Jung-Che Chang

Project #3

The Mutex Stack Challenge

changju@oregonstate.edu

1. Tell what machine you ran this on

   flip

2. Tell what operating system you were using

   Linux

3. Tell what compiler you used

   g++

4. Include, in your writeup, the pieces of code where you implemented the mutexes

```
void
Push(int n)
{
   if (USE_MUTEX)
   {
       omp_set_lock(&Lock);
   }

   StackPtr++;
   Stack[StackPtr] = n;

   if (USE_MUTEX)
   {
       omp_unset_lock(&Lock);
   }
}
```

```
int
Pop()
{
   // If the stack is empty, give the Push() function a chance to put
something on the stack:
   int t = 0;
   while (StackPtr < 0 && t < TIMEOUT)
       t++;

   // If there is nothing to pop, return;
   if (StackPtr < 0)
   {
       return FAILED;
   }

   if (USE_MUTEX)
   {
       omp_set_lock(&Lock);
   }

   int n = Stack[StackPtr];
   StackPtr--;

   if (USE_MUTEX)
   {
       omp_unset_lock(&Lock);
   }

   WasPopped[n] = true;
   return n;
}
```

```
int main(int argc, char *argv[])
{
#ifndef _OPENMP
   fprintf(stderr, "OpenMP is not supported here.\n");
   return 1;
#endif

   // this array is here to be sure all the pops actually happened:
   for (int i = 0; i < NUMN; i++)
   {
       WasPopped[i] = false;
   }
   omp_set_num_threads(2);

   // Initialize the lock
   omp_init_lock(&Lock);

   double time0 = omp_get_wtime();
#pragma omp parallel sections
   {
#pragma omp section
       PushAll();

#pragma omp section
       PopAll();
   }
   double time1 = omp_get_wtime();

   NumPopErrors = 0;
   for (int i = 0; i < NUMN; i++)
   {
       if (!WasPopped[i])
       {
           if (DEBUG)
               fprintf(stderr, "%6d wasn't popped\n", i);
           NumPopErrors++;
       }
   }

   char *useMutexString = (char *)"false";
   if (USE_MUTEX)
       useMutexString = (char *)" true";

   fprintf(stderr, "NUMN = %6d , USE_MUTEX = %s , NumPopErrors = %5d =
%6.2f%% , Elapsed time = %9.2lf microseconds\n",
           NUMN, useMutexString, NumPopErrors, 100. * (float)NumPopErrors /
(float)NUMN, 1000000. * (time1 - time0));

   // Destroy the lock
   omp_destroy_lock(&Lock);

   return 0;
}
```

5. Tell us what you discovered by doing this:

a. Does the non-mutex way of doing this ever work? If so, how often?
   Yes, but the non-mutex work rarely.

b. Does changing NUMN make any difference in the failure percentage?

   Changing the value of NUMN does make a difference in the failure percentage. As the value of NUMN increases, the likelihood of failure in the non-mutex approach also tends to increase.

c. Is there a difference in elapsed execution time between mutex and non-mutex? Why do you suppose this is?

   There is a difference in elapsed execution time between the mutex and non-mutex approaches. The non-mutex approach typically has a shorter execution time. The mutex approach has a longer execution time. I think it is because the mutex approach includes the time spent acquiring and releasing locks to ensure exclusive access to the shared resources, which prevents race conditions and guarantees data consistency.