

# 1. Program Structure and Design

## 1. wget 程序：

在 wget 程序中，只需建立TCP连接，并将 http 报文内容写入，并不断接收并输出内容直到 EOF 即可。

## 2. An in-memory reliable byte stream程序：

- 首先要确定的是buffer的数据类型。本人首先考虑的是 string，但对于 string 类型，pop 操作的复杂度为O(n)，效率较低，而用 queue 来作为 buffer，pop 操作的复杂度为O(1)，因此本人最终选择了 queue 作为 buffer 的数据类型。
- 除此之外，bytestream 类还要求记录剩余空间(available\_capacity)和占用空间(byte\_buffered)，而可通过总容量(capacity)与二者之一作差得到另一个，因此只需要记录一个变量即可。由于在定义类时，占用空间已知为0方便初始化，而剩余空间需要知道总容量(capacity)，因此最终选择记录占用空间(bytes\_buffered\_num)。
- Read::pop(uint64\_t len) 的设计：由于本人使用 queue 作为 buffer 的数据类型，因此当删除字符与对列首元素 queue.front() 长度不一样时，不能完全删除首元素，因此本人使用额外的变量 removedBytes\_ 来记录已经被删掉的字符，在 pop() 时需要综合考虑 len 与首元素长度 queue.front()-removedBytes\_ 之间的大小关系来删除元素。

# 2. Implementation Challenges

1. 在 pop() 函数中，需要比较 len 与首元素剩余长度 size 的大小关系，忽略了 len==size 的情况，导致测试样例 byte\_stream\_basics.cc fail，通过clion的调试功能发现并解决了bug。
2. 在 cmake --build build --target check0 运行测试程序时，总会报 AddressSanitizer:DEADLYSIGNAL 错误，在qq群的FAQ文件中找到了解决方法：终端输入 echo 0 | sudo tee /proc/sys/kernel/randomize\_va\_space，禁止linux地址空间布局随机化 ASLR。

# 3. Remaining Bugs

1. 不太清楚的点：不清楚 peek() 函数的功能是提前看多少个字符，在 return 的时候，返回 string\_view {buffer\_.front()}.substr(removedBytes) 可以pass，但返回 string\_view {buffer\_.front().substr(removedBytes\_)} 时，Address Sanitizer 报 stack-use-after-return 错误，可能原因与 string\_view 有关，需要进一步研究。

# 4. Experimental results and performance

1. 手动获取报文

```
wangyuxi@wangyuxi:~/minnow$ telnet cs144.keithw.org http
Trying 104.196.238.229...
Connected to cs144.keithw.org.
Escape character is '^]'.
GET /hello HTTP/1.1
Host:cs144.keithw.org
Connetction:close

HTTP/1.1 200 OK
Date: Fri, 27 Sep 2024 10:10:11 GMT
Server: Apache
Last-Modified: Thu, 13 Dec 2018 15:45:29 GMT
ETag: "e-57ce93446cb64"
Accept-Ranges: bytes
Content-Length: 14
Content-Type: text/plain

Hello, CS144!
```

## 2. 测试截图:

```
wangyuxi@wangyuxi: ~/minnow
wangyuxi@wangyuxi:~/minnow$ echo 0 | sudo tee /proc/sys/kernel/randomize_va_space
0
wangyuxi@wangyuxi:~/minnow$ cmake --build build --target check0
Test project /home/wangyuxi/minnow/build
  Start 1: compile with bug-checkers
1/10 Test #1: compile with bug-checkers ..... Passed    0.29 sec
  Start 2: t_webget
2/10 Test #2: t_webget ..... Passed    1.34 sec
  Start 3: byte_stream_basics
3/10 Test #3: byte_stream_basics ..... Passed    0.02 sec
  Start 4: byte_stream_capacity
4/10 Test #4: byte_stream_capacity ..... Passed    0.02 sec
  Start 5: byte_stream_one_write
5/10 Test #5: byte_stream_one_write ..... Passed    0.02 sec
  Start 6: byte_stream_two_writes
6/10 Test #6: byte_stream_two_writes ..... Passed    0.03 sec
  Start 7: byte_stream_many_writes
7/10 Test #7: byte_stream_many_writes ..... Passed    0.08 sec
  Start 8: byte_stream_stress_test
8/10 Test #8: byte_stream_stress_test ..... Passed    0.15 sec
  Start 37: compile with optimization
9/10 Test #37: compile with optimization ..... Passed    0.17 sec
  Start 38: byte_stream_speed_test
        ByteStream throughput: 10.17 Gbit/s
10/10 Test #38: byte_stream_speed_test ..... Passed    0.14 sec

100% tests passed, 0 tests failed out of 10

Total Test time (real) =  2.25 sec
Built target check0
wangyuxi@wangyuxi:~/minnow$
```