Gebze Technical University

Department of Computer Engineering

CSE 437

REAL-TIME SYSTEM ARCHITECTURES

Assignment #1

Design Report

Gökbey Gazi KESKİN

1901042631

22.11.2022

Problem Definition

Standart STL data structures are not designed as thread-safe. So, a situation where multiple threads are accessing a data structure and at least one of them modifying it, it results as undefined behavior due to race conditions. The task of the assignment is to implement a generic thread-safe set.

The set should be able to:

-Insert an element (as l-values and r-values)

*No duplicate elements.

*Returns true on successful insertion.

-Remove an element

*Returns true on successful removal.

-Search an element.

*Returns true on element found.

-Clear itself

-Report its size

-Iterate its elements while applying a lambda function on each of them.

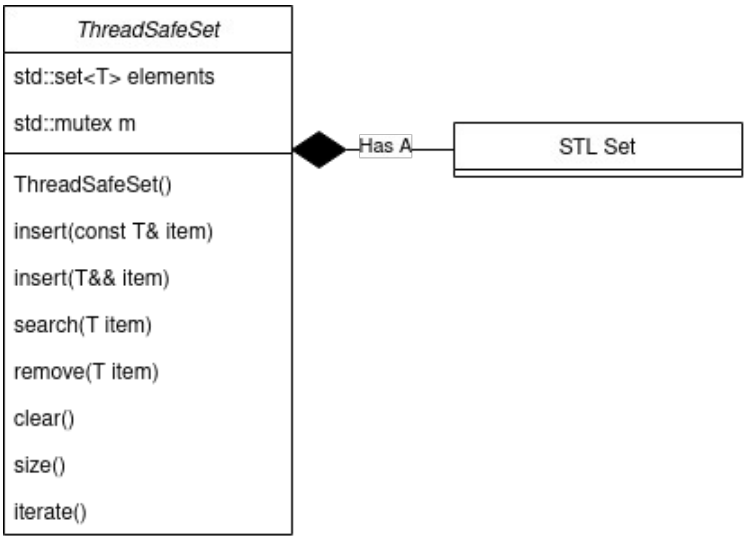
*Will accept a lambda expression as parameter.

Design & Analysis

An STL::Set is used and each modification access to set is considered as a critical region to prevent race conditions.

While testing for multiple threads, each reader thread should report when they are done searching. An thread_local global bool is used for this task.

Modern C++ thread functions are used instead of pthread library.



Implementation

Insertion

```
template <typename T>
bool ThreadSafeSet<T>::insert(const T& item){
    std::unique_lock<mutex> lock(m);
    if(search(item))
        return false;

    elements.insert(item);
    return true;
}

template <typename T>
bool ThreadSafeSet<T>::insert(T&& item){
    return insert(item);
}
```

Removal

```
template <typename T>
bool ThreadSafeSet<T>::remove(const T item){
    std::unique_lock<mutex> lock(m);
    if(!search(item)) return false;
    elements.erase(elements.find(item));
    return true;
}

template <typename T>
void ThreadSafeSet<T>::clear(){
    std::unique_lock<mutex> lock(m);
    elements.clear();
}
```

Size Report

```
template <typename T>
int ThreadSafeSet<T>::size(){
    return elements.size();
}
```

Iteration

```
template <typename T>
void ThreadSafeSet<T>::iterate(std::function<void(T)> func){
    set<int>::iterator itr;
    for(itr = elements.begin();itr != elements.end();itr++){
        func(*itr);
    }
}
```

Searching

```
template <typename T>
bool ThreadSafeSet<T>::search(T item){
    return elements.find(item) != elements.end();
}
```

Testing

Unit Tests

Insertion

```
int main(){

    ThreadSafeSet<double> tsset;
    cout<< "Inserting 5.3, Result:"<<tsset.insert(5.3)<<endl;
    cout<< "Inserting -13.7, Result:" << tsset.insert(-13.7)<<endl;
    cout<< "Inserting 5.3 again, Result:" << tsset.insert(5.3)<<endl;
    cout<< "Inserting 0, Result:"<< tsset.insert(0)<<endl;
    cout<< "Inserting -13.7 again, Result:" <<tsset.insert(-13.7)<<endl;

    cout<<"Iterating on Resulting Set:";
    tsset.iterate([](double x){cout<<x<<"|";});
    return 0;
}
```

```
Inserting 5.3, Result:1
Inserting -13.7, Result:1
Inserting 5.3 again, Result:0
Inserting 0, Result:1
Inserting -13.7 again, Result:0
Iterating on Resulting Set:-13.7|0|5.3|
```

Iteration

```
int main(){
    ThreadSafeSet<int> tsset;
    for(int i=0;i<10000;i++){
        tsset.insert(i);
    }

    tsset.iterate([](int x){
        cout<<"Number:"<<x<<": ";
        if(x%3==0) cout<<"Fizz";
        if(x%5==0) cout<<"Buzz";
        cout << endl;
    });


    return 0;
}
```

```
Number:9972: Fizz
Number:9973:
Number:9974:
Number:9975: FizzBuzz
Number:9976:
Number:9977:
Number:9978: Fizz
Number:9979:
Number:9980: Buzz
Number:9981: Fizz
Number:9982:
Number:9983:
Number:9984: Fizz
Number:9985: Buzz
Number:9986:
Number:9987: Fizz
Number:9988:
Number:9989:
Number:9990: FizzBuzz
Number:9991:
Number:9992:
Number:9993: Fizz
Number:9994:
Number:9995: Buzz
Number:9996: Fizz
Number:9997:
Number:9998:
Number:9999: Fizz
```

Removal

```
int main(){
    ThreadSafeSet<int> tsset;

    cout<<"Trying to remove 0 from empty set, Result:"<<tsset.remove(0)<<endl;

    for(int i=0;i<10;i++) tsset.insert(i);

    cout<<"Initial Set:";
    tsset.iterate([](int x){
        cout<<x<<",";
    });
    cout << endl;

    cout<<"Trying to remove existing number 5, Result:" << tsset.remove(5)<<endl;
    cout<<"Trying to remove 5 again, Result:" << tsset.remove(5)<<endl;
    cout<<"Trying to remove existing number 8, Result:" << tsset.remove(8)<<endl;

    cout<<"Resulting Set:";
    tsset.iterate([](int x){
        cout<<x<<",";
    });
    cout << endl;
    cout <<"Clearing the set..."<<endl;
    tsset.clear();
    cout<<"Resulting Set:";
    tsset.iterate([](int x){
        cout<<x<<",";
    });
```

```
Trying to remove 0 from empty set, Result:0
Initial Set:0,1,2,3,4,5,6,7,8,9,
Trying to remove existing number 5, Result:1
Trying to remove 5 again, Result:0
Trying to remove existing number 8, Result:1
Resulting Set:0,1,2,3,4,6,7,9,
Clearing the set...
Resulting Set:
```

Search

```
int main(){
    ThreadSafeSet<int> tsset;
    cout << "Searching 5 on empty set, Result:" << tsset.search(5) << endl;
    cout << "Adding numbers 0 to 10 to set..." << endl;
    for(int i=0;i<10;i++) tsset.insert(i);
    cout << "Searching 5 on new set, Result:" << tsset.search(5);
}
```

```
Searching 5 on empty set, Result:0
Adding numbers 0 to 10 to set...
Searching 5 on new set, Result:1
```

Integration Tests

Test1

```
/*
One reader, One writer thread with random number generation (between 0-300000).
All threads terminates when the number (369) is found.
*/
```

```
void writer(){
    for(;;){
        int addVal = getRand(0,300000);
        tsset.insert(addVal);
        if(found) return;
    }
}
void reader(int searchNum){
    for(;;){
        found = tsset.search(searchNum);
        cout <<"Searching "<<searchNum<<":"<< found <<endl;
        if(found) return;
    }
}
```

```
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/RTSA_HW1$ ./IntegrationTest1
Elapsed time = 2840897[µs]
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/RTSA_HW1$ ./IntegrationTest1
Elapsed time = 6847791[µs]
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/RTSA_HW1$ ./IntegrationTest1
Elapsed time = 1184035[µs]
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/RTSA_HW1$ ./IntegrationTest1
Elapsed time = 3774500[µs]
```

Test2

```
/*
One reader, One writer thread with sequential number generation (0 to 300000).
All threads terminate when the number (256178) is found.
*/
```

Note: Only the writer thread is changed. insert(i) instead of getRand.

```
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/RTSA_HW1$ ./IntegrationTest2
Elapsed time = 336242[µs]
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/RTSA_HW1$ ./IntegrationTest2
Elapsed time = 344082[µs]
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/RTSA_HW1$ ./IntegrationTest2
Elapsed time = 397700[µs]
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/RTSA_HW1$ ./IntegrationTest2
Elapsed time = 414816[µs]
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/RTSA_HW1$ ./IntegrationTest2
Elapsed time = 407157[µs]
gokbey@gokbey-ABRA-A5-V15-3:~/Desktop/RTSA_HW1$ ./IntegrationTest2
Elapsed time = 344253[µs]
```

Test3

```
/*
Multiple writer (10), Multiple reader(20) threads.
Writers adds random numbers (0-300000) to set.
10 reader threads searches for the numbers 0 to 10
10 reader thread iterates through the set and prints them over and over again.
All threads terminates when all 10 numbers are found.
*/
```

Note: found is a thread_local bool.

```
void writer(){
    for(;;){
        tsset.insert(getRand(0,300000));
        if(foundNumAmt==10) return;
    }
}
void reader(int searchNum){
    for(;;){
        found = tsset.search(searchNum);
        if(found) foundNumAmt++;
        cout <<"Searching "<<searchNum<<":"<< found <<endl;
        if(found) return;
    }
}

void reader2(){
    tsset.iterate([](int num){cout<<"Set contains:"<<num<<endl;});
    if(foundNumAmt==10) return;
}
```

```
Set contains:299987
Set contains:299988
Set contains:299989
Set contains:299991
Set contains:299992
Set contains:299993
Set contains:299994
Set contains:299995
Set contains:299997
Set contains:299998
Set contains:299999
Set contains:300000
Elapsed time = 19995412[µs]
```

Test4

```
/*
Multiple writer (10), Multiple reader(20) threads.
Writers adds number from 0 to 300000 to set.
10 reader threads searches for the numbers 299990 to 300000
10 reader thread iterates through the set and prints them over and over again.
All threads terminates when all 10 numbers are foundNumAmt.
*/
```

Note: Only the writer thread is changed. insert(i) instead of getRand.

```
Set contains:299988
Set contains:299989
Set contains:299990
Set contains:299991
Set contains:299992
Set contains:299993
Set contains:299994
Set contains:299995
Set contains:299996
Set contains:299997
Set contains:299998
Set contains:299999
Elapsed time = 28561255[µs]
```