

Eben Stewart CW2 report

by Eben David Stewart

FILE	2145_EBEN_DAVID_STEWART_EBEN_STEWART_CW2_REPORT_113614_381938513.ODT		
TIME SUBMITTED	10-MAY-2018 11:52PM (UTC+0100)	WORD COUNT	3593
SUBMISSION ID	88922174	CHARACTER COUNT	29834

Introduction

This coursework consisted of constructing a chat server/client system from a skeleton of C++ (with some Qt gui elements) provided. It was designed to test ability with and understanding of C++, of Object Oriented program concepts and design, of TCP socket programming, of basic Qt, and of working with multiple threads of program execution, specifically Posix threads and the pthread library functions.

Much of the TCP aspects, the IP address structures and socket manipulation especially, are actually C code wrapped up in C++ classes. Thus the main.cpp of the server could be said to have strong procedural elements to its design and implementation, although it also has object oriented design mixed in. It makes use of a system of external classes which are designed to be highly cohesive and decoupled, but the server's main and the client's ChatClient class (which pretty much functions as a main) are long series of ~~not very cohesive~~ code, procedural in conception that make use of the fore-mentioned classes. An option of the coursework was to redesign the system to be more completely object oriented. Unfortunately time limitation precluded such redesign in this instance.

Structure

Although designed to be used together, the chat server and client are actually completely separate applications. Indeed, there is not much use for a chat system that cannot run numerous independent clients on computers that are remote from each other. The server and its clients maintain the ability to understand and manipulate each other's messages in ways that are predictable to both through the use of a portion of mirrored codebase in the form of a set of identical classes common to both the client and the server software.

Common Code

Both projects contain a base class 'Message' with multiple derived classes for the manipulation, encoding and decoding of different types of message objects. Both also contain a class, called INet4Address, for the creation and formatting of IPv4 addresses through the use of the pertinent member of the AF_INET address family, namely the structure struct_addr_in [1]. Both also contain a class to manage the creation, manipulation and destruction of BSD sockets[1] the creation and pairing of which establishes a connection between separate entities on the internet. In the server, this class is called ServerConn. In the Client it's called Connection.

API

By utilising the common methods for encoding and decoding the various message types, the client and server programs can make sense of what they receive. In this, the two software packages create their own API or Application Programming Interface [3]. Likewise, the identical INet4Address and Connection classes ensure that addresses are structured and that socket pairing is done according to protocols accepted on the wider internet.

Byte Streaming over TCP

The transport protocol involved, TCP, is a streaming protocol which guarantees that messages are received in the order in which they are sent without duplication or loss. It is a two way dedicated protocol involving acknowledgement of receipt of a transmission [4].

TCP streaming involves reading or writing a series of bytes through the socket interface from source machine to destination machine. Of course, these streams are actually broken into small segments and wrapped up into frames with headers in order to satisfy the requirements of the network layer and other, lower layers of the OSI model. But this is abstracted away, so that for the purposes of the transport layer and higher (application layer), data moves from one paired socket to the other as a stream of bytes with no other identifying information.

It follows then, that communicating beforehand the number of bytes to expect is a requisite of properly handling the bytestream. In this Chat system, this is the first thing to be sent. It is handled by the `readSize` and `writeSize` methods of the Client's `Connection` class and the server's `ServerConn` class (`Connection/ServerConn`) which also handle the conversion back and forth of this integer data from host to network byte order (little to big endian) and back. Then the byte stream is sent by the same classes' `readn` and `written` methods. Without further agreed encoding however, this byte stream has no identification to differentiate the types of the data in the stream for the receiving machine to use to make sense of it. Further agreed encoding is put in place to address this problem.

XDR Encoding

The messages are sent encoded as XDR streams which is a standard protocol designed to abstract away differences in architecture and operating system between the sending and receiving machines [5] [6]. Different architectures and or operating systems may define actual sizes in bytes of primitive types such as `int`, `char` or `float` differently.

By agreeing to translate the machine specific types into XDR standard types, and by also agreeing a common format in which data will be sent and received, as is done in the Message derived classes' encoding and decoding methods, the two machines know how to divide and handle the incoming byte stream.

For example, when the chat client program wants to join the conversation, it first sends the type of message that it is, then send the user's nickname which is used by the server to identify that subsequent messages are coming from someone who has asked to join and who is known, at least in the context of the ongoing conversation. The length of the nickname must be sent before the actual name string, so that the receiving machine can allocate memory to receive it. So the `xdr_encode` method of the `Join` class (which formats the join message) encodes first an `xdr_int` for the type, then another `xdr_int` for the string length of the nickname, then the `xdr_string` of the nickname. The `xdr_decode` method of the Server's `join` method knows exactly how many byte of the stream are taken up by each of those components. It then handles each in the correct way.

Actually, it is a little more complicated than that. Although the different types of message are all encoded in different series of `xdr_ints` or strings, the first part of all of them is the `ttype` of the message. Upon receipt of the byte stream, this first `xdr_int` only is first decoded by the Message base class and that info then used to ascertain the message's type and send it to the correct derived class for handling. For the join message, this means comparing the nickname to a list of nicknames held in a vector to see whether it is already in use by another user. If not it is stored in the user list and also prepended to actual chat messages from that user in order to identify their messages throughout the conversation.

Other children of the Message class have `xdr_encode/decode` methods that their purposes.

Function

In the above example, the essential core functionality of the chat system has been described. There is more of course, but full elucidation is beyond the scope of this short paper. In brief then, Join messages join the user to the chat (as above), leave messages remove them from the user list, chat messages contribute a message for display prepended with the nickname as above), Confirm messages confirm the server's acceptance of validation for the three previous types, Request update and Update messages control the display of messages in the chat client's gui. This gui consists of a Qt window and widgets that the user manipulates and gains information from to operate and participate in the program. The chat client initiates messages which are confirmed (or rejected) by the server. All messages occur in pairs when the program functions properly.

The server doesn't need a gui. It is operated from the command line interface. As seen above, it has storage vectors for received messages and a list of active users. When it first starts it creates an IP address structure as described above and sets up a listening thread. When a client connection request is received and accepted, a new socket is created for that connection and a new thread is spawned to handle it in all the ways described above.

One might think that it would be logical for the new thread to handle a continuously maintained connection for all the messages from one user, starting with the join message through the leave message. However, this is not the case. The acceptance of each single message of any type initiates the creation of a new thread which exists only long enough to handle the pair of messages thus initiated and any actions engendered by that exchange. When the Confirm or Update message is completed, the thread is destroyed.

Structured this way, the system is more modular. Each thread does just the single, simple job it's supposed to do and is thus highly cohesive. An interruption in network communication is more easily recovered from when communicating components are thus decoupled.

The threads are ephemeral. Continuity is provided on a user conceptual level by use of the `messageList` vector to keep track of logged in users' nicknames and pairing of those nicknames to displayed messages. There is no need for such continuity for the server software itself. When asked for an update, it sends all the messages since the last update in the order received, from all the users. All the validation it requests is to know that the user is uniquely identifiable. To this end, all client messages send the user's nickname and all of the server replies validate that before accepting or rejecting.

One complication of using Posix threads is in the management of shared resources to prevent race conditions, the corruption of data by different threads accessing and performing non atomic operations upon shared data at inappropriate times, interrupting other threads' operations to do so. This is an artifact of the unpredictability of the operating system's scheduling algorithms. It is remedied by locking access to shared data structures with mutexes before commencing such operations.

There was an error in the codebase for this project in that the client program didn't send the nickname when requesting an automatic timed update from the server. Fixing this involved recoding almost all of the `RequestUpdate` class. This included adding another overloaded constructor the `xdr_encode` and `decode` methods and the `getXDRSize` method as well. Client and server code also had to be adjusted to accommodate.

There are numerous bugs still in the code which there is no time to remove. Consequently, the system does not properly function. However it does build. It is hoped that the essential structure of the skeleton Chat Client and Server that was provided for this assignment has been, if not actually completed, then fleshed out enough to show a fair understanding of the issues and skills etc which this assignment was designed to test.

Appendix

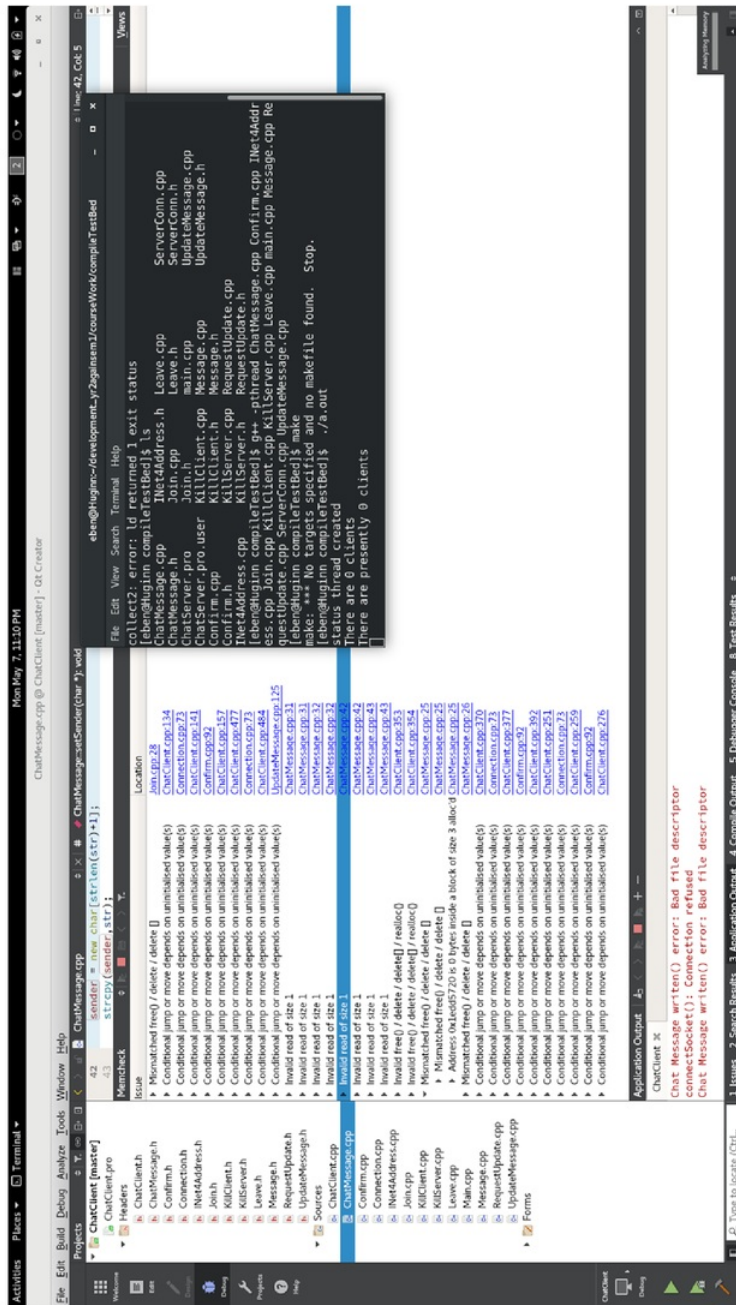


Illustration 1: result of running Valgrind on the client program

References

- [1] W. W. Gay, 'Forming Internet (IPv4) Socket Addresses', in *Linux Socket Programming: by Example*, 1st ed., Indianapolis, Indiana: Que Corp., 2000, pp. 47–55.
- [2] W. W. Gay, *Linux Socket Programming: by Example*. Que Corp., 2000.
- [3] 'What is API - Application Program Interface? Webopedia'. [Online]. Available: <https://www.webopedia.com/TERM/A/API.html>. [Accessed: 10-May-2018].
- [4] 'Transport Layer | Microsoft Docs'. [Online]. Available: [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc977587\(v%3dtechnet.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc977587(v%3dtechnet.10)). [Accessed: 09-May-2018].
- [5] 'xdr_sizeof - man pages section 3: Networking Library Functions'. [Online]. Available: https://docs.oracle.com/cd/E36784_01/html/E36875/xdr-sizeof-3nsl.html. [Accessed: 16-Apr-2018].
- [6] 'RFC 1014 - XDR: External Data Representation standard'. [Online]. Available: <https://tools.ietf.org/html/rfc1014>. [Accessed: 09-May-2018].

Bibliography

- [1]
[Online]. Available: <https://docs.oracle.com/cd/E19253-01/816-1435/rpcguide-2/index.html>.
[Accessed: 05-May-2018].
- [2]
'8.1 The IP address INADDR_ANY'. [Online]. Available: https://www.cs.cmu.edu/~srini/15-441/F01.full/www/assignments/P2/htmlsim_split/node18.html. [Accessed: 18-Apr-2018].
- [3]
'8.12 — Static member functions | Learn C++'. [Online]. Available: <http://www.learncpp.com/cpp-tutorial/812-static-member-functions/>. [Accessed: 25-Apr-2018].
- [4]
'16 bytes in 2 blocks are definitely lost in loss record 1,617 of 9,708 - Google Search'. [Online]. Available:
<https://www.google.com/search?q=16+bytes+in+2+blocks+are+definitely+lost+in+loss+record+1%2C617+of+9%2C708+&ie=utf-8&oe=utf-8&client=firefox-b>. [Accessed: 09-May-2018].
- [5]
Erik Troan, 'Blocking and Non-Blocking I/O', *Linux Magazine*, 15-Sep-1999. [Online]. Available: <http://www.linux-mag.com/id/308/>. [Accessed: 16-Apr-2018].
- [6]
'boolean - In c, in bool, true == 1 and false == 0? - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/40009029/in-c-in-bool-true-1-and-false-0>. [Accessed: 26-Apr-2018].
- [7]
'c - Data Type - socklen_t, sa_family_t', *Stack Overflow*, 15-Sep-2015. [Online]. Available: <https://stackoverflow.com/questions/32592674/data-type-socklen-t-sa-family-t>. [Accessed: 04-Apr-2018].
- [8]
'c - Does memory allocated in a function still stay allocated after the function returns? - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/25567981/does-memory-allocated-in-a-function-still-stay-allocated-after-the-function-retu>. [Accessed: 27-Apr-2018].
- [9]

'c - Does memory allocated in a function still stay allocated after the function returns? - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/25567981/does-memory-allocated-in-a-function-still-stay-allocated-after-the-function-retu>. [Accessed: 27-Apr-2018].

[10]

'c - Is it possible to send the message produced by perror() to /var/log/syslog? - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/34214233/is-it-possible-to-send-the-message-produced-by-perror-to-var-log-syslog>. [Accessed: 18-Apr-2018].

[11]

'c - SIGCHLD Signal Processing - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/9845991/sigchld-signal-processing>. [Accessed: 22-Apr-2018].

[12]

'c - SIGCHLD Signal Processing - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/9845991/sigchld-signal-processing>. [Accessed: 23-Apr-2018].

[13]

'c - undefined reference to pthread_create and pthread_join - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/46750464/undefined-reference-to-pthread-create-and-pthread-join>. [Accessed: 09-May-2018].

[14]

'c - Undefined reference to pthread_create in Linux - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/1662909/undefined-reference-to-pthread-create-in-linux>. [Accessed: 07-May-2018].

[15]

'c - Understanding INADDR_ANY for socket programming - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/16508685/understanding-inaddr-any-for-socket-programming>. [Accessed: 18-Apr-2018].

[16]

'c - Valgrind blocks are definitely lost in loss record - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/26289262/valgrind-blocks-are-definitely-lost-in-loss-record>. [Accessed: 09-May-2018].

[17]

'c - Valgrind: Invalid read of size 1 - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/14067403/valgrind-invalid-read-of-size-1>. [Accessed: 08-May-2018].

[18]

'c - Valgrind: Invalid read of size 1 - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/14067403/valgrind-invalid-read-of-size-1>. [Accessed: 09-May-2018].

[19]

'c - What is the correct way of casting const char* to char* without changing the API and not getting a warning - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/19131965/what-is-the-correct-way-of-casting-const-char-to-char-without-changing-the-api>. [Accessed: 06-Apr-2018].

[20]

'c - When to use pthread_exit() and when to use pthread_join() in Linux? - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/20824229/when-to-use-pthread-exit-and-when-to-use-pthread-join-in-linux>. [Accessed: 29-Apr-2018].

[21]

'c - Why use bzero over memset? - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/17096990/why-use-bzero-over-memset>. [Accessed: 06-Apr-2018].

[22]

- 'C library function - memset()'. [Online]. Available:
https://www.tutorialspoint.com/c_standard_library/c_function_memset.htm. [Accessed: 03-Apr-2018]. [23]
- 'C socket: recv and send all data - Stack Overflow'. [Online]. Available:
<https://stackoverflow.com/questions/13479760/c-socket-recv-and-send-all-data>. [Accessed: 16-Apr-2018]. [24]
- 'C++ - Accepting Multiple Clients - C++ Forum'. [Online]. Available:
<http://www.cplusplus.com/forum/general/91361/>. [Accessed: 19-Apr-2018]. [25]
- 'c++ - Converting QString to char* - Stack Overflow'. [Online]. Available:
<https://stackoverflow.com/questions/5505221/converting-qstring-to-char>. [Accessed: 04-Apr-2018]. [26]
- 'c++ - Difference between std::remove and erase for vector? - Stack Overflow'. [Online]. Available:
<https://stackoverflow.com/questions/19296958/difference-between-stdremove-and-erase-for-vector>. [Accessed: 30-Apr-2018]. [27]
- 'c++ - How do I iterate over a vector and also know the index of the element? - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/12495382/how-do-i-iterate-over-a-vector-and-also-know-the-index-of-the-element>. [Accessed: 30-Apr-2018]. [28]
- 'c++ - How do I remove an item from a stl vector with a certain value? - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/39912/how-do-i-remove-an-item-from-a-stl-vector-with-a-certain-value>. [Accessed: 30-Apr-2018]. [29]
- 'c++ - Iteration over std::vector: unsigned vs signed index variable - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/409348/iteration-over-stdvector-unsigned-vs-signed-index-variable>. [Accessed: 26-Apr-2018]. [30]
- 'c++ - malloc and free vs new and delete - Stack Overflow'. [Online]. Available:
<https://stackoverflow.com/questions/28100404/malloc-and-free-vs-new-and-delete>. [Accessed: 29-Apr-2018]. [31]
- 'c++ - Meaning of (QWidget *parent = 0) in constructor MyClass(QWidget *parent = 0); - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/10591059/meaning-of-qwidget-parent-0-in-constructor-myclassQWidget-parent-0>. [Accessed: 03-Apr-2018]. [32]
- 'c++ - mutexes and locks - Stack Overflow'. [Online]. Available:
<https://stackoverflow.com/questions/1751611/mutexes-and-locks>. [Accessed: 29-Apr-2018]. [33]
- 'c++ - Mutexes in multithread Linux application - Stack Overflow'. [Online]. Available:
<https://stackoverflow.com/questions/6496242/mutexes-in-multithread-linux-application>. [Accessed: 29-Apr-2018]. [34]
- 'c++ - Simple Client-Server multithreading. Error - Stack Overflow'. [Online]. Available:
<https://stackoverflow.com/questions/48749550/simple-client-server-multithreading-error>. [Accessed: 19-Apr-2018]. [35]

'c++ - using exit(1) to return from a function - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/1972953/using-exit1-to-return-from-a-function>. [Accessed: 29-Apr-2018].

[36]

'c++ - What's the difference between sockaddr, sockaddr_in, and sockaddr_in6? - Stack Overflow', *Stack Overflow*, 05-Sep-2013. [Online]. Available: <https://stackoverflow.com/questions/18609397/whats-the-difference-between-sockaddr-sockaddr-in-and-sockaddr-in6>. [Accessed: 03-Apr-2018].

[37]

Gabriel Fleseriu, 'C++ Tutorial: A Beginner's Guide to std::vector, Part 1', *codeguru*, 25-Feb-2003. [Online]. Available: https://www.codeguru.com/cpp/cpp/cpp_mfc/stl/article.php/c4027/C-Tutorial-A-Beginners-Guide-to-stdvector-Part-1.htm. [Accessed: 25-Apr-2018].

[38]

'coding style - Iterate through a C++ Vector using a "for" loop - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/12702561/iterate-through-a-c-vector-using-a-for-loop/12702744>. [Accessed: 26-Apr-2018].

[39]

'Data Flow in the OSI Model', *TechNet*. [Online]. Available: <https://technet.microsoft.com/en-us/library/cc977591.aspx>. [Accessed: 22-Mar-2018].

[40]

'delphi - How do you protect a common resource using mutexes? - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/2470921/how-do-you-protect-a-common-resource-using-mutexes>. [Accessed: 29-Apr-2018].

[41]

'difference between exit 1 and exit'. [Online]. Available: <https://www.linuxquestions.org/questions/programming-9/difference-between-exit-1-and-exit-374707/>. [Accessed: 18-Apr-2018].

[42]

'Find Qt version: command line, compile time, run time | The Linux Page'. [Online]. Available: <https://linux.m2osw.com/find-qt-version-command-line-compile-time-run-time>. [Accessed: 07-Apr-2018].

[43]

'Five pitfalls of Linux sockets programming', *IBM - developerWorks®*. [Online]. Available: <https://www.ibm.com/developerworks/library/l-sockpit/>. [Accessed: 18-Apr-2018].

[44]

W. W. Gay, 'Forming Internet (IPv4) Socket Addresses', in *Linux Socket Programming: by Example*, 1st ed., Indianapolis, Indiana: Que Corp., 2000, pp. 47–55.

[45]

'How do UNIX linkers work.' [Online]. Available: <http://webpages.charter.net/ppluzhnikov/linker.html>. [Accessed: 07-May-2018].

[46]

'How do UNIX linkers work.' [Online]. Available: <http://webpages.charter.net/ppluzhnikov/linker.html>. [Accessed: 09-May-2018].

[47]

'how strtcat works - C++ Forum'. [Online]. Available: <http://www.cplusplus.com/forum/beginner/28063/>. [Accessed: 03-May-2018].

[48]

'How to find out the return value if my C++ "new" memory allocation failure? - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/29283847/how-to-find-out-the-return-value-if-my-c-new-memory-allocation-failure>. [Accessed: 22-Apr-2018].

[49]

- 'How to navigate through a vector using iterators? (C++) - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/2395275/how-to-navigate-through-a-vector-using-iterators-c>. [Accessed: 26-Apr-2018]. [50]
- 'htons - convert values between host and network - Linux Man Pages (3)', *Linux Man Pages*. [Online]. Available: <https://www.systutorials.com/docs/linux/man/3-htons/>. [Accessed: 03-Apr-2018]. [51]
- 'IBM Knowledge Center - pthread_exit()--Terminate Calling Thread'. [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/ssw_i5_54/apis/users_18.htm. [Accessed: 04-May-2018]. [52]
- 'IBM Knowledge Center - PTON'. [Online]. Available: https://www.ibm.com/support/knowledgecenter/en/SSLTBW_2.1.0/com.ibm.zos.v2r1.hali001/ptonall.htm. [Accessed: 03-Apr-2018]. [53]
- 'inet_pton - convert IPv4 and IPv6 addresses from text to binary form - Linux Man Pages (3)'. [Online]. Available: https://www.systutorials.com/docs/linux/man/3-inet_pton/. [Accessed: 04-Apr-2018]. [54]
- Y. Lou, 'Introduction to C - Threads', presented at the CS 2022, Spring 2011, Lecture 11, Cornell University, 16-Feb-2011. [55]
- 'invalid conversion from 'int*' to 'socklen_t*'. [Online]. Available: https://www.linuxquestions.org/questions/programming-9/invalid-conversion-from-%60int%2A%27-to-%60socklen_t%2A%27-145992/. [Accessed: 07-May-2018]. [56]
- 'ip(7) - Linux manual page'. [Online]. Available: <http://man7.org/linux/man-pages/man7/ip.7.html>. [Accessed: 03-Apr-2018]. [57]
- 'Is there a performance difference between i++ and ++i in C++? - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/24901/is-there-a-performance-difference-between-i-and-i-in-c>. [Accessed: 30-Apr-2018]. [58]
- 'Iterate through vectors - C++ Forum'. [Online]. Available: <http://www.cplusplus.com/forum/beginner/121511/>. [Accessed: 26-Apr-2018]. [59]
- 'least significant - Google Search'. [Online]. Available: https://www.google.co.uk/search?q=least+significant&ie=utf-8&oe=utf-8&client=firefox-b&gfe_rd=cr&dcr=0&ei=ybHDWuyABrTP8Aft1IDoCg. [Accessed: 03-Apr-2018]. [60]
- 'linker problem for qt5 - qt4 runs ok -> undefined reference to 'QWidget::styleChange'. [Online]. Available: <http://www.qtcentre.org/threads/67029-linker-problem-for-qt5-qt4-runs-ok-gt-undefined-reference-to-QWidget-styleChange>. [Accessed: 07-May-2018]. [61]
- 'linux - invalid conversion from int to socklen - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/9197689/invalid-conversion-from-int-to-socklen>. [Accessed: 07-May-2018]. [62]
- W. W. Gay, *Linux Socket Programming: by Example*. Que Corp., 2000. [63]

- Alfred Park, 'Multithreaded Programming (POSIX pthreads Tutorial)', *Multithreaded Programming*. [Online]. Available: <https://randu.org/tutorials/threads/>. [Accessed: 25-Apr-2018]. [64]
- S. Microsystems and S. Clara, 'Multithreaded Programming Guide', p. 268. [65]
- 'multithreading - How to use mutexes in C - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/7303372/how-to-use-mutexes-in-c>. [Accessed: 28-Apr-2018]. [66]
- 'Mutex vs Semaphore - GeeksforGeeks'. [Online]. Available: <https://www.geeksforgeeks.org/mutex-vs-semaphore/>. [Accessed: 28-Apr-2018]. [67]
- 'operator new - C++ Reference'. [Online]. Available: <http://www.cplusplus.com/reference/new/operator%20new/>. [Accessed: 20-Apr-2018]. [68]
- 'perror - C++ Reference'. [Online]. Available: <http://www.cplusplus.com/reference/cstdio/perror/>. [Accessed: 17-Apr-2018]. [69]
- 'pointers - Behaviour of delete in C++ vs. free in C - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/19917667/behaviour-of-delete-in-c-vs-free-in-c>. [Accessed: 28-Apr-2018]. [70]
- 'pthread_cancel(3) - Linux manual page'. [Online]. Available: http://man7.org/linux/man-pages/man3/pthread_cancel.3.html. [Accessed: 25-Apr-2018]. [71]
- 'pthread_create site:man7.org/linux/man-pages - Google Search'. [Online]. Available: https://www.google.com/search?q=pthread_create&site=man7.org%2Flinux%2Fman-pages&sa=Search+online+pages. [Accessed: 25-Apr-2018]. [72]
- Michael Kerrisk, 'pthread_create(3)', *Linux manual page*, 15-Sep-2017. [Online]. Available: http://man7.org/linux/man-pages/man3/pthread_create.3.html. [Accessed: 25-Apr-2018]. [73]
- Michael Kerrisk, 'pthread_detach(3) - Linux manual page', 15-Sep-2017. [Online]. Available: http://man7.org/linux/man-pages/man3/pthread_detach.3.html. [Accessed: 25-Apr-2018]. [74]
- 'pthread_exit'. [Online]. Available: http://pubs.opengroup.org/onlinepubs/7908799/xsh/pthread_exit.html. [Accessed: 29-Apr-2018]. [75]
- 'PTHREAD_MUTEX_INIT(3) manual page'. [Online]. Available: https://www.sourceware.org/pthreads-win32/manual/pthread_mutex_init.html. [Accessed: 29-Apr-2018]. [76]
- 'pthread_mutex_lock'. [Online]. Available: http://pubs.opengroup.org/onlinepubs/009695399/functions/pthread_mutex_lock.html. [Accessed: 29-Apr-2018]. [77]
- 'pthread_mutex_lock(3): lock/unlock mutex - Linux man page'. [Online]. Available: https://linux.die.net/man/3/pthread_mutex_lock. [Accessed: 29-Apr-2018]. [78]
- 'pthread_self(3) - Linux manual page'. [Online]. Available: http://man7.org/linux/man-pages/man3/pthread_self.3.html. [Accessed: 24-Apr-2018]. [79]

- 'pthread(7) - Linux manual page'. [Online]. Available: <http://man7.org/linux/man-pages/man7/pthreads.7.html>. [Accessed: 25-Apr-2018]. [80]
- 'QByteArray Class | Qt Core 5.10'. [Online]. Available: <https://doc.qt.io/qt-5/qbytearray.html>. [Accessed: 05-Apr-2018]. [81]
- 'QString Class | Qt Core 5.10'. [Online]. Available: <https://doc.qt.io/qt-5/qstring.html#toLatin1>. [Accessed: 05-Apr-2018]. [82]
- 'QString Class | Qt Core 5.10'. [Online]. Available: <https://doc.qt.io/qt-5/qstring.html#isEmpty>. [Accessed: 05-May-2018]. [83]
- 'qt - How to use QTimer - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/11651852/how-to-use-qtimer>. [Accessed: 05-May-2018]. [84]
- 'qt - QByteArray convert to/from unsigned char * - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/8939935/qbytearray-convert-to-from-unsigned-char>. [Accessed: 05-Apr-2018]. [85]
- 'QTimer Class | Qt Core 5.7'. [Online]. Available: <https://doc.qt.io/archives/qt-5.7/qtimer.html#interval-prop>. [Accessed: 05-May-2018]. [86]
- 'RFC 1014 - XDR: External Data Representation standard'. [Online]. Available: <https://tools.ietf.org/html/rfc1014>. [Accessed: 09-May-2018]. [87]
- 'send'. [Online]. Available: <http://pubs.opengroup.org/onlinepubs/009696899/functions/send.html>. [Accessed: 16-Apr-2018]. [88]
- 'send(2): send message on socket - Linux man page'. [Online]. Available: <https://linux.die.net/man/2/send>. [Accessed: 16-Apr-2018]. [89]
- 'SIGCHLD handler | Linux Programming Blog'. [Online]. Available: <https://www.linuxprogrammingblog.com/code-examples/SIGCHLD-handler>. [Accessed: 22-Apr-2018]. [90]
- 'std::vector::at - cppreference.com'. [Online]. Available: <http://en.cppreference.com/w/cpp/container/vector/at>. [Accessed: 07-May-2018]. [91]
- T. Macieira, 'String Theory - Qt Blog', *Qt*, 28-Apr-2008. [Online]. Available: <https://blog.qt.io/blog/2008/04/28/string-theory/>. [Accessed: 06-Apr-2018]. [92]
- 'struct sockaddr and pals'. [Online]. Available: https://beej.us/guide/bgnet/html/multi/sockaddr_inman.html. [Accessed: 03-Apr-2018]. [93]
- 'struct sockaddr_in, struct in_addr'. [Online]. Available: https://www.gta.ufrj.br/ensino/ee1878/sockets/sockaddr_inman.html. [Accessed: 03-Apr-2018]. [94]
- 'The Linux kernel: Signals'. [Online]. Available: <https://www.win.tue.nl/~aeb/linux/lk/lk-5.html>. [Accessed: 23-Apr-2018]. [95]

'The New C Standard: 6.5.3.4 The sizeof operator', 30-Jan-2008. [Online]. Available: <http://c0x.coding-guidelines.com/6.5.3.4.html>. [Accessed: 04-Apr-2018].

[96]

'Transport Layer | Microsoft Docs'. [Online]. Available: [https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc977587\(v%3dtechnet.10\)](https://docs.microsoft.com/en-us/previous-versions/windows/it-pro/windows-2000-server/cc977587(v%3dtechnet.10)). [Accessed: 09-May-2018].

[97]

'Valgrind Home'. [Online]. Available: <http://www.valgrind.org/>. [Accessed: 30-Apr-2018].

[98]

'vector - C++ Reference', *cplusplus.com*. [Online]. Available: <http://www.cplusplus.com/reference/vector/vector/>. [Accessed: 25-Apr-2018].

[99]

'Vector in C++ STL', *GeeksforGeeks*. [Online]. Available: <https://www.geeksforgeeks.org/vector-in-cpp-stl/>. [Accessed: 25-Apr-2018].

[100]

'Vector in C++ STL - GeeksforGeeks'. [Online]. Available: <https://www.geeksforgeeks.org/vector-in-cpp-stl/>. [Accessed: 26-Apr-2018].

[101]

'vector iterators - C++ Forum'. [Online]. Available: <http://www.cplusplus.com/forum/beginner/64854/>. [Accessed: 29-Apr-2018].

[102]

'vector::begin - C++ Reference'. [Online]. Available: <http://www.cplusplus.com/reference/vector/vector/begin/>. [Accessed: 26-Apr-2018].

[103]

'vector::operator[] - C++ Reference'. [Online]. Available: [http://net.pku.edu.cn/~course/cs101/2007/resource/www.cplusplus.com/reference/stl/vector/operator\[\].html](http://net.pku.edu.cn/~course/cs101/2007/resource/www.cplusplus.com/reference/stl/vector/operator[].html). [Accessed: 07-May-2018].

[104]

'vector::push_back - C++ Reference'. [Online]. Available: http://www.cplusplus.com/reference/vector/vector/push_back/. [Accessed: 26-Apr-2018].

[105]

'waitpid(2) (man pages section 2: System Calls)'. [Online]. Available: <https://docs.oracle.com/cd/E19455-01/806-0626/6j9vgh6ap/index.html>. [Accessed: 22-Apr-2018].

[106]

'Well-Known TCP/IP Port Numbers - Meridian Outpost'. [Online]. Available: <http://www.meridianoutpost.com/resources/articles/well-known-tcpip-ports.php>. [Accessed: 07-Apr-2018].

[107]

'What are the differences between a pointer variable and a reference variable in C++? - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/57483/what-are-the-differences-between-a-pointer-variable-and-a-reference-variable-in?rq=1>. [Accessed: 28-Apr-2018].

[108]

'What is API - Application Program Interface? Webopedia'. [Online]. Available: <https://www.webopedia.com/TERM/A/API.html>. [Accessed: 10-May-2018].

[109]

'What is SIGCHLD in Linux? - Quora'. [Online]. Available: <https://www.quora.com/What-is-SIGCHLD-in-Linux>. [Accessed: 22-Apr-2018].

[110]

'Who handles C++ "new" memory allocation failure? - Stack Overflow'. [Online]. Available: <https://stackoverflow.com/questions/29262078/who-handles-c-new-memory-allocation-failure?noredirect=1&lq=1>. [Accessed: 22-Apr-2018].

[111]
'Why doesn't new in C++ return NULL on failure - Stack Overflow'. [Online]. Available:
<https://stackoverflow.com/questions/26419786/why-doesnt-new-in-c-return-null-on-failure>.
[Accessed: 22-Apr-2018].

[112]
'xdr_proc_t - Google Search'. [Online]. Available:
https://www.google.com/search?q=xdr_proc_t&client=firefox-b&tbm=isch&tbo=u&source=univ&sa=X&ved=0ahUKEwie-PO8uO_aAhVeF8AKHZ1CAkUQsAQILQ&biw=1920&bih=888#imgsrc=Mr-dugt6eHXdCM:
[Accessed: 05-May-2018].

[113]
'xdr_sizeof - man pages section 3: Networking Library Functions'. [Online]. Available:
https://docs.oracle.com/cd/E36784_01/html/E36875/xdr-sizeof-3nsl.html. [Accessed: 16-Apr-2018].

[114]
'xdr(3) - Linux manual page'. [Online]. Available: <http://man7.org/linux/man-pages/man3/xdr.3.html>. [Accessed: 16-Apr-2018].

Eben Stewart CW2 report

GRADEMARK REPORT

FINAL GRADE

GENERAL COMMENTS

Instructor

49/100

PAGE 1

PAGE 2

PAGE 3

PAGE 4

PAGE 5

PAGE 6

PAGE 7

PAGE 8

PAGE 9

PAGE 10

PAGE 11

PAGE 12

PAGE 13

GRADING FORM: SMU GENERIC_GJ

EBEN DAVID STEWART

49

RESEARCH-INFORMED LITERATURE

-
- | | | |
|---|--|----------|
|  | An appropriately-sized reference list and an outstanding number of bibliography sources - I've seen fewer in a PhD thesis! The report generally is good. | 4 |
|---|--|----------|

KNOWLEDGE AND UNDERSTANDING

- | | | |
|---|---|----------|
|  | Reasonable evidence of knowledge and understanding, in the context of the work not quite being finished. A good attempt made. | 6 |
|---|---|----------|

ANALYSIS

- | | | |
|---|--|----------|
|  | Some reasonable synthesis, with conclusions drawn. | 5 |
|---|--|----------|

PRACTICAL APPLICATION AND DEP.

- | | | |
|---|---|-----------|
|  | As you state in the report, significant progress has been made - you didn't quite get there, but a very valid attempt made. | 31 |
|---|---|-----------|

SKILLS FOR PROFESSIONAL PRAC.

- | | | |
|---|--|----------|
|  | A well-written and presented report. Avoid contractions in academic writing. | 3 |
|---|--|----------|