

CS 7345 ADVANCED APPLICATIONS: LAB 3

Qing Gao 48358925

This is a library borrowing system. When the inventory of books is sufficient, customers can borrow a Book (num-1) from the library. If the inventory is insufficient, the borrowing fails.

**1.Document API for library**

**Book**

/**/get inventory of books**

int Book::getNum() {

return num;

}

/**/get book price**

double Book::getPrice() {

return price;

}

**//get book id number**

int Book::getBookNumber() {

return BookNumber;

}

**//borrow a book, book inventory minus 1**

void Book::borrow() {

num--;

cout<<"The Book: "<<BookNumber<<"has been borrowed"<<endl;

}

**Library system**

**//create borrow event system**

BorrowSystem\* BorrowSystem::CreateOrGet(){

if(!book\_BorrowSystem){

book\_BorrowSystem=new BorrowSystem();

}

return book\_BorrowSystem;

}

**//create borrow event thread**

void BorrowSystem::CreateBorrowThread(int num,int BookNumber,Book\* book){

BorrowThread\* newBorrow=new BorrowThread(num,BookNumber,book);

book\_BorrowThreadsMutex.lock();

book\_BorrowThreads.push\_back(newBorrow);

newBorrow->Startup();

book\_BorrowThreadsMutex.unlock();

}

**Library Thread**

**//initialize borrow event thread**

BorrowThread::BorrowThread(int num,int BookNumber,Book\* book){

book\_num=num;

book\_bookNumber=BookNumber;

book\_book=book;

}

**//start a borrow event thread**

void BorrowThread::Startup(){

book\_thread=new thread(BorrowThreadMain,this);

}

**//Main event handling of thread: get the inventory of book**

void BorrowThread::Work(){

int num=book\_book->getNum();

while(num>0){

book\_StatusMutex.lock();

book\_book->borrow();

num--;

book\_StatusMutex.unlock();

}

}

**//start work**

void BorrowThread::BorrowThreadMain(void\* BorrowThreadObject){

BorrowThread\* thisBorrow=(BorrowThread\*)BorrowThreadObject;

thisBorrow->Work();

}

**2.Multithreading architecture and execution model**

**Library Thread**

**//initialize borrow event thread**

BorrowThread::BorrowThread(int num,int BookNumber,Book\* book){

book\_num=num;

book\_bookNumber=BookNumber;

book\_book=book;

}

**Initalize borrow event thread with attribute.book\_num represents the inventory of the book, book\_bookNumber represents the id of the book.**

**//start a borrow event thread**

void BorrowThread::Startup(){

book\_thread=new thread(BorrowThreadMain,this);

}

**//Main event handling of thread: get the inventory of book**

void BorrowThread::Work(){

int num=book\_book->getNum();

while(num>0){

book\_StatusMutex.lock();

book\_book->borrow();

num--;

book\_StatusMutex.unlock();

}

}

**Borrow a book means the inventory of book -1, and we use mutex.lock()to protect this process.Through this process, we can prevent two people borrow the same book at the same time, resulting in errors caused by insufficient inventory.**

**//start work**

void BorrowThread::BorrowThreadMain(void\* BorrowThreadObject){

BorrowThread\* thisBorrow=(BorrowThread\*)BorrowThreadObject;

thisBorrow->Work();

}

**Library System**

//create borrow event thread

void BorrowSystem::CreateBorrowThread(int num,int BookNumber,Book\* book){

BorrowThread\* newBorrow=new BorrowThread(num,BookNumber,book);

book\_BorrowThreadsMutex.lock();

book\_BorrowThreads.push\_back(newBorrow);

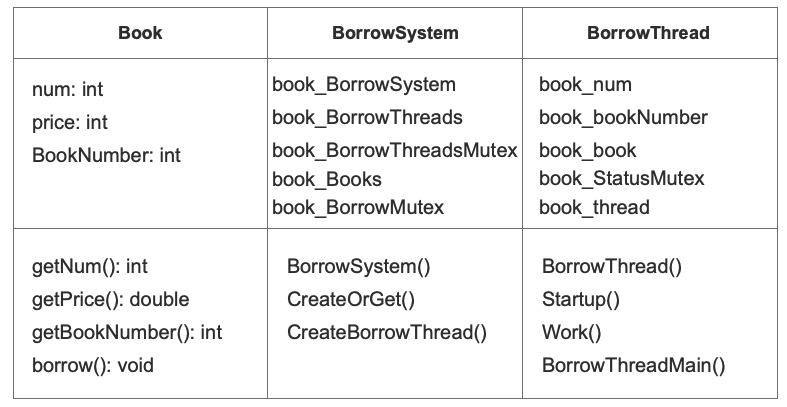
newBorrow->Startup();

book\_BorrowThreadsMutex.unlock();

}

**Through Mutex.lock to protect the borrowing and creation process of books.**

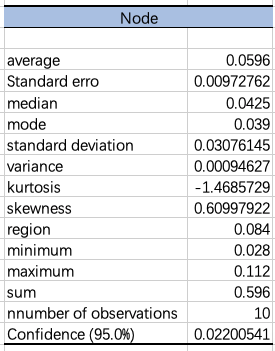
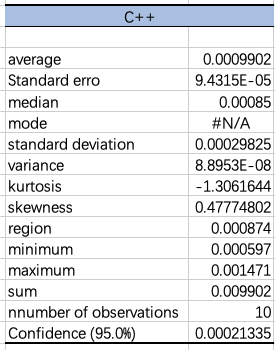
**3.UML**

****

**4.Comparison**

For native main.cpp, I use the MakeFile to run the code.

For index.html, I use the node.js to run the code.

****

From the above statistical results, we can see that the average running time of C + + is 0.0009902s, while that of node is 0.0596s, So C + + is significantly faster than node.

By comparing the confidence interval, we can see that the local operation of C + + is more stable than that of Node.

In conclusion, I think the local operation of C + + is more suitable for this project