thread 1 to finish \*/

{

printf("\n ERROR joining thread");

exit(1);

}

if(pthread\_join(tid2, NULL)) /\* wait for the thread 2 to finish \*/

{

printf("\n ERROR joining thread");

exit(1);

}

if (cnt < 2 \* NITER)

printf("\n BOOM! cnt is [%d], should be %d\n", cnt, 2\*NITER);

else

printf("\n OK! cnt is [%d]\n", cnt);

pthread\_exit(NULL);

}

**Mutex Lock:**

#include <pthread.h>

#include <semaphore.h>

#include <stdio.h>

#include <stdlib.h>

#define NITER 1000000

pthread\_mutex\_t mutex;

int cnt = 0;

void \* Count(void \* a)

{

int i, tmp;

pthread\_mutex\_lock(&mutex);

for(i = 0; i < NITER; i++)

{

tmp = cnt; /\* copy the global cnt locally \*/

tmp = tmp+1; /\* increment the local copy \*/

cnt = tmp; /\* store the local value into the global cnt \*/

}

pthread\_mutex\_unlock(&mutex);

}

int main(int argc, char \* argv[])

{

pthread\_t tid1, tid2;

pthread\_mutex\_init(&mutex,NULL);

if(pthread\_create(&tid1, NULL, Count, NULL))

{

printf("\n ERROR creating thread 1");

exit(1);

}

if(pthread\_create(&tid2, NULL, Count, NULL))

{

printf("\n ERROR creating thread 2");

exit(1);

}

if(pthread\_join(tid1, NULL)) /\* wait for the thread 1 to finish \*/

{

printf("\n ERROR joining thread");

exit(1);

}

if(pthread\_join(tid2, NULL)) /\* wait for the thread 2 to finish \*/

{

printf("\n ERROR joining thread");

exit(1);

}

if (cnt < 2 \* NITER)

printf("\n BOOM! cnt is [%d], should be %d\n", cnt, 2\*NITER);

else

printf("\n OK! cnt is [%d]\n", cnt);

pthread\_exit(NULL);

}