Encrypted Messenger Application in Haskell Version 2.0 12/10/2015

INTRODUCTION

————————

- The version 2.0 of this application is a simple encrypted instant message application over a network in Haskell.

GENERAL USAGE NOTES

—————————————

- The use of a terminal with ghci is required.

- The use one terminal for the server and any other terminals for each new client is recommended.

- The server only connects on the localhost, so client instances must be executed locally.

- The .hs files can be executed in ghc by calling main (with no arguments), although the compiled version is preferred.

- Encrypted messages are just implemented for proof of concept. However, it does not decrypt. Its hardcoded at 2 for testing purposes.

- CaesarCipher.hs, MonoAlpha.hs and PolyAlpha.hs can be opened and tested for the actual encrypt/decrypt functions

- Compiled and tested on OS X El Capitan, Windows 8.1, and Ubuntu 14.0x

LIBRARIES USED

—————————

- Network.Socket

- System.IO

- Control.Exception

- Control.Concurrent

- Control.Concurrent.Chan

- Control.Monad

- Control.Monad.Fix (fix)

- Data.Char

MODULES IMPORTED

————————

- CaesarCipher.hs —> this module is a simple substitution cipher that has 25 possible shifts, but can iterate over those shifts indefinitely.

- MonoAlpha.hs —> this module is doing the encryption of the message without space. The encryption key is a number.

- PolyAlpha.hs -> this module is doing the encryption of the message with space. The encryption key is a string.

HOW TO COMPILE AND TEST WITH JUST ENCRYPTION

——————————————————————————————————

- First: compile the server file Server\_Options.hs with “ghc –threaded –make Server\_Options.hs”

- Second: compile client Client\_Options.hs with “ghc –threaded –make Client\_Options.hs”

- Third: Execute Server with ./Server\_Options, or Server\_Options.exe.

- Fourth: Client\_Options with ./Client\_Options or Client\_Options.exe

- Then you can repeat the fourth step to create any new client.

(WARNING: Has not been tested with more than two clients)

* NOTE: This version ONLY encrypts with one shift, it DOES NOT decrypt, as this is a proof of concept for the Midpoint Deliverable. This has not been thoroughly debugged.

AUTHORS

—————

Romain Brunie and Matthew Tennis.