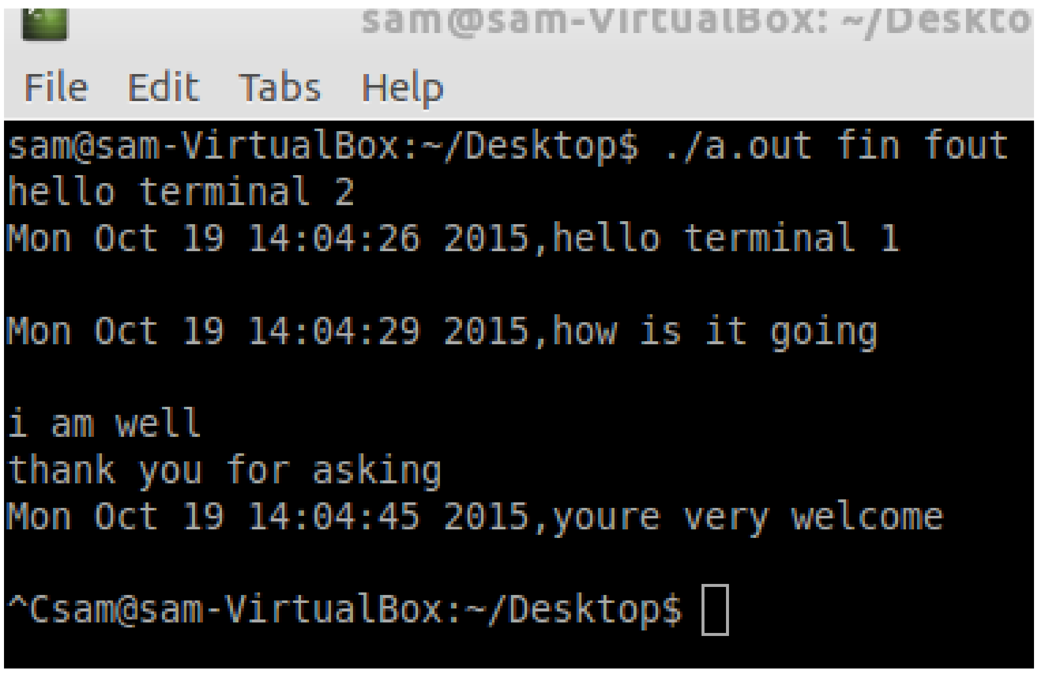
File Chat Program

This project requires that you write a C or C++ application to read from the screen and write that data with a timestamp to an output file. The application should also read from an input file and print the data read to the screen. Reads are normally blocking calls. Setting up timer generation in a certain way will allow your program to break out of those calls.

1. Your program **shall** accept the name of a file on the command line as the first argument. This is the *input file* for the application.
2. Your program **shall** accept the name of a file on the command line as the second argument. This is the *output file* for the application.
3. Your program **shall** read from standard input, accepting input data from the user.
   (a) Your program **shall** write all data received from the user to the *output file*.
   (b) Your program **shall** time stamp all output data written into the *output file*.
   (c) The format of your *output file* **shall** be `TIMESTAMP,STRING\n`. This output file should contain comma separated values with two columns: the time stamp and the input data.
4. Your program **shall** read from the input file.
   (a) Your program **shall** write all data read from the input file to the screen. If there is no data to be written, go back to reading input from the screen.
5. Your program **shall** not spend more than three (3) seconds waiting for input. If the user doesn't enter anything on the screen, switch to reading the file and vice versa.
6. Your program **shall** continually read from standard in and the input file forever. Do not exit. Keep going until you get interrupted with Ctrl-C (then you can exit normally).
7. Your program **shall** correctly close all opened file. This should happen even if you get the interrupt signal in the terminal.
8. Your program **shall** meet the basic requirements listed below:
   (a) Your program **shall** print error information for each error in so much as your program was directly responsible for the method call that produced the error.
   (b) Your program **shall** terminate after encountering an unrecoverable error (after printing error information).
   (c) You **shall** submit all source code and a makefile in a tar file.
      i. The makefile **shall** be able to build your source in a 32 bit Lubuntu VM.
      ii. The built program **shall** be able to run in a 32 bit x86 Lubuntu VM.

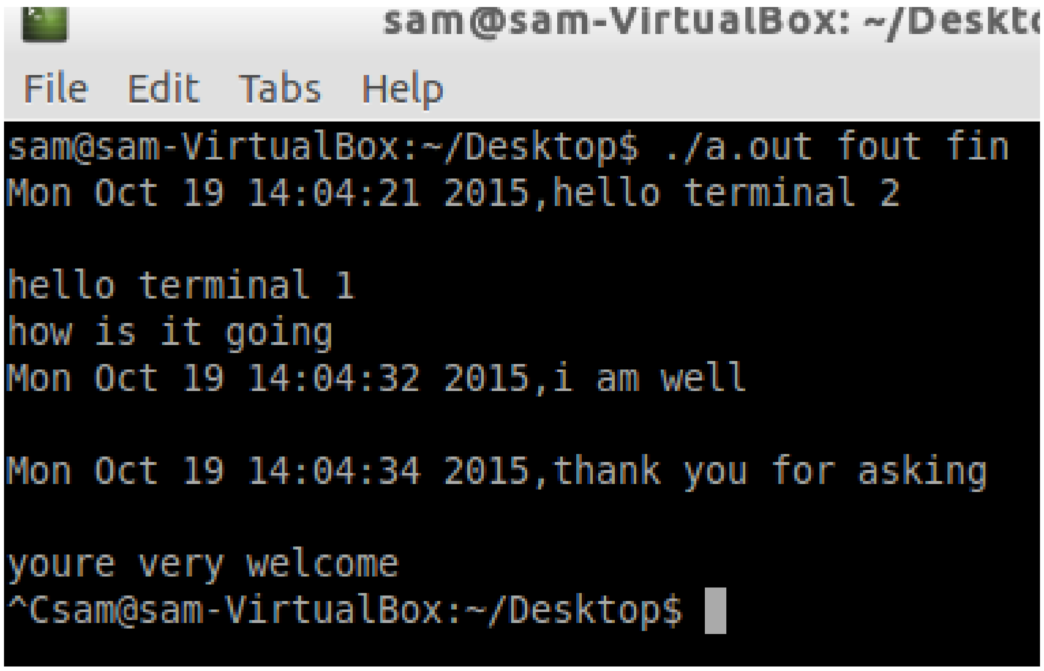


```
sam@sam-VirtualBox:~/Desktop$ ./a.out fin fout
hello terminal 2
Mon Oct 19 14:04:26 2015,hello terminal 1

Mon Oct 19 14:04:29 2015,how is it going

i am well
thank you for asking
Mon Oct 19 14:04:45 2015,youre very welcome

^Csam@sam-VirtualBox:~/Desktop$
```



```
sam@sam-VirtualBox:~/Desktop$ ./a.out fout fin
Mon Oct 19 14:04:21 2015,hello terminal 2

hello terminal 1
how is it going
Mon Oct 19 14:04:32 2015,i am well

Mon Oct 19 14:04:34 2015,thank you for asking

youre very welcome
^Csam@sam-VirtualBox:~/Desktop$
```

Figure 1: Example output for the program. Your output should pretty much match exactly (given the same input). This will probably make more sense after we see it in class.