Lab1 实验报告

王卫东　221900332

2024 年 10 月 9 日

一　Program Structure and Design

For the reassmbler.hh, I use a map structure to store the disjoint intervals:

```
  class Reassembler
{
  public:
  // Construct Reassembler to write into given ByteStream.
  // explicit Reassembler( ByteStream&& output ) : output_( std::move( output ) )
       {}
  explicit Reassembler( ByteStream&& output ) : output_( std::move( output ) ),
       pending_data_() {}

  void insert( uint64_t first_index, std::string data, bool is_last_substring );
  // How many bytes are stored in the Reassembler itself?
  uint64_t bytes_pending() const;
  // Access output stream reader
  Reader& reader() { return output_.reader(); }
  const Reader& reader() const { return output_.reader(); }
  // Access output stream writer, but const-only (can't write from outside)
  const Writer& writer() const { return output_.writer(); }

  private:
    ByteStream output_; // the Reassembler writes to this ByteStream
    std::map<uint64_t, std::string> pending_data_; // indexed substrings that can't
         yet be written
    uint64_t cur_index_ { 0 }; // the index of the first byte in the reassembler
    uint64_t eof_index_ { std::numeric_limits<uint64_t>::max() }; // the index of
        the last byte in the entire stream
};
```

And for the reassmbler.cc, I implement the insert function as follows:

```
void Reassembler::insert( uint64_t first_index, string data, bool
     is_last_substring )
{
  // Your code here.
  if ( first_index >= cur_index_ + output_.writer().available_capacity() ) {
    return; // 超出可用容量，丢弃数据
  }
  uint64_t insert_end = first_index + data.size();
  if ( is_last_substring ) {
    eof_index_ = insert_end;
  }
  // 截取超出部分
  insert_end = min( insert_end, cur_index_ + output_.writer().available_capacity
       () );
  // 插入或合并区间
  // 若insert_end < cur_index，说明数据已经被丢弃，不需要插入
  if ( insert_end > cur_index_ ) {
    auto it = pending_data_.lower_bound( first_index );
    if ( it != pending_data_.begin() && prev( it )->first + prev( it )->second.
         size() >= first_index ) {
      --it;
    }
    // string new_data = data.substr(0, insert_end - first_index);
    uint64_t insert_start = max( cur_index_, first_index );
    string new_data = data.substr( insert_start - first_index, insert_end -
         insert_start );
    while ( it != pending_data_.end() && it->first <= insert_end ) {
      if ( it->first < first_index ) {
        insert_start = it->first;
        new_data = it->second.substr( 0, first_index - it->first ) + new_data;
      }
      if ( it->first + it->second.size() > insert_end ) {
        new_data += it->second.substr( insert_end - it->first, it->second.size() -
              ( insert_end - it->first ) );
      }
      auto tmp = it;
      ++it;
      pending_data_.erase( tmp );
    }
    pending_data_[insert_start] = new_data;
  }
  // 输出可以写入的部分
  auto writable_it = pending_data_.find( cur_index_ );
  while ( writable_it != pending_data_.end() && writable_it->first == cur_index_
       ) {
    output_.writer().push( writable_it->second );
```

```
      cur_index_ += writable_it->second.size();
      writable_it = pending_data_.erase( writable_it );
    }

    // 判断是否应关闭输出流
    if ( cur_index_ == eof_index_ ) {
      output_.writer().close();
    }
  }
```

二 Experimental Results

```
      Start 11: reassembler_seq
10/17 Test #11: reassembler_seq ..................   Passed    0.02 sec
      Start 12: reassembler_dup
11/17 Test #12: reassembler_dup ..................   Passed    0.03 sec
      Start 13: reassembler_holes
12/17 Test #13: reassembler_holes ................   Passed    0.01 sec
      Start 14: reassembler_overlapping
13/17 Test #14: reassembler_overlapping ..........   Passed    0.01 sec
      Start 15: reassembler_win
14/17 Test #15: reassembler_win ..................   Passed    0.35 sec
      Start 37: compile with optimization
15/17 Test #37: compile with optimization ........   Passed    0.09 sec
      Start 38: byte_stream_speed_test
             ByteStream throughput: 5.03 Gbit/s
16/17 Test #38: byte_stream_speed_test ...........   Passed    0.09 sec
      Start 39: reassembler_speed_test
             Reassembler throughput: 9.69 Gbit/s
17/17 Test #39: reassembler_speed_test ...........   Passed    0.14 sec

100% tests passed, 0 tests failed out of 17

Total Test time (real) =   1.04 sec
Built target check1
kenaz@Kenaz:~/minnow$
```

图 1: passing lab1 tests

三 Challenge

While using map to store the disjoint intervals, I need to take care that the intervals are left-closed and right-open.And I fix bugs in my initialization of the map, since `insert_start` and `newdata` need to be sliced at the beginning of the insert.