

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

```

Code listings

Listing 6: main.cc

```

1  #include "main.ih"

```

```

2  #include <iostream>
3  #include <chrono>
4  #include <ctime>
5
6  int main(int argc, char **argv)
7  {
8      // Time block
9      chrono::time_point<chrono::system_clock> start,
10         end;
11
12     start = chrono::system_clock::now();
13
14     if (argc <= 1)
15     {
16         cerr << "Please supply a folder path.\n";
17         return 0;
18     }
19
20     mutex statusMutex;
21     bool completed = false;
22     size_t bytes = 0;
23     thread byteCounter(countBytes,
24         std::ref(statusMutex), std::ref(completed),
25         std::ref(bytes), argv[1]);
26     thread progressBar(timeProcess,
27         std::ref(statusMutex), std::ref(completed));
28
29     byteCounter.join();
30
31     end = chrono::system_clock::now();
32     progressBar.join();
33     cout << bytes << " bytes\n";
34
35     // Print block
36
37     time_t startTime =
38         chrono::system_clock::to_time_t(start);
39     cout << " Program starts at "
40         << ctime(&startTime) << '\n';
41

```

```

42     time_t endTime =
43         chrono::system_clock::to_time_t(end);
44     cout << " Program ends at "
45         << ctime(&endTime) << '\n';
46
47     chrono::duration<double> totalTime =
48         end - start;
49     cout << "Total time passed "
50         << totalTime.count() << '\n';
51 }

```

Exercise 40, thread-safe queue

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

Code listings

Listing 7: safequeue.ih

```

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

```

Listing 8: 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         SafeQueue *d_queue;
19         std::string &d_string;
20
21     public:
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 9: semaphore.ih

```

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

```

Listing 10: 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 11: 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 12: semaphoreconstr.cc

```

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

```

Listing 13: 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 14: back.cc

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

Listing 15: empty.cc

```
1 #include "safequeue.ih"
2
3 bool SafeQueue::empty()
4 {
5     return d_queue.empty();
6 }
```

Listing 16: front.cc

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

Listing 17: 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 18: 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 19: 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 20: promotor.cc

```

1 #include "safequeue.ih"

```

```

2
3 SafeQueue::RefProxy::operator string const &() const
4 {
5     return d_string;
6 }

```

Listing 21: 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 22: 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 23: main.cc

```

1 #include "main.h"
2
3 int main()
4 {

```

```

5   int fileDescr[2];
6   if (pipe(fileDescr) == -1)
7       exit(1);
8
9   int pID = fork();
10  if (pID == 0) //Child
11  {
12      while ((dup2(fileDescr[1],
13          STDOUT_FILENO) == -1)) {}
14      close(fileDescr[1]);
15      close(fileDescr[0]);
16      execl("/bin/ls", "ls", (char *) 0);
17      _exit(1);
18  }
19
20  char buffer[4096];
21  while (1)
22  {
23      int count = read(fileDescr[0], buffer,
24          sizeof(buffer));
25      if (count == -1)
26      {
27          exit(1);
28      }
29      else if (count == 0)
30      {
31          break;
32      }
33      else
34      {
35          usechildoutput(count, buffer);
36          exit(1);
37      }
38  }
39  close(fileDescr[0]);
40  wait(0);
41  }

```

Listing 24: main.h

```

1  #include <stdio.h>

```

```

2 #include <unistd.h>
3 #include <stdlib.h>
4 #include <errno.h>
5 #include <sys/wait.h>
6 #include <iostream>
7
8 using namespace std;
9
10 void usechildoutput(int count, char buffer[]);

```

Listing 25: 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, hexadecimal, digits and punctuation character in a file passed to the program.

Code listings

Listing 26: 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 27: task.ih

```

1 #include "task.h"
2 #include <algorithm>
3 #include <iterator>
4 #include <cctype>
5
6 using namespace std;

```

Listing 28: 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,end;
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,
16                             &task);

```

```

17     vowelThread.join();
18     thread digitThread(&Task::countDigit,
19         &task);
20     digitThread.join();
21     thread hexdecThread(&Task::countHexDec,
22         &task);
23     hexdecThread.join();
24     thread punctThread(&Task::countPunctChar,
25         &task);
26     punctThread.join();
27 }
28 else //perform threads in parallel
29 {
30     thread vowelThread(&Task::countVowel,
31         &task);
32     thread digitThread(&Task::countDigit,
33         &task);
34     thread hexdecThread(&Task::countHexDec,
35         &task);
36     thread punctThread(&Task::countPunctChar,
37         &task);
38     vowelThread.join();
39     digitThread.join();
40     hexdecThread.join();
41     punctThread.join();
42 }
43
44 end = system_clock::now();
45 duration<double> program_runtime = end-start;
46 cout << "Program runtime: "
47     << program_runtime.count() << "s \n";
48 }

```

Listing 29: task.cc

```

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

```

```

7     back_inserter(*this));
8 }

```

Listing 30: countdigit.cc

```

1 #include "task.ih"
2
3 void Task::countDigit()
4 {
5     size_t ret = 0;
6     for_each(begin(), end(),
7         [&] (char c)
8         {
9             if (string("1234567890").find(c)
10                 != string::npos)
11
12                 ++ret;
13         }
14     );
15     cout << "Digits: " << ret << "\n";
16 }

```

Listing 31: counthexdec.cc

```

1 #include "task.ih"
2
3 void Task::countHexDec()
4 {
5     size_t ret = 0;
6     for_each(begin(), end(),
7         [&] (char c)
8         {
9             if (isxdigit(c))
10                 ++ret;
11         }
12     );
13     cout << "Hexadecimals: " << ret << "\n";
14 }

```

Listing 32: countpunctchar.cc

```

1 #include "task.ih"

```

```

2
3 void Task::countPunctChar()
4 {
5     size_t ret = 0;
6     for_each(begin(), end(),
7         [&] (char c)
8         {
9             if (ispunct(c))
10                 ++ret;
11         }
12     );
13     cout << "Punctuation characters: "
14         << ret << "\n";
15 }

```

Listing 33: countvowelcc

```

1 #include "task.ih"
2
3 void Task::countVowel()
4 {
5     size_t ret = 0;
6     for_each(begin(), end(),
7         [&] (char c)
8         {
9             if (string("aeiouAEIOU").find(c)
10                 != string::npos)
11
12                 ++ret;
13         }
14     );
15     cout << "Vowels: " << ret << "\n";
16 }

```