

Programming in C/C++

Exercises set six: multi-threading 1

Christiaan Steenkist
Jaime Betancor Valado
Remco Bos

December 21, 2016

Exercise 38, basic multithreading with recursive program

We were tasked to design a program that recursively visits all elements of a starting directory and computes the total size of its regular files.

Code listings

Listing 1: main.ih

```
1 #include "main.h"
2
3 #include <thread>
4 #include <chrono>
5
6 #include <sys/types.h>
7 #include <sys/stat.h>
8 #include <dirent.h>
9
10 using namespace std;
```

Listing 2: main.cc

```
1 #include "main.ih"
2
3 int main(int argc, char **argv)
4 {
5     if (argc <= 1)
```

```

6  {
7      cerr << "Please supply a folder path.\n";
8      return 1;
9  }
10
11  bool completed = false;
12  size_t bytes = 0;
13  thread byteCounter(countBytes,
14      std::ref(completed), std::ref(bytes), argv[1]);
15  thread progressBar(timeProcess,
16      std::ref(completed));
17
18  byteCounter.join();
19  progressBar.join();
20
21  cout << bytes << " bytes\n";
22 }

```

Listing 3: openfolder.cc

```

1  #include "main.ih"
2
3  size_t openFolder(string path)
4  {
5      DIR *dir = opendir(path.c_str());
6      if (dir == 0)
7          return 0;
8
9      size_t bytes = 0;
10     struct dirent *pent = 0;
11     while((pent = readdir(dir))
12     {
13         struct stat sb;
14         lstat(pent->d_name, &sb);
15
16         bytes += sb.st_size;
17
18         if (S_ISDIR(sb.st_mode) == 0)
19         {
20             string newPath = path + '/' + pent->d_name;
21             bytes += openFolder(newPath);

```

```

22     }
23 }
24     closedir(dir);
25
26     return bytes;
27 }

```

Listing 4: timeprocess.cc

```

1  #include "main.ih"
2
3  void timeProcess(bool &status)
4  {
5      while (!status)
6      {
7          cout << '.' << std::flush;
8          this_thread::sleep_for(chrono::seconds(1));
9      }
10
11     cout << std::endl;
12 }

```

Exercise 39, using chrono/clock facilities

We were tasked to display the time at the beginning and end of a program.

Output

Listing 5: output

```

1  .
2  203509 bytes
3  Program starts at Thu Dec 15 14:44:36 2016
4
5  Program ends at Thu Dec 15 14:44:36 2016
6
7  Total time passed 0.0112944 s

```

Code listings

Listing 6: watch.h

```

1  #ifndef WATCH_H

```

```

2  #define WATCH_H
3
4  #include <iostream>
5  #include <chrono>
6  #include <ctime>
7
8  class watch
9  {
10     std::chrono::time_point<std::chrono::system_clock>
        d_start, d_end;
11     public:
12         void startPoint();
13         void endPoint();
14         void startShowTime();
15         void endShowTime();
16         void timePassed();
17 };
18
19 #endif

```

Listing 7: main.cc

```

1  #include "main.ih"
2
3  int main(int argc, char **argv)
4  {
5      watch clock;
6      clock.startPoint();
7
8      if (argc <= 1)
9      {
10         cerr << "Please supply a folder path.\n";
11         return 1;
12     }
13
14     bool completed = false;
15     size_t bytes = 0;
16     thread byteCounter(countBytes,
17         std::ref(completed), std::ref(bytes), argv[1]);
18     thread progressBar(timeProcess,
19         std::ref(completed));

```

```

20
21   byteCounter.join();
22
23   clock.endPoint();
24
25   progressBar.join();
26
27   cout << bytes << " bytes\n";
28
29   clock.startShowTime();
30
31   clock.endShowTime();
32
33   clock.timePassed();
34 }

```

Listing 8: endPoint.cc

```

1  #include "main.ih"
2
3  void watch::endPoint()
4  {
5      d_end = std::chrono::system_clock::now();
6  }

```

Listing 9: endShowTime.cc

```

1  #include "main.ih"
2
3  void watch::endShowTime()
4  {
5      std::time_t endTime =
6          std::chrono::system_clock::to_time_t(d_end);
7      std::cout << " Program ends at "
8          << std::ctime(&endTime) << '\n';
9  }

```

Listing 10: startPoint.cc

```

1  #include "main.ih"
2
3  void watch::startPoint()

```

```

4 {
5     d_start = std::chrono::system_clock::now();
6 }

```

Listing 11: startShowTime.cc

```

1 #include "main.ih"
2
3 void watch::startShowTime()
4 {
5     std::time_t startTime =
6         std::chrono::system_clock::to_time_t(d_start);
7     std::cout << " Program starts at "
8         << std::ctime(&startTime) << '\n';
9 }

```

Listing 12: timePassed.cc

```

1 #include "watch.h"
2
3 void watch::timePassed()
4 {
5     std::chrono::duration<double> totalTime =
6         d_end - d_start;
7     std::cout << "Total time passed "
8         << totalTime.count() << '\n';
9 }

```

Exercise 40, thread-safe queue

A proxy and a lot of lock guards attempt to make this queue thread-safe.

Code listings

Listing 13: safequeue.ih

```

1 #include "safequeue.h"
2
3 using namespace std;

```

Listing 14: safequeue.h

```

1 #ifndef SAFEQUEUE_H

```

```

2  #define SAFEQUEUE_H
3
4  #include "semaphore.h"
5  #include <queue>
6  #include <string>
7
8  class SafeQueue
9  {
10     Semaphore d_semaphore;
11     std::mutex d_mutex;
12     std::queue<std::string> d_queue;
13
14     // The RefProxy locks the mutex at creation
15     // and unlocks it at destruction.
16     class RefProxy
17     {
18     public:
19         SafeQueue *d_queue;
20         std::string &d_string;
21
22         RefProxy(SafeQueue *queue, std::string &ref);
23         ~RefProxy();
24
25         RefProxy &operator=(
26             RefProxy const &rhs) = delete;
27
28         std::string const &operator=(
29             std::string const &rhs);
30         operator std::string const &() const;
31     };
32
33     std::mutex *mutex();
34
35     public:
36     bool empty();
37
38     RefProxy front();
39     RefProxy back();
40
41     void pop();

```

```

42     void push(std::string const &item);
43 };
44
45 #endif

```

Semaphore

Listing 15: semaphore.ih

```

1  #include "semaphore.h"
2
3  using namespace std;

```

Listing 16: semaphore.h

```

1  #ifndef SEMAPHORE_H
2  #define SEMAPHORE_H
3
4  #include <condition_variable>
5  #include <mutex>
6  #include <cstdint>
7
8  class Semaphore
9  {
10     std::mutex d_mutex;
11     std::condition_variable d_condition;
12     std::size_t d_counter = 0;
13
14     public:
15         Semaphore() = default;
16         Semaphore(std::size_t count);
17
18         void notify();
19         void wait();
20 };
21
22 #endif

```

Listing 17: notify.cc

```

1  #include "semaphore.ih"
2

```



```

3 void Semaphore::notify()
4 {
5     lock_guard<mutex> lock(d_mutex);
6     if (d_counter++ == 0)
7         d_condition.notify_all();
8 }

```

Listing 18: semaphoreconstr.cc

```

1 #include "semaphore.ih"
2
3 Semaphore::Semaphore(size_t count)
4 :
5     d_counter(count)
6 {
7 }

```

Listing 19: wait.cc

```

1 #include "semaphore.ih"
2
3 void Semaphore::wait()
4 {
5     unique_lock<mutex> lock(d_mutex);
6     while (d_counter == 0)
7         d_condition.wait(lock);
8
9     --d_counter;
10 }

```

SafeQueue

Listing 20: back.cc

```

1 #include "safequeue.ih"
2
3 SafeQueue::RefProxy SafeQueue::back()
4 {
5     return RefProxy(this, d_queue.back());
6 }

```

Listing 21: empty.cc

```

1 #include "safequeue.ih"

```

```

2
3 bool SafeQueue::empty()
4 {
5     return d_queue.empty();
6 }

```

Listing 22: front.cc

```

1 #include "safequeue.ih"
2
3 SafeQueue::RefProxy SafeQueue::front()
4 {
5     return RefProxy(this, d_queue.front());
6 }

```

Listing 23: pop.cc

```

1 #include "safequeue.ih"
2
3 void SafeQueue::pop()
4 {
5     d_mutex.lock();
6     while(d_queue.size() == 0)
7     {
8         d_mutex.unlock();
9         d_semaphore.wait();
10        d_mutex.lock();
11    }
12    d_queue.pop();
13    d_mutex.unlock();
14 }

```

Listing 24: push.cc

```

1 #include "safequeue.ih"
2
3 void SafeQueue::push(string const &item)
4 {
5     d_mutex.lock();
6     d_queue.push(item);
7     d_mutex.unlock();
8     d_semaphore.notify();
9 }

```

SafeQueue::RefProxy

Listing 25: operator=.cc

```
1 #include "safequeue.ih"
2
3 string const &SafeQueue::RefProxy::operator=(
4     std::string const &rhs)
5 {
6     d_string = rhs;
7     return d_string;
8 }
```

Listing 26: promotor.cc

```
1 #include "safequeue.ih"
2
3 SafeQueue::RefProxy::operator string const &() const
4 {
5     return d_string;
6 }
```

Listing 27: proxyconstr.cc

```
1 #include "safequeue.ih"
2
3 SafeQueue::RefProxy::RefProxy(SafeQueue *queue,
4     string &ref)
5 :
6     d_queue(queue),
7     d_string(ref)
8 {
9     d_queue->mutex()->lock();
10 }
```

Listing 28: proxyconstr.cc

```
1 #include "safequeue.ih"
2
3 SafeQueue::RefProxy::~RefProxy()
4 {
5     d_queue->mutex()->unlock();
6 }
```

Exercise 42, establish connection between a parent and child process

We were tasked to pass output from a child process to the parent process by using fork and exec.

Code listings

Listing 29: main.cc

```
1  #include "main.h"
2
3  int main()
4  {
5      pid_t pID;
6
7      pID = fork();
8      if (pID == 0) // Child
9      {
10         dup2(1, STDOUT_FILENO); // Copies from one file
                                   // descriptor to another
11                                   // Argument 1 to 2, 0=
                                   // stdin, 1=stdout, 2=stderr
12         execl("/bin/ls", "ls", (char *) 0); // execute
                                   // command
13
14         close(1); // close file descriptor
15     }
16     else if (pID > 0) // Parent
17     {
18         // should access file descriptor, (argument 2 in
19         // line 10)
20         // must display the standard output(files+
21         // directories), lines and characters
22         cout << "lines: " << << '\n';
23     }
24     else
25     {
26         cout << "Fork failed: \n";
27         return 1;
28     }
29 }
```

Listing 30: main.h

```
1 #include <stdio.h>
2 #include <unistd.h>
3 #include <iostream>
4 #include <algorithm>
5 #include <vector>
6
7 using namespace std;
8
9 void usechildoutput(int count, char buffer[]);
```

Listing 31: usechildoutput.cc

```
1 #include "main.h"
2
3 void usechildoutput(int count, char buffer[])
4 {
5     int lines = 0;
6     for (int bufferElement = 0;
7         bufferElement <= count; ++bufferElement)
8     {
9         cout << buffer[bufferElement];
10        if (buffer[bufferElement] == '\n')
11            lines += 1;
12    }
13    cout << "Number of characters: " << count << "\n";
14    cout << "Number of lines: " << lines << "\n";
15 }
```

Exercise 43: design a simple multi-thread program

We were tasked to make a program with threads counting vowels, hexadecimals, digits and punctuation character in a file passed to the program.

Code listings

Listing 32: task.h

```
1 #ifndef TASK_H
2 #define TASK_H
3
```

```

4 #include <iostream>
5 #include <vector>
6
7 class Task: public std::vector<char>
8 {
9     public:
10         Task(std::istream &file);
11         void countVowel();
12         void countDigit();
13         void countHexDec();
14         void countPunctChar();
15 };
16
17 #endif

```

Listing 33: task.ih

```

1 #include "task.h"
2 #include <algorithm>
3 #include <iterator>
4 #include <cctype>
5 #include <cstring>
6
7 using namespace std;
8
9 bool countChar(string &characterType, char character);

```

Listing 34: main.cc

```

1 #include "task.ih"
2 #include <thread>
3 #include <chrono>
4
5 int main(int argc, char **argv)
6 {
7     using namespace std::chrono;
8     time_point<system_clock> start;
9     start = system_clock::now();
10
11     Task task(cin);
12

```

```

13     if (argc >=2) //perform threads in sequence
14     {
15         thread vowelThread(&Task::countVowel, task);
16         vowelThread.join();
17         thread digitThread(&Task::countDigit, task);
18         digitThread.join();
19         thread hexdecThread(&Task::countHexDec, task);
20         hexdecThread.join();
21         thread punctThread(&Task::countPunctChar, task
22     );
23     punctThread.join();
24     }
25     else //perform threads in parallel
26     {
27         thread vowelThread(&Task::countVowel, task);
28         thread digitThread(&Task::countDigit, task);
29         thread hexdecThread(&Task::countHexDec, task);
30         thread punctThread(&Task::countPunctChar, task
31     );
32     vowelThread.join();
33     digitThread.join();
34     hexdecThread.join();
35     punctThread.join();
36     }
37     time_point<system_clock> end;
38     end = system_clock::now();
39     duration<double> program_runtime = end-start;
40     cout << "Program runtime: " << program_runtime.
41     count() << "s \n";

```

Listing 35: task.cc

```

1  #include "task.ih"
2
3  Task::Task(std::istream &file)
4  {
5      copy(istream_iterator<char>(file),
6      istream_iterator<char>(), back_inserter(*this));

```

Listing 36: countdigit.cc

```
1 #include "task.ih"
2
3 void Task::countDigit()
4 {
5     string digit = "digit";
6     int myCount = count_if(begin(), end(),
7         [&] (char character)
8         {
9             return countChar(digit, character);
10        }
11    );
12    cout << "Digits: " << myCount << "\n";
13 }
```

Listing 37: counthexdec.cc

```
1 #include "task.ih"
2
3 void Task::countHexDec()
4 {
5     string hexadecimal = "hexadecimal";
6     int myCount = count_if(begin(), end(),
7         [&] (char character)
8         {
9             return countChar(hexadecimal, character);
10        }
11    );
12    cout << "Hexadecimals: " << myCount << "\n";
13 }
```

Listing 38: countpunctchar.cc

```
1 #include "task.ih"
2
3 void Task::countPunctChar()
4 {
5     string punctuation = "punctuation";
6     int myCount = count_if(begin(), end(),
7         [&] (char character)
8         {
```



```

9             return countChar(punctuation, character);
10         }
11     };
12     cout << "Punctuation characters: " << myCount << "
    \n";
13 }

```

Listing 39: countvowelcc

```

1  #include "task.ih"
2
3  void Task::countVowel()
4  {
5      string vowel = "vowel";
6      int myCount = count_if(begin(), end(),
7          [&] (char character)
8          {
9              return countChar(vowel, character);
10         }
11     );
12     cout << "Vowels: " << myCount << "\n";
13 }

```