CS114 - Homework 1, part 1[*]

Assigned January 18th, 2023; Due 11:59pm on February 2nd, 2023

Prof. Daniel Votipka

1 A Simple, Unencrypted P2P Instant Messenger {60 points}

To introduce you to network programming, you will build a simple unencrypted instant messenger. The program reads from standard input and sends input messages to another instance of the program running on a different machine; received messages are sent to standard output.

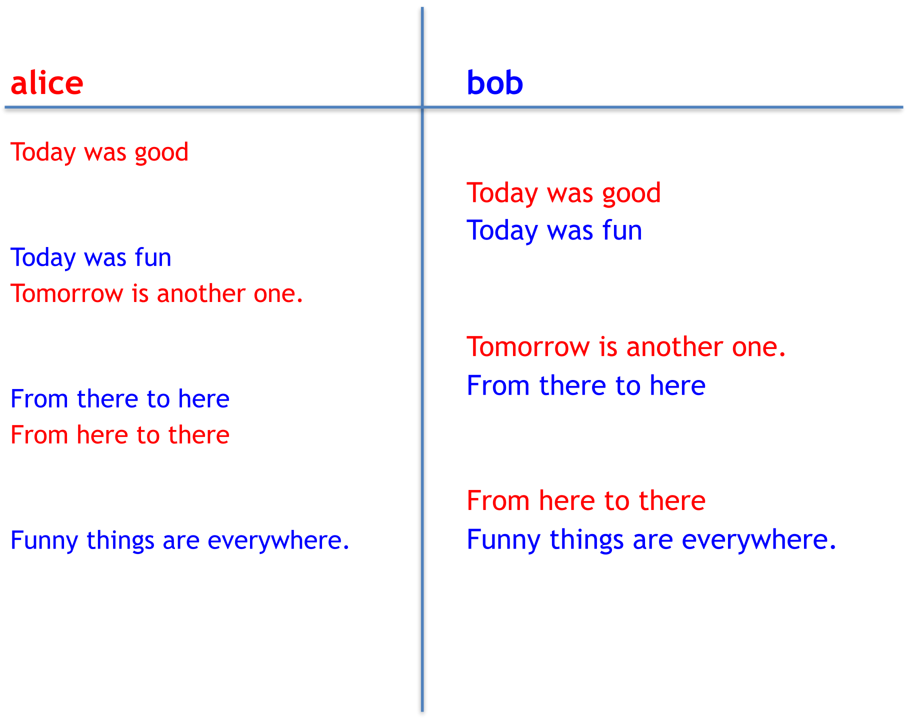

Figure 1: An example conversation between two nodes, as seen by each of the two nodes. Red text is entered by the user at program instance alice and blue text is entered by the user at program instance bob.

Figure 1 shows a conversation between two running instances of the IM client , alice and bob.

[*]Last revised on January 30, 2023.

Your IM client should use TCP to send messages between hosts. To test your code, you can run two instances of the program locally on your own computer that communicate over the localhost (127.0.0.1) interface. If you use a Unix-based OS (i.e., Ubuntu, Fedora, Mac, etc) you can simply install python (the autograder will use python 3.6.9) open two **Terminal** instances, and run one instance of the program to listen in one window and have the other instance connect from the other window.

If you are not using a Unix OS (i.e., Windows), we recommend you download a premade 64-bit Kali Linux VM from https://www.offensive-security.com/kali-linux-vm-vmware-virtualbox-image-download/ This VM has all the tools you'll need (and more) installed—e.g., python, Wireshark, nmap, tcpdump, etc. You can download a VM for use with VMWare Workstation or VirtualBox. Virtual box is free and open source. VMWare Workstation (or Fusion for MacOS) is a commercial product, but you can get a free one year license via Tufts CS at https://vmap-tufts.onthehub.com/WebStore/Welcome.aspx

To create a new virtual machine in VMWare Workstation, follow these instructions.

To open an instance of the **Terminal** in the provided VM, first click on *Activities* in the top left corner of the screen. That will show a search bar at the top of the screen. Type *Terminal* into the search bar and select the icon labeled *Terminal*. This will open a UNIX command prompt that you can run commands from.

Program description. Your program should read from standard input and send all input data to the other instance of your application (running on the other host), via TCP/IP over port 9999.

Here's the tricky bit: Received messages should be *immediately* written to standard output [1]. To do this, you will need to use the *select* call to block and wait for input *either* on standard input or the network socket. Descriptions of Python's `select` call are available online.

Your program **must** be called `unencryptedim.py`.

Your program should have the following command-line options:

```
    unencryptedim.py --s|--c hostname
```

where the `--s` argument indicates that the program should wait for an incoming TCP/IP connection on port 9999; the `--c` argument (with its required `hostname` parameter) indicates that the program should connect to the machine `hostname` (over TCP/IP on port 9999).

For example, you may run "`python unencryptedim.py --s`" on *netid*-alice, and then start "`python unencryptedim.py --c netid-alice`" on *netid*-bob. Note that the instance with the `-s` option must be started before the other instance.

[1] Make sure you flush stdout (`sys.stdout.flush()`). Otherwise, the autograder might hang.

Additional requirements and hints. Please make sure that your program conforms to the following:

- You should strongly consider using the argparse module to parse the command-line.
- You must write your program in Python. To aid with automated grading, your program must be called `unencryptedim.py`.
- You may only use libraries already installed on *netid*-alice and *netid*-bob. *Hint:* For this assignment, you should only need to import the `argparse` (or `optparse`), `select`, `socket`, and `sys` modules. If you want to get fancy, you may also import `signal` (i.e., to trap CTRL-C keypresses).
- **You may not collaborate on this homework. This project should be done individually.** You may use the Python documentation pages (`https://docs.python.org/`) as a programming language reference, and may look at any Python programming tutorial. However, you may not base your code off of online tutorials that essentially "solve" this programming assignment. Please consult the teaching staff via a **private** Piazza note if you are unsure whether a particular online resource is allowed.
- To aid in automated testing/grading, do not provide a prompt to the user, and only write received messages to standard out. (Text entered into standard input can also be shown; see Figure 1.) We will be using automated testing tools to evaluate your solutions, and printing additional messages or characters makes such automation far more difficult.
- Your program should not take in any additional command-line options other than the `--s` or `--c hostname` options described above.
- It is OK if messages are only sent after the user presses [ENTER] after entering a line of text. However, incoming messages should be displayed immediately after they are received by the kernel.
- Your program can terminate either when the user presses CTRL-C, or when end-of-file (EOF) is received. To generate EOF from the terminal, press CTRL-D.
- Hint: To stress test your code, try using your program to copy a file between machines. You should be able to do this by redirecting standard input (at one end) and standard output (at the other).

In part II of this homework (assigned shortly after the due date!), we will add a layer of encryption to our IM applications.

Grading

This portion of HW1 is worth 60 points. A non-comprehensive list of deductions is provided in Table 1.

We will award partial credit when possible and appropriate. To maximize opportunities for partial credit, please rigorously comment your code. If we cannot understand what you intended, we cannot award partial credit.

Tests	**Scoring**
IMs are successfully transmitted	+15
IMs are successfully received	+15
Client and receiver can successfully communicate back and forth	+20
Passes complex input tests	+10
Deductions	
Compilation / interpreter errors	-30
Received messages only appear after user presses [ENTER] (indicates that *select* is used improperly)	-10
Non-conformant command-line options (hinders automated testing)	-5
Includes unnecessary prompts (hinders automated testing)	-3

Table 1: Grading rubric. Note that this grading rubric is not intended to be comprehensive.

Submission Instructions

Submit your solution to Gradescope. Upload your assignment before 11:59pm on February 2nd.

Please post questions (especially requests for clarification) about this homework to Piazza.