Vrije Universiteit Amsterdam
Computer Science Dept.
Computer Networks
Lecturer: Jesse Donkervliet

A TELNET Client in C/C++

Author: Andrea Di Dio
Group/Student Number: Group 42/2593888

March 22, 2018

Contents

1 Introduction

I have decided to implement the TELNET Client as a bonus assignment because I felt that getting my "hands on" a real-life (even though not used as much today) Application layer protocol would have been a nice challenge. I have decided to implement the program in C++ with a few "fallbacks" into C. I made this decision because I wanted the "Object-Oriented" approach possible in C++ but wanted to use certain C system calls which were useful for the final outcome of the assignment. In this document I will walk through my thought process as to why I decided to use certain functions and I will also report the Errata and problems that exist along with the challenges I encountered during the implementation. Usage of the program:

```
./telnet {Hostname} {Port Number}
```

2 Brief Demonstration of Command Negotiations

In order to understand better the TELNET protocol I used *PUTTY* to send some requests to TELNET endpoints provided and analysed how the negotiation process worked by capturing the packets on *Wireshark*. Once I implemented my own client, I wanted to be sure that I was negotiating options and suboptions with the server correctly and hence I once again captured packets via Wireshark to be certain. I hereby report an example of captured packets from Wireshark when using my telnet client to connect to the remote host "telehack.com:23".

Figure 1: Server request to initiate negotiation.

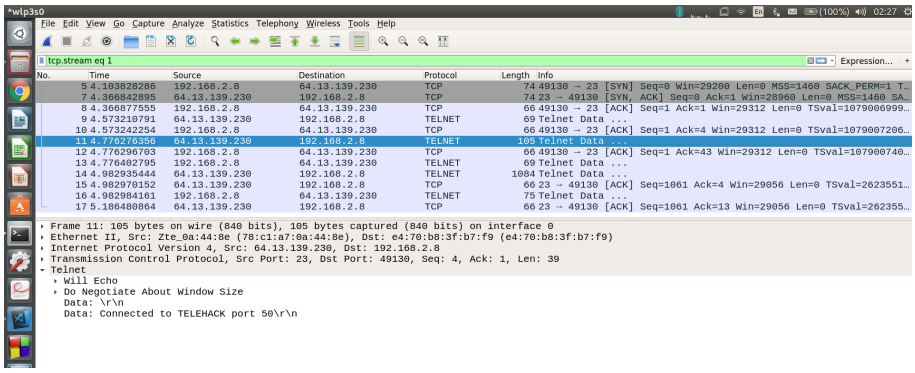
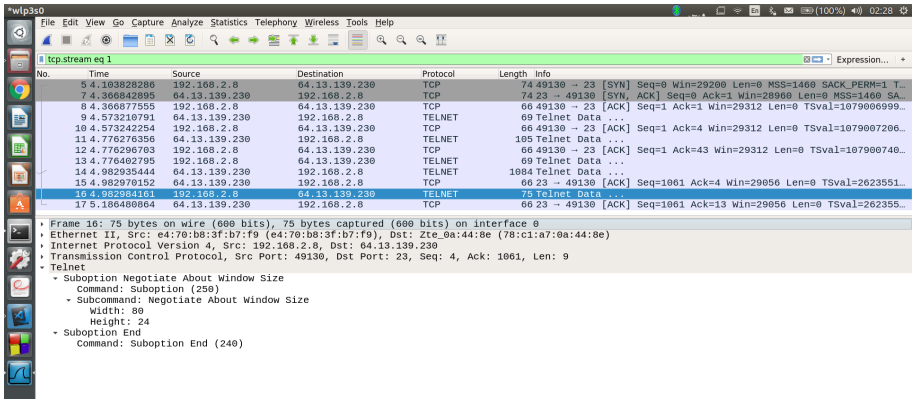


Figure 2: Client response to negotiation.



3 Difficulties Encountered

Whilst implementing the TELNET protocol, I have encountered some difficulties which I will discuss now. Firstly, The program when receiving data from the server was outputting *"weird characters"* but this was a minor problem as I was using normal (signed) chars to receive data without taking into account that the option negotiations are unsigned chars. I fixed this by using unsigned chars for my buffer.However, when printing out to the console in order to avoid printing weird chars, I had to hard-code null-termination of the string by adding '\0'. Secondly, when testing my program after I implemented what I thought was enough, I encountered a deadlock situation. When the server sent all the data it had to send, it was waiting for some input, however, the client was waiting for more data from the server (Ouch!). After searching the Web for some similar problems, i realised that some kind of synchronisation between the client-server I/O operations was in order and solved this problem using the *pselect()*[1] system call and the *fd_set* typedef[4] in order to work with the socket file descriptor and the stdin. Lastly, as the terminal was still not accepting input correctly, I came to the conclusion that I had to "spawn" a new raw terminal in order to interact with the server. This works in most cases, however some telnet endpoints require non-raw ("cooked") terminal which means that the input will not be visible in the terminal and the '\n' will have to be inputted by both pressing "Enter" and "Ctrl-J". This happens with SMTP as well. I will include a list of endpoints with such inconveniences in the tables of Section 5.

4 Code Decisions

4.1 MAX_BUFFER_SIZE

In the definition of MAX_BUFFER_SIZE I decided to use 1023 chars because according to the linux man page[1], if the size of the buffer is >= to FD_SETSIZE which is 1024 in the pselect() definition, it might cause undefined behaviour. Because I captured packets on Wireshark, I saw that 1023 bytes was still enough to process all the data received from the server.

4.2 makeTemporaryTerminal()

When "spawning" a new terminal, I decided to make it a raw terminal because it disables and enables certain flags of the termios struct that were useful to interact with some of the TELNET endpoints.However, the original raw terminal made with "cfmakeraw()" also disables Signal Interrupts which I wanted to keep, so I appended it to the bitmask for the local flags. This adds the possibility for the user to interrupt the process for example, through the use of "Ctrl-C" or "Ctrl-Z".[2], [3]

4.3 checkForData(fd_set& sync, struct timespec& timeout)

This function is used to synchronise the file descriptors in use (socket and stdin) between client and server. As I said in Section 3, this was used to avoid the deadlocking situation. I used pselect() instead of the more common select() because I wanted to have the timespec struct and not the timeval struct because the latter gets modified by the file descriptor, and

I did not want that as I wanted a constant timeout period of 10 seconds. See subsection 4.5 to see how I selected file descriptors.[4]

4.4 negotiate(unsigned char input[ ])

This function is used to negotiate some of the options with the remote host. When capturing packets on Wireshark, I noted what the most used options were and implemented those(TTYPE, SGA etc...). If other options are trying to be negotiated, the client will respond "WON'T" to every server "DO" and "DONT" to every "WILL". This is to avoid leaving negotiations pending and by rejecting them, the TELNET NVT should adapt. So whenever I receive a negotiation that I have not implemented, the client will reject it.

4.5 fd_set

In order to test file descriptors and make a set of these in order to check when they are "ready", I used four macros from the <sys/select.h> library.

```
1 FD_ZERO(&sync);
2 FD_SET(socket.getSocket(), &sync);/*adds to set file descriptor as socket
    to check if data has to be received*/
3 FD_SET(0, &sync);//adds to set file descriptor to 0 (stdin)
4 FD_ISSET(socket.getSocket(), &sync)/*checks if sock fd is in set*/
5 FD_ISSET(0, &sync)/*checks if stdin is in set*/
6 FD_CLR(socket.getSocket(), &sync);/*removes sock fd from set temporarily
   */
```

5 Testing With Endpoints

TELNET		
Endpoint	Working?	Remarks
rainmaker.wunderground.com:23	YES(+)	When connecting to this host, input is not shown and to put a '\n', you have to press "Enter" + "Ctrl-J"
nyancat.dakko.us:23	YES(++)	Works as expected. Press "Ctrl-C" to stop animation
mapscii.me:23	YES(++)	Works as expected
india.colorado.edu:13	YES(++)	Works as expected
telnet.wmflabs.org:23	NO(–)	After outputting the start screen, doesn't free stdin and just closes connection remotely.
telehack.com:23	YES(++)	Works as expected
freechess.org:5000	YES(+)	When connecting to this host, input is not shown and to put a '\n', you have to press "Enter" + "Ctrl-J"
towel.blinkenlights.nl:23	YES(++)	Works as expected
mtrek.com:1701	YES(++)	Works as expected (did not try playing though)

Table 1: Miscellaneous Endpoint Testing

TELNET		
Endpoint	Working?	Remarks
bbs.archaicbinary.net:23	NO(-/+)	Needed to negotiate more options to show the correct "GUI" but functionality works
ateraan.com:4002	YES(+)	When connecting to this host, input is not shown and to put a '\n', you have to press "Ctrl-J"
avalon-rpg.com:23	YES(+)	When connecting to this host, input is not shown but "Enter" works
aardmud.org:4000	YES(+)	When connecting to this host, input is not shown but "Enter" works
bbs.armageddonbbs.com:23	NO(–)	Cannot connect to remote host.
52.88.68.92:1234	YES(+)	When connecting to this host, input is not shown but "Enter" works
TextMMOde.com:23	NO(–)	Cannot connect to remote host.
thehatshop.mudhosting.net:3000	YES(+)	When connecting to this host, input is not shown and to put a '\n', you have to press "Enter" + "Ctrl-J"
batmud.bat.org:23	YES(+)	When connecting to this host, input is not shown but "Enter" works

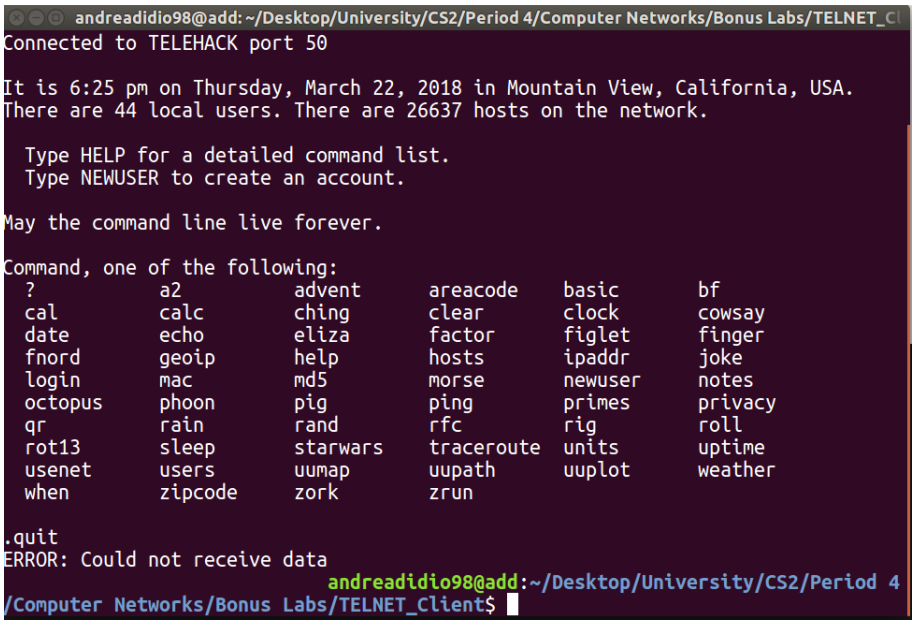
Table 2: Muds, Talkers, BBS, and other systems Endpoint Testing

With SMTP (:25) and HTTP(:80) The problem of the raw terminal acts up meaning that the input in "real-time" is invisible and to input '\n', you have to hit "Enter" + "Ctrl-J". However, it is functional. SMTP was tested with mail.port25com:25 and HTTP/1.1 was tested with various hosts.

6 Errata and Strange Behaviour

As i have already discussed, there are some problems with the raw terminal mode that works well with some endpoints, however has strange behaviour with others such as not showing the input in "real-time" and not "catching" the '\n' character. In addition, When the program exits my terminal remains into raw mode and hence in order to restart the program the terminal has to be killed and reopened because it shows some strange indentations as in the figure below. This does not happen if instead of quitting within the terminal endpoint we interrupt the process with "Ctrl-C".

Figure 3: Strange behaviour of the terminal.



-THE END-

Bibliography

[1] https://linux.die.net/man/3/pselect
[2] http://man7.org/linux/man-pages/man3/termios.3.html
[3] https://linux.die.net/man/3/cfmakeraw
[4] https://linux.die.net/man/3/fd_set
[5] http://mars.netanya.ac.il/ unesco/cdrom/booklet/HTML/NETWORKING/node300.html
[6] http://pubs.opengroup.org/onlinepubs/7908799/xsh/termios.h.html
[7] https://www.telnet.org/htm/places.htm
[8] https://www.gnu.org/software/libc/manual/html_node/Waiting-for-I_002fO.html
[9] http://www.tcpipguide.com/free/t_TelnetOptionsandOptionNegotiation.htm
[10] http://www.linuxhowtos.org/C_C++/socket.htm
[11] http://www.retran.com/beej/index.html
[12] http://users.cs.cf.ac.uk/Dave.Marshall/Internet/node141.html
[13] https://tools.ietf.org/html/rfc854