Messager Documentation

A simple command line messaging service.

Goals:

* Peer to peer.
* User types in IP of pc on network and network name to connect to a messaging network.
* User can connect to a single network.
* User can adjust settings.

Distant goals:

* User can be connected to multiple networks.

Structure:

* /
  + Makefile, README.md and project folders.
* /bin
  + Object files (if needed) and project executable.
* /src
  + Source files (subdirectories may be needed).
* /src/headers
  + Source file headers.
* documentation
  + documentation folder (duh).
* /tests
  + Tests source files, test executable, and test makefile.
* /tests/headers
  + Test source file headers.
* /tests/testfiles
  + Where test data files are kept.

Source files (in src):

* main.c:
  + Insertion point for code.
  + Don’t put functions that need to be unit tested in here.
* fileio.c:
  + Input and output for files.
* networkio.c:
  + For sending and receiving things on the network.
* userio.c
  + For displaying and getting strings to/from the user.
* mainloop.c
  + This is where the mainloop(s) are kept (i.e. network io, and user io).
* definitions.h
  + General definitions to be used through the program, such as TRUE and FALSE.

Program navigation:

This is the policy for how the user will navigate the program.

When the user first starts the program they will be greeted with a welcome message and a command prompt the input state is set to the command state.

Command state:

* “settings”: Open settings (sets settings state).
* “create”: Create network.
* “join”: Join network (sets join network state).
* “help” or “h”: Print help.
* “quit”: Quit program.

Settings state:

* “[setting] = [val]”: Set a setting.
* “save”: Save settings.
* “reset”: Reset settings.
* “discard”: Discard changes.
* “help” or “h”: Get help.
* “return”: Return (sets command state).

Join network state:

* “[network\_name]”: Input a network name (sets password state).
* “return”: return (sets command state).

Messaging state:

* “[message]”: Input a message.
* “:help” or “h”: Print help.
* “:return”: Return (sets command state).

Input password state:

* “[password]”: Input a password (sets message state on network join or asks for password again).
* “return”: Return (sets command state).

Create network state:

* “[network\_name]”: Input a network name (sets set password state).
* “return”: Return (sets command state).

Set password state:

* “[password]”: Set a password (returns to command state on success or prompts user for another password on invalid password).
* “return”: Return (sets command state).

User commands:

User input command functions are defined void fun(char\* args[MAX\_ARGS]);  
where char\* args[MAX\_ARGS] = {command, arg1, arg2, arg3, …., argmax - 1};

User commands are stored in char\* USER\_COMMAND[NUM\_USER\_COMMANDS];  
Command functions are stored in USER\_COMMAND\_FUNCS[NUM\_USER\_COMMANDS];  
USER\_COMMANDS[n] maps to USER\_COMMAND\_FUNCS[n]

Test structure:

* Each source file will have its own testing c module which will contain one insertion point and return a 1 or 0 value (0 for fail and 1 for true).
* Insertion point function will receive a pointer to a 256 character long piece of memory to which it can add any error message, which will be displayed if the function returns 0.
* Tests are run from testsmain.c.