Lab 2 Writeup

My name: Zuo, Qikun

My Student number : 201830013

This lab took me about 6 hours to do. I did attend the lab session.

1. Program Structure and Design:

In byte_stream.hh, I define a deque '_buf' to the byte stream which is unread but is assembled. Thus, in the function 'write' in byte_stream.cc, the bytes are pushed to the back of '_buf', and in the function 'read', the bytes are popped from the front of '_buf' after being read. Something which deserves special attention is that the total number of bytes being written to '_buf' can never exceed the value 'first_unaccepted - first_unassembled'. And the total number of bytes being read from '_buf' can never exceed the value 'first_unassembled - first_unread'.

In stream_reassembler.hh, I define an unordered map '_stream' which serves as the auxiliary storage to accommodate the unassembled bytes. And in the function 'push_substring' in stream_reassembler.cc, several different cases should be taken into consideration.

The first case is that, when the 'index' of the substring equals to 'first_assembled', the reassembler needs to call the function 'write' of '_output' to write the valid part (not exceed 'first_unaccepted') of 'data' to '_buf'. And then the reassembler needs to visit the unordered map '_stream' to check whether there are other bytes which could be assembled and written to the '_buf'. That is, they need to form a consecutive sequence and follow the the valid part of 'data' which has been written to '_buf'. And finally, the consecutive sequence (if exists) should also be written to '_buf' and then be erased from '_stream'.

The second case is that, when the 'index' of the substring is bigger than 'first_assembled', then the reassembler only need to insert each byte of the valid part (not exceed 'first_unaccepted') of the substring to '_stream'.

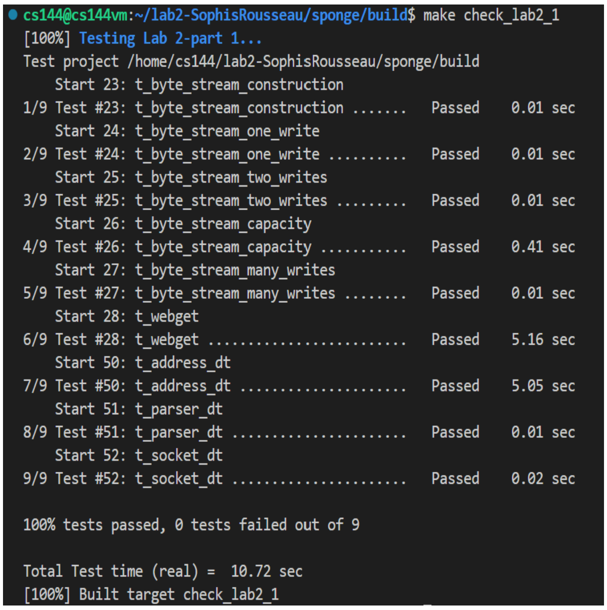
The final case is that, when the 'index' of the substring is smaller than 'first_assembled', there are two sub-cases needs to the be considered. When the index of the final byte of the substring is smaller than 'first_assembled', then the whole substring could be ignored. Otherwise, the valid part (exceed 'first_unassembled') of the substring should be taken into consideration (I use the recursion to implement this sub-case).

The byte stream will end input when receiving 'eof' from the substring and no unassembled bytes are left in '_buf'.

2. Implementation Challenges:

The biggest challenges are designing the data structure to accommodate the unassembled bytes and implementing the function 'push_substring' in stream_reassembler.cc. For the data structure, I first designed a string '_stream' of size '_capacity' and a bool vector '_valid' of size '_capacity' to implement this. The unassembled byte whose index is 'i' is stored at '_stream[i % _capacity]' and its corresponding valid byte '_valid[i % _capacity]' is set to 'true'. When the unassembled bytes are assembled and written to the byte stream, their valid bytes are set to 'false'. Thus it only costs O(1) to store each unassembled byte and check whether an unassembled byte exists according to the index 'i' provided, i.e, if '_valid[i % _capacity]' is true. This data structure makes the function 'push_substring' run fast but its shortcoming is that when '_capacity' is very large, the string and the vector will consume a large amount of storage space while there may exist only a small number of unassembled bytes. For the implementation of the function 'push_substring', because there are several different cases and edge conditions, it takes me quite a long time to consider all the cases in a right way. The picture presented in the tutorial helps me a lot in the implementation of the function.

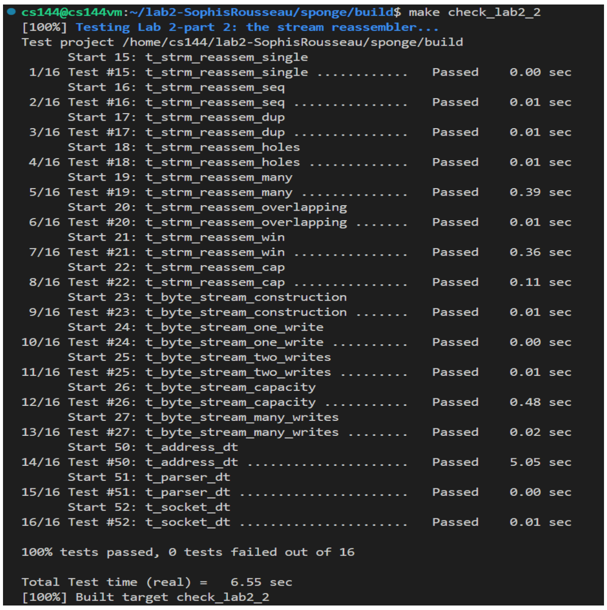
3. Remaining Bugs:



```
cs144@cs144vm:~/lab2-SophisRousseau/sponge/build$ make check_lab2_1
[100%] Testing Lab 2-part 1...
Test project /home/cs144/lab2-SophisRousseau/sponge/build
    Start 23: t_byte_stream_construction
1/9 Test #23: t_byte_stream_construction .......   Passed    0.01 sec
    Start 24: t_byte_stream_one_write
2/9 Test #24: t_byte_stream_one_write ..........   Passed    0.01 sec
    Start 25: t_byte_stream_two_writes
3/9 Test #25: t_byte_stream_two_writes .........   Passed    0.01 sec
    Start 26: t_byte_stream_capacity
4/9 Test #26: t_byte_stream_capacity ...........   Passed    0.41 sec
    Start 27: t_byte_stream_many_writes
5/9 Test #27: t_byte_stream_many_writes ........   Passed    0.01 sec
    Start 28: t_webget
6/9 Test #28: t_webget .........................   Passed    5.16 sec
    Start 50: t_address_dt
7/9 Test #50: t_address_dt .....................   Passed    5.05 sec
    Start 51: t_parser_dt
8/9 Test #51: t_parser_dt ......................   Passed    0.01 sec
    Start 52: t_socket_dt
9/9 Test #52: t_socket_dt ......................   Passed    0.02 sec

100% tests passed, 0 tests failed out of 9

Total Test time (real) =  10.72 sec
[100%] Built target check_lab2_1
```



```
cs144@cs144vm:~/lab2-SophisRousseau/sponge/build$ make check_lab2_2
[100%] Testing Lab 2-part 2: the stream reassembler...
Test project /home/cs144/lab2-SophisRousseau/sponge/build
      Start 15: t_strm_reassem_single
 1/16 Test #15: t_strm_reassem_single ............   Passed    0.00 sec
      Start 16: t_strm_reassem_seq
 2/16 Test #16: t_strm_reassem_seq ...............   Passed    0.01 sec
      Start 17: t_strm_reassem_dup
 3/16 Test #17: t_strm_reassem_dup ...............   Passed    0.01 sec
      Start 18: t_strm_reassem_holes
 4/16 Test #18: t_strm_reassem_holes .............   Passed    0.01 sec
      Start 19: t_strm_reassem_many
 5/16 Test #19: t_strm_reassem_many ..............   Passed    0.39 sec
      Start 20: t_strm_reassem_overlapping
 6/16 Test #20: t_strm_reassem_overlapping .......   Passed    0.01 sec
      Start 21: t_strm_reassem_win
 7/16 Test #21: t_strm_reassem_win ...............   Passed    0.36 sec
      Start 22: t_strm_reassem_cap
 8/16 Test #22: t_strm_reassem_cap ...............   Passed    0.11 sec
      Start 23: t_byte_stream_construction
 9/16 Test #23: t_byte_stream_construction .......   Passed    0.01 sec
      Start 24: t_byte_stream_one_write
10/16 Test #24: t_byte_stream_one_write ..........   Passed    0.00 sec
      Start 25: t_byte_stream_two_writes
11/16 Test #25: t_byte_stream_two_writes .........   Passed    0.01 sec
      Start 26: t_byte_stream_capacity
12/16 Test #26: t_byte_stream_capacity ...........   Passed    0.48 sec
      Start 27: t_byte_stream_many_writes
13/16 Test #27: t_byte_stream_many_writes ........   Passed    0.02 sec
      Start 50: t_address_dt
14/16 Test #50: t_address_dt .....................   Passed    5.05 sec
      Start 51: t_parser_dt
15/16 Test #51: t_parser_dt ......................   Passed    0.00 sec
      Start 52: t_socket_dt
16/16 Test #52: t_socket_dt ......................   Passed    0.01 sec

100% tests passed, 0 tests failed out of 16

Total Test time (real) =   6.55 sec
[100%] Built target check_lab2_2
```

Until now, no bugs are found in the code submitted.