University of Mississippi, Department of Electrical Engineering

El E 425 — TerminalTalk

Bryan Harper

November 4, 2016

Contents

Part I

1 Introduction

1.1 Overview

Test 123

1.2 The End Product

In order to determine the application architecture and protocol with which to implement the software it is first necessary to develop a clear picture of what is being built. The following is a description of what an end user can expect when running the *TerminalTalk* application.

The application will be completely run from the command line (terminal on Linux). There will be no additional graphical user interface beyond what the terminal provides (ANSI characters, cursor animation, colors, etc.). Thus, any interaction the user has with the application will be through their keyboard, in the form of normal input or specialized commands.

Upon starting the application — with the option of specifying a moniker (username) — the user will be connected to a *chat room*. The user's arrival will be announced to all users currently connected:

```
[moniker] has entered.
```

There will also be available to the user a non-intrusive prompt for the user to deliver a message to the room:

```
Talk:
```

If the user types a message and then presses `'Enter'` on their keyboard, the user will simply see their message displayed after the prompt, and then another prompt on a new line.

```
Talk:  [message inputed by user]
Talk:
```

However, any other user in the room will see something like:

```
[moniker1]:  [message inputed by user1]
Talk:
```

Note that the first user's moniker is colored. When a user first begins the application the program randomly assigns a color to the user. However, if a user wishes to change their assigned color, they can use the command `\setcolor`:

```
Talk:  \setcolor green
Your color has been set to green.
```

There is also a command to change the user's moniker:

```
Talk:  \setmoniker Alan
Hello Alan.
```

And a command to disconnect:

```
Talk:  \disconnect
You have disconnected.
```

Seen from the perspective of other connected users, the output resulting from a user running the above commands might look like this:

```
Alan has entered.
Anonymous:  Hello everyone!
Anonymous:  How's it going?
Alan:  Uh oh, I've got to run!
Alan has exited.
```

It's expected that the chat room will always be available for a user to connect with at any time. It is also expected that messages are transferred without error. To see the necessity of this, consider if the letter 'e' were to be dropped from the word 'appeal' in a message. Clearly, such an error could lead to a detrimental misunderstanding.

In a similar vein, it is necessary that messages are received in order. If, for example, a disconnect command were received out of order and before another message, the user would be disconnected before their message got through. This is clearly an unacceptable result.

2 Theoretical Discussion

2.1 Architecture

2.2 Application Protocol

3 Implementation

Part II

API

4 Namespace Index

4.1 Packages

Here are the packages with brief descriptions (if available):

5 Class Index

5.1 Class List

Here are the classes, structs, unions and interfaces with brief descriptions:

6 File Index

6.1 File List

Here is a list of all documented files with brief descriptions:

7 Namespace Documentation

7.1 client Namespace Reference

Functions and definitions specific to TerminalTalk client.

7.1.1 Detailed Description

Functions and definitions specific to TerminalTalk client.

7.2 font Namespace Reference

A collection of functions that style fonts in terminal.

7.2.1 Detailed Description

A collection of functions that style fonts in terminal.

Each function flanks the inputed text with ANSI codes.

7.3 server Namespace Reference

Functions and definitions specific to TerminalTalk server.

7.3.1 Detailed Description

Functions and definitions specific to TerminalTalk server.

8 Class Documentation

8.1 font.constants.Colors Class Reference

Contains constants for ANSI color codes.

Static Public Attributes

- string **BLACK** = '\033[30m]'
- string **RED** = '\033[31m'
- string **GREEN** = '\033[32m'
- string **YELLOW** = '\033[33m'
- string **BLUE** = '\033[34m'
- string **MAGENTA** = '\033[35m'
- string **CYAN** = '\033[36m'
- string **WHITE** = '\033[37m'

8.2 Detailed Description

Contains constants for ANSI color codes.

Prefix a string with color constant to change subsequent text to the color. Example: print(font.Colors.↩ RED + "This text will be red.")

The documentation for this class was generated from the following file:

- /TerminalTalk/font/constants.py

8.3 server.Orator.Orator Class Reference

Stores information about a connected client.

Public Member Functions

- def __init__ (self, telegraph)
  Constructor for Orator class.

Public Attributes

- **telegraph**
- **moniker**
- **color**

8.3.1 Detailed Description

Stores information about a connected client.

An Orator objects is used to represent a client. Each client has a moniker and color associated with them. The moniker is provided by the client. The color is assigned randomly each time the client connects.

8.3.2 Constructor & Destructor Documentation

8.3.2.1 def server.Orator.Orator.__init__ (self, telegraph)

Constructor for Orator class.

Parameters

telegraph	The socket used to connect with client.

The documentation for this class was generated from the following file:

- /TerminalTalk/server/Orator.py

8.4 font.constants.Styles Class Reference

Contains constants for ANSI text style codes.

Static Public Attributes

- string **RESET** = '\033[0m'
- string **BOLD** = '\033[1m'
- string **BOLD_OFF** = '\033[22m'
- string **ITALIC** = '\033[3m'

- string **ITALIC_OFF** = '\033[23m'
- string **UNDERLINE** = '\033[4m'
- string **UNDERLINE_OFF** = '\033[24m'
- string **STRIKETHROUGH** = '\033[9m'
- string **STRIKETHROUHG_OFF** = '\033[29m'
- string **INVERSE** = '\033[7m'
- string **INVERSE_OFF** = '\033[27m'

8.4.1 Detailed Description

Contains constants for ANSI text style codes.

Prefix a string with a style constant to change subsequent text style. Example: print(font.Styles.BOLD + "This text will be bold.")

The documentation for this class was generated from the following file:

- /TerminalTalk/font/constants.py

9 File Documentation

9.1 /TerminalTalk/client/eavesdrop.py File Reference

Functions

- def client.eavesdrop.eavesdrop ()
  Monitors for user input from terminal.

9.2 /TerminalTalk/font/constants.py File Reference

Defines classes contains ANSI constants for formating text.

Classes

- class font.constants.Styles
  Contains constants for ANSI text style codes.
- class font.constants.Colors
  Contains constants for ANSI color codes.

9.2.1 Detailed Description

Defines classes contains ANSI constants for formating text.

9.3 /TerminalTalk/font/functions.py File Reference

Defines functions for surronding strings with ANSI codes for formating text.

Functions

- def font.functions.blue (txt)

  Makes inputed string blue.
- def font.functions.cyan (txt)

  Makes inputed string cyan.
- def font.functions.green (txt)

  Makes inputed string green.
- def font.functions.magenta (txt)

  Makes inputed string magenta.
- def font.functions.red (txt)

  Makes inputed string red.
- def font.functions.yellow (txt)

  Makes inputed string red.
- def font.functions.underline (txt)

  Makes inputed string underline.
- def font.functions.bold (txt)

  Makes inputed string bold.
- def font.functions.italic (txt)

  Makes inputed string italic.
- def font.functions.default (txt)

  Gives inputed string default formatting.
- def font.functions.highlight (txt)

  Highlights inputed string.

9.3.1 Detailed Description

Defines functions for surronding strings with ANSI codes for formating text.

9.3.2 Function Documentation

9.3.2.1 def font.functions.blue (txt)

Makes inputed string blue.

Returns with inputed string with the prefix '\033[34m' and suffix '\033[0m'. The prefix is the ANSI code for blue foreground. The suffix is the ANSI code for reset (white foreground, black background).

Parameters

txt	A string. Will be displayed as blue.

9.3.2.2 def font.functions.bold (txt)

Makes inputed string bold.

Returns with inputed string with the prefix '\033[1m' and suffix '\033[22m'. The prefix is the ANSI code for bold font on. The suffix is the ANSI code for bold font off.

Parameters

txt	A string. Will be made bold.

9.3.2.3 def font.functions.cyan (txt)

Makes inputed string cyan.

Returns with inputed string with the prefix '\033[36m' and suffix '\033[0m'. The prefix is the ANSI code for cyan foreground. The suffix is the ANSI code for reset (white forground, black background).

Parameters

txt	A string. Will be displayed as cyan.

9.3.2.4 def font.functions.default (txt)

Gives inputed string default formatting.

Returns with inputed string with the prefix '\033[0m'. The prefix is the ANSI code for reset (white foreground, black background).

Parameters

txt	A string. Will be formatted as default.

9.3.2.5 def font.functions.green (txt)

Makes inputed string green.

Returns with inputed string with the prefix '\033[32m' and suffix '\033[0m'. The prefix is the ANSI code for green foreground. The suffix is the ANSI code for reset (white forground, black background).

Parameters

txt	A string. Will be displayed as green.

9.3.2.6 def font.functions.highlight (txt)

Highlights inputed string.

Returns with inputed string with the prefixes '\033[47m', '\033[30m', and'\033[3m', and with the suffix '\033[0m'. The prefix is the ANSI code for italic, white background, and black foreground. The suffix is the ANSI code for reset (white forground, black background).

Parameters

txt	A string. Will be highlighted.

9.3.2.7 def font.functions.italic (txt)

Makes inputed string italic.

Returns with inputed string with the prefix '\033[3m' and suffix '\033[23m'. The prefix is the ANSI code for italic on. The suffix is the ANSI code for italic off.

Parameters

txt	A string. Will be displayed as italic.

9.3.2.8 def font.functions.magenta (txt)

Makes inputed string magenta.

Returns with inputed string with the prefix '\033[35m' and suffix '\033[0m'. The prefix is the ANSI code for magenta foreground. The suffix is the ANSI code for reset (white forground, black background).

Parameters

txt	A string. Will be displayed as magenta.

9.3.2.9 def font.functions.red (txt)

Makes inputed string red.

Returns with inputed string with the prefix '\033[31m' and suffix '\033[0m'. The prefix is the ANSI code for red foreground. The suffix is the ANSI code for reset (white forground, black background).

Parameters

txt	A string. Will be displayed as red.

9.3.2.10 def font.functions.underline (txt)

Makes inputed string underline.

Returns with inputed string with the prefix '\033[4m' and suffix '\033[24m'. The prefix is the ANSI code for underline on. The suffix is the ANSI code for underline off.

Parameters

txt	A string. Will be underlined.

9.3.2.11 def font.functions.yellow (txt)

Makes inputed string red.

Returns with inputed string with the prefix '\033[33m' and suffix '\033[0m'. The prefix is the ANSI code for red foreground. The suffix is the ANSI code for reset (white forground, black background).

Parameters

txt	A string. Will be displayed as red.

9.4 /TerminalTalk/server/Orator.py File Reference

Classes

- class server.Orator.Orator

  Stores information about a connected client.

9.5 /TerminalTalk/server/pontification.py File Reference

Functions

- def server.pontification.megaphone (verbiage, connections, ear_trumpet, orators)

  Make announcements from server to everyone connected.
- def server.pontification.pontificate (orator, verbiage, connections, ear_trumpet, orators)

  Transmit a clients message to all other connected users.

9.5.1 Function Documentation

9.5.1.1 def server.pontification.megaphone (verbiage, connections, ear_trumpet, orators)

Make announcements from server to everyone connected.

The message will be formatted (see font.highlight) so as to stick out from all other text. Every connected user will see the message.

Parameters

verbiage	The message displayed.
connections	A list of all current connections (sockets).
ear_trumpet	The server's socket. Used to ensure message isn't sent to server itself (this would break pipe).
orators	A list of Orator objects for currently connected clients. If a disconnection is discovered while function is being run, it is necessary to remove the object from the list.

9.5.1.2 def server.pontification.pontificate (orator, verbiage, connections, ear_trumpet, orators)

Transmit a clients message to all other connected users.

Parameters

orator	The sending client's Orator object.

Parameters

verbiage	The message displayed.
connections	A list of all current connections (sockets).
ear_trumpet	The server's socket. Used to ensure message isn't sent to server itself (this would break pipe).
orators	A list of Orator objects for currently connected clients. If a disconnection is discovered while function is being run, it is necessary to remove the object from the list.

9.6 /TerminalTalk/TerminalTalk.py File Reference

Main code for TerminalTalk client.

Variables

- string **TerminalTalk.moniker** = "Anonymous"
- int **TerminalTalk.buffer_size** = 2
- int **TerminalTalk.server_port** = 7777
- string **TerminalTalk.server_ip** = "192.168.1.77"
- tuple **TerminalTalk.server_address** = (server_ip, server_port)
- **TerminalTalk.megaphone** = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
- **TerminalTalk.request** = megaphone.recv(buffer_size)
- list **TerminalTalk.connections** = [megaphone, sys.stdin]
- **TerminalTalk.readables**
- **TerminalTalk.writables**
- **TerminalTalk.errors**
- **TerminalTalk.missive** = telegraph_i.recv(buffer_size)
- **TerminalTalk.verbiage** = sys.stdin.readline()

9.6.1 Detailed Description

Main code for TerminalTalk client.

Run this file to use as client.

9.7 /TerminalTalk/TerminalTalk_server.py File Reference

Main code for TerminalTalk server.

Variables

- int **TerminalTalk_server.port** = 7777
- int **TerminalTalk_server.buffer_size** = 2
- tuple **TerminalTalk_server.server_address** = ("", port)
- list **TerminalTalk_server.connections** = []
- list **TerminalTalk_server.orators** = []
- **TerminalTalk_server.ear_trumpet** = socket.socket(socket.AF_INET, socket.SOCK_STR↩ EAM)
- **TerminalTalk_server.readables**
- **TerminalTalk_server.writables**
- **TerminalTalk_server.errors**
- **TerminalTalk_server.file_descriptor**
- **TerminalTalk_server.address**
- **TerminalTalk_server.moniker** = file_descriptor.recv(buffer_size)
- string **TerminalTalk_server.entrance_message** = moniker+" has entered."
- int **TerminalTalk_server.orator_index** = 0
- **TerminalTalk_server.verbiage** = telegraph_i.recv(buffer_size)
- string **TerminalTalk_server.missive** = orators[orator_index].moniker+" has exited."

9.7.1 Detailed Description

Main code for TerminalTalk server.

Run this file to start server.