
x/84 Documentation

Release 2.0.9

Jeff Quast

April 21, 2015

1	Introduction	3
1.1	Technologies	3
1.2	Quickstart	4
1.3	Documentation, Support, Issue Tracking	4
2	Project Details	5
2.1	Compatible Clients	5
2.2	Binding to port 23	6
2.3	Other Telnet BBS Systems	6
2.4	History	8
2.5	What does x/84 mean?	8
2.6	Future Directions	8
3	Running a message server	9
3.1	Configuring a hub	9
3.2	Configuring a leaf node	10
3.3	Authorship	10
4	Doors	11
4.1	Dosemu	12
5	Web server	13
5.1	Starting a web server	13
5.2	Lookup path	13
5.3	Serving static files	14
5.4	Writing a web module	14
6	Developers	17
6.1	Requirements	17
6.2	Contributing using git	18
7	Customizing your board	19
7.1	main(), gosub, and goto	19
7.2	Basic example	19
8	Engine	21
8.1	x84.engine	21
8.2	x84.db	22

9	Protocols	25
9.1	x84.server	25
9.2	x84.client	26
9.3	x84.telnet	27
9.4	x84.ssh	30
9.5	x84.rlogin	30
9.6	x84.sftp	31
9.7	x84.webserve	31
10	Service Plugins	33
10.1	x84.fail2ban	33
10.2	x84.msgpoll	33
11	Terminal I/O	35
11.1	x84.terminal	35
12	Userland/scripting API	39
12.1	x84.bbs.door	39
12.2	x84.bbs.editor	42
12.3	x84.bbs.ini	45
12.4	x84.bbs.ipc	46
12.5	x84.bbs.lightbar	46
12.6	x84.bbs.output	48
12.7	x84.bbs.pager	50
12.8	x84.bbs.session	52
12.9	x84.bbs.userbase	56
12.10	x84.bbs.ansiwin	58
12.11	x84.bbs.dbproxy	59
12.12	x84.bbs.exception	61
12.13	x84.bbs.modem	61
12.14	x84.bbs.msgbase	61
12.15	x84.bbs.selector	62
12.16	x84.bbs.telnet	64
13	Future Directions	65
14	Indexes	67
	Python Module Index	69

Read the [pdf version](#)

Contents:

Introduction

An experimental python Telnet and SSH server framework

The primary purpose of x/84 is to provide a server framework for building environments that emulate the feeling of an era that predates the world wide web. It may be used for developing a classic bulletin board system (BBS) – one is provided as the ‘default’ scripting layer. It may also be used to develop a MUD, a text-based game, or a game-hosting server such as done by dgamelaunch.

You may access the “default board” provided by x/84 at telnet host 1984.ws:

```
telnet 1984.ws

# or
ssh anonymous@1984.ws

# or
rlogin 1984.ws
```

1.1 Technologies

x/84 supplies a [scripting](#) engine for developing character-at a time telnet or ssh server, such as **MUD** or **BBS** systems. Technologies used in x/84 are derived from [miniboa](#) (Apache 2.0 Licensed) for telnet, [blessed](#) (MIT Licensed) for terminal capabilities, [sqlitedict](#) (Public Domain) for persistent data, [paramiko](#) for ssh and sftp services, and [web.py](#) for http service.

Asynchronous inter-process communication between sessions is provided through an event queuing framework, for scripting of ‘shared’ experiences. Several examples of these are provided, such as [chat.py](#). The default board provides several demonstrating activities.

All terminal types supported by curses (the [termlib](#) and [terminfo](#)) databases are allowed, with a “pythonic” terminal framework supplied through [blessed](#).

Portability is as equal to python, and has been tested on Raspberry Pi, Android, Mac, OpenBSD, Solaris, etc.

ANSI Art, such as found on [ACiD dark domains](#) DVD, is translated for reasonably accurate reproductions for both UTF-8 and IBM CP437 terminals. This allows classic DOS art to be used on modern terminals such as Terminal.app, or classic emulating terminals such as [syncterm](#). Artwork with [Sauce](#) records are also supported.

See [clients](#) for a list of compatible clients.

1.2 Quickstart

Note that only Linux, BSD, or OSX is supported, due to the `blessed` dependency on curses.

1. Install `python 2.7` and `pip`. More than likely this is possible through your preferred distribution packaging system.

3. Install `x/84`:

```
pip install x84[with_crypto]
```

Or, if C compiler and `libssl`, etc. is not available, simply:

```
pip install x84
```

Please note however that without the `[with_crypto]` option, you will not be able to run any of the web, ssh, and sftp servers, and password hashing (and verification) will be significantly slower.

4. Launch the `x84.engine` python module:

```
x84
```

5. Telnet to 127.0.0.1 6023, Assuming a `bsd telnet` client:

```
telnet localhost 6023
```

All data files are written to `~/x84/`. To create a custom board, you might copy the default folder of the `x/84` python module to a local path, and point the `scriptpath` variable of `~/x84/default.ini` to point to that folder.

Simply edit and save changes, and re-login to see them. Adjust the `show_traceback` variable to display any errors directly to your telnet or ssh client.

1.3 Documentation, Support, Issue Tracking

See [Documentation](#) for API and general tutorials, especially the [developers](#) section for preparing a developer's environment if you wish to contribute upstream. Of note, the *Terminal* interface is used for keyboard input and screen output, and is very well-documented in `blessed`.

This project isn't terribly serious (for example, there are no tests), though contributions (especially fixes and documentation) are welcome. See the project on [github](#) for source tree and issue tracking. If there are features, bugs, or changes you would like to see, feel free to open an issue.

If you would like to chat with developers of `x/84`, we are in channel `#1984` on `irc.efnet.org`.

Project Details

General information useful for prospective developers and users.

2.1 Compatible Clients

Any UTF-8 client is compatible. For Apple systems, **Andale Mono** works wonderfully for cp437 blockart. Please note that many modern terminal emulators (especially Apple) modify the default 16 colors away from their original CGA specification. This will cause CP437 block-art to appear milky and poor, you should ensure your colorscheme is configured exactly as the CGA specification http://en.wikipedia.org/wiki/Color_Graphics_Adapter#Color_palette

- PuTTY
 - Under preference item *Window* -> *Translation*, option *Remote character set*, change *iso8859-1* to *UTF-8*.
- iTerm/iTerm2
 - Menu item *iTerm* -> *Preferences*, section *Profiles*, select tab *Text*, chose **Andale Mono** font.
- Terminal.app
 - Menu item *Terminal* -> *Preferences*, chose profile *Pro*, select Font **Andale Mono**, and enable **use bright colors for bold text**.
- uxterm
 - Or other utf-8 rxvt and xterm variants: urxvt, dtterm. Recommended font is **Deja Vu Sans Mono**.
- Amtelnet (Amiga workbench)
 - Enable the tool type *NOSCROLLER* in the Amtelnet icon file in order to disable the scrollbar and enter full screen width.
- Non-unicode Terminals
 - Other than UTF-8, only IBM CP437 encoding is supported. Any telnet client with CP437 font is (currently) supported.
 - Examples of these include **PuTTY**, **SyncTerm**, **mtel**, **netrunner**, various minix/linux/bsd consoles with a linux or bsd telnet client.
 - Some non-DOS terminal emulators may require installing a fontset, such as **Terminus** to provide CP437 art.

2.2 Binding to port 23

x/84 does not require privileged access, and its basic configuration binds to port 6023 for telnet and 6022 for ssh. Multi-user systems do not typically allow non-root users to bind to port 23 or 22. Below are various techniques for allowing it.

Alternatively, you can always use port forwarding on a NAT firewall.

2.2.1 Linux

using `privbind`, run the BBS as user ‘nobody’, group ‘nogroup’:

```
sudo privbind -u nobody -g nogroup x84
```

The default board, `1984.ws` runs from the git master branch from a virtualenv using command:

```
PYTHON_EGG_CACHE=/tmp/nobody.python-eggs sudo privbind -u nobody -g nogroup `which python` -mx84.egg
```

with system files `/etc/x84/default.ini` and `/etc/x84/logging.ini` configured to save data in nobody-owned files and folders at path `/var/x84`.

2.2.2 Solaris 10

grant `net_privaddr` privilege to user ‘bbs’:

```
usermod -K defaultpriv=basic,net_privaddr bbs
```

2.2.3 BSD

redirection using `pf(4)`:

```
pass in on egress inet from any to any port telnet rdr-to 192.168.1.11 port 6023
```

2.2.4 Other

Using `socat`, listen on 192.168.1.11 and for each connection, fork as ‘nobody’, and pipe the connection to 127.0.0.1 port 6023:

```
sudo socat -d -d -llocal12 TCP4-LISTEN:23,bind=192.168.1.11,su=nobody,fork,reuseaddr TCP4:127.0.0.1:
```

This has the disadvantage that x84 is unable to identify the originating IP.

2.3 Other Telnet BBS Systems

Listed here is software known in the “bbs-scene” as still being actively used, in descending order of their (estimated) popularity.

- `synchronet`: C formerly commercial, now open source.
- `mystic`: Pascal, create a sourceforge account to access source code.
- `daydream`: C open source.

- [enthral](#): C++ open source.

Many more systems can be found at [List_of_BBS_software](#)

2.3.1 How x/84 compares

It might best to compare x/84 with the most popularly used surviving BBS systems, mainly: [mystic](#), [synchronet](#), and [daydream](#).

Process Management

- All other systems are single process: executed as a “login shell” by xinet.d or similar, they depend on additional 3rd-party systems and distribution packages for telnet or ssh support.
- x/84 on the other hand, is a single process that manages the telnet, ssh, sftp, web, and rlogin server. This means no additional steps are required to start a working bbs once installed; no special user accounts, xinet.d, or database setup required, only python.
- This tight integration allows one to login by ssh or sftp with your bbs user account and public key, for example. Or to react to and determine window-size changes over telnet and ssh.
- as a dynamic language, it also allows one to rapidly develop on much of the system without compilation or publishing layer – simply login again to see the new changes afresh without restarting the server, and without a compilation step.
- a “script stack” allows exceptions in scripts to be managed and optionally displayed to the client. One can rapidly develop a script from the main menu, try it, see an exception such as a `SyntaxError` thrown, with the traceback and offending line. Then, fix and save changes from your editor, and select the menu option to try it again – without ever logging off!

Scripting Layer

- All other systems are written in C or Pascal, published in binary form, providing a limited subset of functionality through a scripting layer in an entirely different language, such as a particular dialect of javascript, python, perl, or pascal.
- x/84 is python throughout – you may extend the engine layer to provide new features in the same language and with full access in the scripting layer without providing any stubs, function exports, or facilitating modules. The same methods used in the engine for session and user management are available in the scripting layer.

Customization

- Most systems take an approach of providing a proprietary layer of customization: special menu files with codes for navigating between other menus and scripts, or displaying artfiles with special codes for displaying dynamic data such as a login name.
- x/84 customization is done only by python scripting. Making a menu is simply writing a script to do so. One may simply echo out the contents of an artfile, move the cursor to the desired location, and echo out any variable. Special functions are provided to gain access to, for example, “Terminal” and “Session”, but do not necessarily require it. There are no limitations, you may use anything python is capable of.

Encoding

- All other systems are completely agnostic of encoding – so most systems assume an IBM-PC CP437 encoding, or must specify which “character set” to use. This means a bbs must either conform to english-only, or require connecting clients to chose a specific character set for their terminal emulator, which means compromising to ascii-only art.
- x/84 primarily supports only UTF-8, with special accommodation for CP437-only terminal encodings, such as SyncTerm. This allows the same BBS containing CP437-encoded artwork and DOS-emulated Doors (such as

Lord) to be presented on modern terminals, yet host any number of UTF-8 supported languages such as japanese, swedish, russian, etc.

2.4 History

In 2002, [Jeff Quast](#), author of x84 ran [mystic](#) on Linux which gained popularity due to its association with a pirate channel he managed on efnet, regularly receiving 30-50 daily callers, which exposed numerous bugs and design issues. Frustrated by its closed-source nature and the (intermittent) abandonment of the author, Jeff set out to write his own from-scratch.

He and [Johannes Lundberg](#) of Sweden met who had already began writing his own system, initially named just “pybbs”, this was authored in the Python language. Overnight, a 5,000-line patch was returned to Johannes and they agreed to collaborate on a new system, with focus on the new Unix developer traditions and open source.

They grew apart over time with their forks, Johannes providing a new redesign called “The Progressive (PRSV)”, which Jeff re-based and began to contribute to when they re-combined efforts years later. Johannes continually asserted that he would maintain and later release PRSV, but as his involvement waned, Jeff renamed his fork as x/84, with the intent to merge upstream some day.

x/84 retains only some of the design and basic variables, such as the concept of a session but is otherwise completely rewritten by the work of Jeff alone through 2013, when many contributions over github were received after being released to pypi.

2.5 What does x/84 mean?

x/84 is a re-imagination of the early dial-up systems. Targeted for, but not limited to, running a bulletin board over the TCP/IP protocol. The name x/84 is derived from the theme of an “amiexpress-style system for an Orwellian future”.

It was thought of as a small part of a science fiction universe: an alternative future where governments have banned internet anonymity and free speech, and those who wish to have it must gateway to underground systems such as these to communicate.

It was a lot farther on the “science fiction” end of the spectrum 10 years ago...

2.6 Future Directions

basic v3.0 roadmap:

- python3 using async i/o
- windows support, requires ansi.sys support emulation for PDCurses in blessed
- ftp, ftps, fxp support
- modeling (using ‘schematics’ project) for userbase, messagebase, etc.
- support for agoranet, zeronet, etc. messaging networks

Feel free to contribute ideas as a github issue.

Running a message server

Through the web modules system, x/84 provides a clever ability of intra-bbs messaging through a json-formatted RESTful API.

This is an experimental feature recently added to v2.0, herein describes the process for beginning a message server, “hub”, and a polling and publishing message client, “leaf”. Both the hub and leaf nodes are x/84 systems: the hub, running a https server and the client polling for messages and publishing through the REST api of the hub.

3.1 Configuring a hub

Firstly, an SSL certificate and matching dnsname of the hub is *required*. The following sections assume files, given the domain 1984.ws, and were created by using `sslmate`:

```
-rw-r--r-- 1 root root 5982 Jan 01 00:00 /etc/ssl/www.1984.ws.chained.crt
-rw-r--r-- 1 root root 1879 Jan 01 00:00 /etc/ssl/www.1984.ws.crt
-rw----- 1 root root 1679 Jan 01 00:00 /etc/ssl/www.1984.ws.key
```

Then, the `default.ini` file is modified to be extended with the following details:

```
[web]
enabled = yes
addr = 88.80.6.213
port = 8443
key = /etc/ssl/www.1984.ws.key
cert = /etc/ssl/www.1984.ws.crt
chain = /etc/ssl/www.1984.ws.chained.crt
modules = msgserve
```

```
[msg]
server_tags = defnet
```

The `addr` and `port` of section `[web]` keys define the TCP/IP address and port binded by the web server, and `modules` defines a list of scripts from folder `x84/webmodules` served – here, we define `msgserve`. As our serving host has multiple external IP addresses, we choose only the IP address matching our dnsname **1984.ws**. The `key`, `chain`, and `cert` are references to the SSL certificate files retrieved when running the `sslmate` purchase utility.

Messaging on x/84 implements the concept of “tags” – the most common of them are `tags public` and `tags private` – though any arbitrary tag may be applied. The `server_tags` value of section `[msg]` defines a single “tag”, that, for all messages with such tag, are served externally to the leaf nodes that poll for new messages. Here, we chose `defnet` – to signify the “default x/84 messaging network”.

When restarting x/84, we may see the log info message:

```
INFO webserve.py:223 https listening on 88.80.6.213:8443/tcp
```

3.2 Configuring a leaf node

On the hub system as a user of the ‘sysop’ group, enter the ‘sysop’ menu from the main menu, and choose ‘add new leaf node.’

It’s output will be the recommended configuration for the leaf node’s `default.ini`. You may need to adjust the `base_url` value to reflect your external dnsname (the local bind address is used, by default):

```
[msgnet_defnet]
url_base = https://88.80.6.213:8443/
board_id = 1
token = 6MvmGtvMfDF9mkuCfyGxU2IBMmFPhP8ZC70oI0hwKBk=
poll_interval = 300

[msg]
network_tags = defnet
```

Then, provide the sysop of the client bbs this output, and suggest to augment their `default.ini` with its contents and restart the leaf node.

3.3 Authorship

This extension to x/84 was authored by [@haliphax](#), who also hosted the first *hub* server on host `oddnetwork.org`.

Doors

Of the default board, a “sesame.py” script is provided (`x84/default/sesame.py`) along with dynamic addition of doors by the main menu (`x84/default/main.py`) for any scripts defined by a special notation of the `default.ini` configuration file.

A very simple unix door of `/bin/bash`, which is accessible only for users that are a member of the ‘sysop’ group is as follows:

```
[sesame]
bash = /bin/bash
bash_key = bash
bash_text = bash shell
bash_sysop_only = yes
bash_env_PATH = /bin:/usr/bin:/usr/local/bin
```

Description of sesame configuration options:

- `{name}`: The ‘basename’ name of the door file, with the value of executable and arguments used. It is only included in the main menu of the command exists, and may be disabled by using value of `no`. The command path may access information from the bbs session instance, such as `{session['handle']}`, or system-wide configuration such as `{system['datapath']}`. The special format argument `{node}` is also supplied. If it exists, a unique per-door and per-session node is acquired through the bbs global lock system.
- `{name}_env_{ENVKEY}`: Override any environment variables by `{ENVKEY}` and value.
- `{name}_key`: Command key in the main menu used to launch this door.
- `{name}_text`: Text displayed for main menu option.
- `{name}_droptype`: Any of `DOORSYS`, `DOOR32`, `CALLINFOBBS`, or `DORINFO`. This value is only honored if the command path is targets a binary named `dosemu`. The default value is `DOORSYS` if unspecified.
- `{name}_droppath`: The linux-local folder where the dropfile is saved. The dropfile will only be saved when this parameter is set..
- `{name}_nodes`: The number of nodes this door supports.
- `{name}_cols` and `{name}_rows`: Suggest the user to resize their terminal to this window size.
- `{name}_cp437` (bool): whether or not to decode the program’s output as cp437.
- `{name}_sysop_only` (bool): whether this door is limited to only sysops.

4.1 Dosemu

Doors using `dosemu` are very popular (note: only works on linux). We can configure a popular game of LORD as follows. For file `/etc/dosemu.conf`, we use configuration options:

```
$_cpu = "80486"  
$_cpu_emu = "vm86"  
$_external_char_set = "utf8"  
$_internal_char_set = "cp437"  
$_term_updfreq = (8)  
$_layout = "us"  
$_rawkeyboard = (0)
```

Of note, we use the `vm86` cpu emulator to allow real-mode emulation on virtual machines, and we use `utf8` for the external character and `cp437` for the internal character set, to allow `dosemu` to perform the codepage translations on our behalf.

We create an `X:` drive folder at `/DOS/X` containing an installation of LORD at `X:\LORD`, configured for **DORINFO** dropfiles (by running **LORDCFG.EXE**), and add the program `bnu_` to “drive C” `/DOS/.dosemu/drive_c` with `autoexec.bat` contents:

```
@echo off  
path d:\bin;d:\gnu;d:\dosemu  
set TEMP=c:\tmp  
prompt $P$G  
C:\BNU\BNU.COM /L0:57600,8N1 /F  
lredir.com x: linux\fs\DOS\X  
unix -e
```

The `unix -e` option allows passing subsequent commands by command line parameter, which is what we’ll use to offer any number of doors with the same `autoexec.bat` file. We also make sure to modify `lord’s START.BAT` to ensure the folder is changed to `X:\LORD` before starting.

Finally, we add `lord` to the `sesame` configuration:

```
[sesame]  
lord = /usr/bin/dosemu -quiet -f /etc/dosemu/dosemu.conf -I '$_com1 = "virtual"' 'X:\LORD\START.BAT  
lord_env_HOME = /DOS  
lord_key = lord  
lord_text = play lord  
lord_droptype = DORINFO  
lord_droppath = /DOS/X/lord  
lord_nodes = 32  
lord_cols = 80  
lord_rows = 25
```

Which then allows us to run this game by typing “lord” in the main menu.

Please note, that there is a 4 second pause before any input is accepted, (so you may not immediately press return at the `<MORE>` prompt). This is to work around a `dosemu` bug where input becomes garbaged and bit-shifted if any keyboard input is received during startup.

Web server

An optional web server is provided in x/84 using the basic `web.py` python library. It is possible to build web “end-points” that may make use of x/84’s database and configuration items, these are called “web modules”. Of the default board, intra-bbs messaging is provided by a web module, for example.

5.1 Starting a web server

Of your `~/x84/default.ini` file, set the configuration of the `[web]` section value `enabled = yes` (by default, it is `no`). You will also require a certificate, key, and sometimes a chain certificate file – **only** HTTPS is supported at this time. This is documented in more detail in the “Configuring a hub” section of the message network page.

For the server to successfully launch, at least one module must be enabled, the simple example modules `oneliners`, `lastcallers` may be enabled, for example:

```
[web]
enabled = yes
addr = 123.123.123.123
port = 8443
key = /etc/ssl/www.1984.ws.key
cert = /etc/ssl/www.1984.ws.crt
chain = /etc/ssl/www.1984.ws.chained.crt
modules = oneliners, lastcallers
```

If everything is configured properly, you should see something like this at startup:

```
Mon-01-01 12:00AM INFO      webserve.py:207 https listening on 123.123.123.123:8443/tcp
```

5.2 Lookup path

There are only two lookup paths for the values defined by modules, preferably, the sub-folder, `webmodules/` of your `scriptpath` configuration of section `[system]` in your `~/x84/default.ini` file. These are imported by their python module name, so file `scriptpath/webmodules/oneliners.py` is simply `oneliners`. If the file is not found there, it will then look for it in the package path of x84, which can be found using command:

```
$ python -c 'import os, x84.webmodules; print(os.path.dirname(x84.webmodules.__file__))'
```

5.3 Serving static files

One of x/84’s internal web modules is called `static`. If you enable this module, x/84 will serve static file content from the `www-static` subdirectory of your system’s top-level `scriptpath`. The top-level refers to the first item in this array. If you wish to set the document root to some other location, use the `document_root` option in the `[web]` section of your configuration file.

```
[web]
; other configuration here
modules = static
document_root = /var/www
```

The static files are served from `/www-static/`, so if your server is `https://123.123.123.123:8443`, and the file is `style.css`, it would be served as `https://123.123.123.123:8443/www-static/style.css`.

5.4 Writing a web module

While some web modules, such as the message network module, operate outside of userland and are leveraged by the engine for low-level functionality. However, you can write your own modules—and even override the internal modules—by placing your scripts in the `webmodules` subdirectory of your x/84 system’s script directory and adding them to the `modules` list in the `[web]` section of your configuration file.

As examples, two web modules have been included with the “default board” installed alongside x/84: **`:module:'x84.default.oneliners'`** and **`:module:'x84.default.lastcallers'`**. These are rudimentary examples which both read information from `DBProxy` objects and format them for display on the web. They serve to demonstrate interacting with the engine layer outside of a terminal session; accepting command options through the use of GET parameters; how Python classes ultimately translate into URL handlers; and exposing URL handlers to the x/84 engine process.

5.4.1 The handler class

First and foremost, we need to build a class which will be handling our HTTP requests. x/84’s web server uses `web.py` internally, and so we give our class a method function for each `HTTP verb` we want it to respond to. For the purposes of demonstration, the class below will only be responding to GET requests.

```
class EchoHandler(object):

    """ Demonstration URL Handler """

    def GET(self, echo=None):
        """ Echo back to the user. """

        if not echo:
            echo = u"I can't hear you!"

        return echo
```

This class will echo back whatever the user writes in the URL. If the user doesn’t write anything, it will display, “I can’t hear you!”

5.4.2 The REST API

Now, we need to inform the x/84 engine process about the existence of our web module and what URL pattern(s) it should be invoked for. We do this by putting a root-level `web_module` function in our script that returns a `dict` object with this information.

```
def web_module():
    """ Return a dict of our REST API. """

    return {'urls': ('/echo(.*)?', 'echo'), 'funcs': {'echo': EchoHandler}}
```

The first `dict` entry, `urls`, is a list where pairs of URL patterns and keywords are associated with one another. The pattern is that each even-numbered entry (0, 2, 4, 6, ...) is a URL pattern and each following odd-numbered entry (1, 3, 5, 7, ...) is a keyword for which URL handler should be invoked for this URL pattern.

The next `dict` entry, `funcs`, is a `dict` that translates those keywords into the class of the web module. In our example, we are translating the keyword, `echo`, into the class, `EchoHandler`.

5.4.3 Enabling the module

Now that we've finished with the code, we need to add our new module to the `modules` option in the `[web]` section of our configuration file. If we saved our script as `echo.py` in the `webmodules` subdirectory of our x/84 system's script path, we would use the name `echo` to refer to it in the configuration file:

```
[web]
; other configuration here
modules = echo
```

Next, we will have to restart x/84 in order for the module to be loaded.

5.4.4 Testing the module

Now, if we visit `https://123.123.123.123:8443/echo/test` in our web browser, we will see:

```
test
```

And if we visit `https://123.123.123.123:8443/echo` in our web browser, we will see:

```
I can't hear you!
```

5.4.5 Take it further

This is a very simple example. For a bit more advanced functionality, look at the source of the `:module:'x84.default.webmodules.oneliners'` and `:module:'x84.default.webmodules.lastcallers'` modules. To take it a step further still, consider looking at the `:module:'x84.webmodules.msgserve'` module in the x/84 server code.

Developers

The x/84 telnet system is written in the [Python](#) programming language. With prior programming experience you should be able to pick up the language quickly by looking at the provided sample mods in the `x84/default` folder. If you are completely new to [Python](#), it's recommended to read more about the language, like provided by the free [Dive Into Python](#) book by Mark Pilgrim.

6.1 Requirements

The following is step-by-step instructions for creating a developer environment for making your own customizations of x/84's engine and api and building your own 'scriptpath' (defined by `~/.x84/default.ini`). You may also simply install x/84 using [pip](#).

Debian, Ubuntu, Mint

You should install the following packages:

```
$ sudo apt-get install build-essential git libffi-dev libssl-dev python-dev python-setuptools python-
```

And please make sure you're using an up-to-date version of pip:

```
$ sudo pip-2.7 --upgrade pip
```

Arch Linux

You should install the following packages:

```
$ sudo pacman -S gcc git libffi python2 python2-pip python2-virtualenv python-virtualenvwrapper python-
```

And please make sure you're using an up-to-date version of pip:

```
$ sudo pip-2.7 --upgrade pip
```

6.1.1 Virtualenv

Optional but recommended, using [virtualenv](#) and/or [virtualenvwrapper](#) ensures you can install x/84 and its dependencies without root access and quickly activate the environment at any time, but without affecting system libraries or other python projects.

1. Load [virtualenvwrapper](#):

```
. `which virtualenvwrapper.sh`
```

There are techniques to automatically load `virtualenvwrapper` from your shell profile, or to active a `virtualenv` when you change to a project folder. See [virtualenv tips and tricks](#) if you're interested.

2. Make a `virtualenv` (named 'x84') using python version 2.7:

```
mkvirtualenv -p `which python2.7` x84
```

Anytime you want to load x/84 environment in a new login shell, source `virtualenvwrapper.sh` (as in step #2) and activate using command:

```
workon x84
```

6.1.2 Install editable version

Instead of installing x84 as a complete package, we use `pip` to install an *editable* version – this is so that when a modification is done to the files in our local project directory, they are immediately reflected in the x84 server anytime the `virtualenv` is activated:

```
pip install --editable .[with_crypto]
```

6.1.3 Starting x/84

```
x84
```

6.1.4 As another user

When installing x84 as an *editable* version inside a `virtualenv`, some care must be taken in regards to using `sudo` and `privbind`. This is the method used by the default board:

```
PYTHON_EGG_CACHE=/tmp/.python-eggs sudo privbind -u nobody -g nogroup `which python` -mx84.engine
```

The `\`which python\`` ensures the `virtualenv`-activated python version of the current user is used, and instead of running the `x84` script which would be not be found in the system path of target user *nobody*, we instead load the `x84.engine` module directly.

6.1.5 x84 Usage

Optional command line arguments,

```
--config= alternate bbs configuration filepath  
--logger= alternate logging configuration filepath
```

By default these are, in order of preference: `/etc/x84/default.ini` and `/etc/x84/logging.ini`, or `~/x84/default.ini` and `~/x84/logging.ini`.

6.2 Contributing using git

If you intend to contribute patches or new mods to the x/84 telnet system, you should [fork the repository](#) and clone over ssh.

Features should be developed into a branch, pushed to github, and when satisfied with your changes and you wish to have them included in the base distribution, you should [create a pull request](#).

Customizing your board

The `default.ini` file option, `scriptpath`, of section `[system]`, defines folder `'default/'`, containing the scripts documented in this section. `scriptpath` accepts a comma delimited list of directories in which to store your customizations. Noting that the left most entry is of the highest preference.

For example.

```
scriptpath = /opt/bbs/scripts,/usr/local/src/x84/x84/default
```

x84 searches for scripts in `/opt/bbs/scripts` first and then `/usr/local/src/x84/x84/default`. This allows you to keep any customizations outside of the main source tree and then fall back to x84 defaults if they're not present in your customizations directory.

Additional scripts can be found at <https://github.com/x84-extras>

This folder may be changed to a folder of your own choosing, and populated with your own scripts. A good start would be to copy the `default/` folder, or even perform a checkout from github.

By default, `matrix.py` is called on-connect, with variations for sftp and ssh as `matrix_sftp.py` and `matrix_ssh.py` set by the `default.ini` file option `script` of section `[matrix]`. This script calls out to `nua.py` for new account creation, `top.py` when authenticated, and `main.py` for a main menu.

7.1 main(), gosub, and goto

All scripts to be called by `goto` or `gosub` must supply a `main` function. Keyword and positional arguments are allowed.

If a script fails due to import or runtime error, the exception is caught, (optionally displayed by `default.ini` option `show_traceback`), and the previous script is re-started.

If a script returns, and was called by `gosub`, the return value is returned by `gosub`.

If a script returns, and was called by `goto`, the session ends and the client is disconnected.

7.2 Basic example

Let's start with a bare minimum mod, that just shows a *hello world*-style welcome to the user:

```
def main():
    from x84.bbs import echo, getterminal
    term = getterminal()
    echo(term.bold_red(u'Hello, scene!\r\n'))
```

```
echo(u'Press a key to continue...')
term.inkey()
```

So what happens here?

7.2.1 `def main()` :

This is the main entry point for your mod, as called by the previous `gosub` or `goto` call. If you supply additional arguments to either of the two, they will be passed as-is to the function invocation. We have no arguments in this example.

7.2.2 `from x84.bbs import ...`

x/84 encourages to do runtime imports, so you can change most parts of the system at runtime, without having the need to restart the whole system. Also, some of the logic is available to the local thread only, and should not leak into the global Python scope.

7.2.3 `echo(...)`

As you may have guessed, the `echo` function prints text on the user's terminal. Notice that we use *unicode* strings here. The BBS engine knows a lot about the user's terminal capabilities, including its encoding. So offering everything encoded as unicode, the engine can translate to the correct encoding for each client.

7.2.4 `term.bold_red(...)`

We use `blessed` to display the given text in `bold_red` using whichever special terminal attributes are defined by the clients `TERM` setting.

7.2.5 `term.inkey()`

Retrieves a single keystroke from the user's terminal. If the key stroke was a normal alphanumeric key, you will receive a single character that was typed as unicode, otherwise you'll get the full multibyte string, such as `\x1b[A` for the up arrow – a `code` attribute is available that can be compared with complimentary attributes of the `term` instance. See `blessed` for details.

8.1 x84.engine

Command-line launcher and event loop for x/84.

`x84.engine.accept` (*log, server, check_ban*)

Accept new connection from server, spawning an unmanaged thread.

Connecting socket accepted is `server.server_socket`, instantiate a new instance of `client_factory`, with optional keyword arguments defined by `server.client_factory_kwargs`, registering it with dictionary `server.clients`, and spawning an unmanaged thread using `connect_factory`, with optional keyword arguments `server.connect_factory_kwargs`.

`x84.engine.client_recv` (*servers, ready_fds, log*)

Test all clients for `recv_ready()`.

If any data is available, then `client.socket_recv()` is called, buffering the data for the session which is exhausted by `session_send()`.

`x84.engine.client_send` (*terminals, log*)

Test all clients for `send_ready()`.

If any data is available, then `tty.client.send()` is called. This is data sent from the session to the tcp client.

`x84.engine.find_server` (*servers, fd*)

Find matching `server.server_socket` for given file descriptor.

`x84.engine.get_servers` (*CFG*)

Instantiate and return enabled servers by configuration *CFG*.

`x84.engine.get_session_output_fds` (*servers*)

Return file descriptors of all `tty.master_read` pipes.

`x84.engine.handle_lock` (*locks, tty, event, data, tap_events, log*)

handle locking event of (*lock-key, (method, stale)*).

`x84.engine.main` ()

x84 main entry point. The system begins and ends here.

Command line arguments to `engine.py`:

- `--config=` location of alternate configuration file
- `--logger=` location of alternate logging.ini file

`x84.engine.session_recv` (*locks, terminals, log, tap_events*)

Receive data waiting for terminal sessions.

All data received from subprocess is handled here.

`x84.engine.session_send` (*terminals*)

Test all tty clients for `input_ready()`.

Meaning, tcp data has been buffered to be received by the tty session, and send it to the tty input queue (`tty.master_write`). Also, test all sessions for idle timeout, signaling exit to subprocess when reached.

8.2 x84.db

Database request handler for x/84.

class `x84.db.DBHandler` (*queue, event, data*)

Bases: `threading.Thread`

This handler receives and handles a dictionary-based “database command”.

See complimenting `x84.bbs.dbproxy.DBProxy`, which behaves as a dictionary and “packs” command iterables through an IPC event queue which is then dispatched by the engine.

The return values are sent to the session queue with equal ‘event’ name.

Class initializer.

Parameters

- **queue** (*multiprocessing.Pipe*) – parent input end of a tty session ipc queue (`tty.master_write`).
- **event** (*str*) – database schema in form of string ‘db-schema’ or ‘db=schema’. When ‘-’ is used, the result is returned as a single transfer. When ‘=’, an iterable is yielded and the data is transferred via the IPC Queue as a stream.
- **data** (*tuple*) – a dict method proxy command sequence in form of (*table, command, arguments*). For example, “(‘unnamed’, ‘pop’, 0).

run ()

Execute database command and return results to session queue.

`x84.db.check_db` (*filepath*)

Verify permission access of given database file.

Raises

- **AssertionError** – file or folder is not writable.
- **OSError** – could not write containing folder.

`x84.db.get_database` (*filepath, table*)

Return `sqlitedict.SqliteDict` instance for given database.

`x84.db.get_db_filepath` (*schema*)

Return filesystem path of given database schema.

`x84.db.get_db_func` (*dictdb, cmd*)

Return callable function of method on `dictdb`.

Raises AssertionError not a valid method or not callable.

x84.db.**get_db_lock** (*schema, table*)

Return database lock for given (*schema, table*).

x84.db.**log_db_cmd** (*log, schema, cmd, args*)

Log database command (when `tap_db` ini option is used).

x84.db.**parse_dbevent** (*event*)

Parse a database event into (*iterable, schema*).

Called by class initializer, to determine if the event should return an iterable, and for what database name (*schema*).

Return type tuple

9.1 x84.server

Package provides base server for x/84.

class `x84.server.BaseServer`

Bases: `object`

Base class for server implementations.

LISTEN_BACKLOG = 5

Number of clients that can wait to be accepted

MAX_CONNECTIONS = 100

Maximum number of clients

client_count ()

Return number of active connections.

client_factory = None

Client factory should be a class defining what should be instantiated for the client instance.

classmethod **client_factory_kwargs** (*instance*)

Return keyword arguments for the `client_factory`.

Method should be derived and modified, A dictionary may be substituted. The default return value is an empty dictionary.

:rtype dict

client_fds ()

Return list of client file descriptors.

client_list ()

Return list of connected clients.

clients = {}

Dictionary of active clients, (file descriptor, Client, ...)

clients_ready (*ready_fds=None*)

Return list of clients with data ready to be receive.

Parameters **ready_fds** (*list*) – file descriptors already known to be ready

connect_factory = None

Connect factory should be a class, derived from `threading.Thread`, that should be instantiated on-connect to perform negotiation and launch the bbs session upon success.

classmethod connect_factory_kwargs (*instance*)

Return keyword arguments for the connect_factory.

Method should be derived and modified, A dictionary may be substituted. The default return value is an empty dictionary.

:rtype dict

env = {}

Dictionary of environment variables received by negotiation

threads = []

List of on-connect negotiating threads.

9.2 x84.client

Base classes for clients and connections of x/84.

class x84.client.**BaseClient** (*sock, address_pair, on_naws=None*)

Bases: `object`

Base class for remote client implementations.

Instantiated by the corresponding `BaseServer` class.

Class initializer.

BLOCKSIZE_RECV = 64

maximum unit of data received for each call to `socket_recv()`

TTYPE_UNDETECTED = 'unknown'

terminal type identifier when not yet negotiated

addrport

IP address and port of connection as string (ip:port).

close ()

Close connection with the client.

deactivate ()

Flag client for disconnection by engine loop.

duration ()

Time elapsed since connection was made.

fileno ()

File descriptor number of socket.

get_input ()

Receive input from client into `self.recv_buffer`.

Should be called conditionally when `input_ready()` returns True.

idle ()

Time elapsed since data was last received.

input_ready ()

Whether any data is buffered for reading.

is_active ()

Whether this connection is active (bool).

kind = None

Override in subclass: a general string identifier for the connecting protocol (for example, 'telnet', 'ssh', 'rlogin')

recv_ready ()

Subclass and implement: whether `socket_recv()` should be called.

:raises `NotImplementedError`

send ()

Send any data buffered and return number of bytes send.

Raises `Disconnected` client has disconnected (cannot write to socket).

send_ready ()

Whether any data is buffered for delivery.

send_str (bstr)

Buffer bytestring for client.

send_unicode (ucs, encoding='utf8')

Buffer unicode string, encoded for client as 'encoding'.

shutdown ()

Shutdown and close socket.

Called by event loop after client is marked by `deactivate ()`.

socket_recv ()

Receive data from socket, returns number of bytes received.

Raises `Disconnect` client has disconnected.

Return type int

class `x84.client.BaseConnect (client)`

Bases: `threading.Thread`

Base class for client connect factories.

Class initializer.

banner ()

Write data on-connect, callback from `run ()`.

run ()

Negotiate a connecting session.

In the case of telnet and ssh, for example, negotiates and inquires about terminal type, telnet options, window size, and tcp socket options before spawning a new session.

stopped = False

whether this thread is completed. Set to `True` to cause an on-connect thread to forcefully exit.

9.3 x84.telnet

Telnet server for x84.

Limitations:

- No linemode support, character-at-a-time only.
- No out-of-band / data mark (DM) / sync supported

- No flow control (^S, ^Q)

This is a modified version of miniboa retrieved from svn address <http://miniboa.googlecode.com/svn/trunk/miniboa> which is meant for MUD's. This server would not be safe for most (linemode) MUD clients.

Changes from miniboa:

- character-at-a-time input instead of linemode
- encoding option on send
- strict rejection of linemode
- terminal type detection
- environment variable support
- GA and SGA
- utf-8 safe

class `x84.telnet.ConnectTelnet` (*client*)

Bases: `x84.client.BaseConnect`

Accept new Telnet Connection and negotiate options.

Class initializer.

TIME_NEGOTIATE = 2.5

maximum time elapsed allowed to begin on-connect negotiation

TIME_POLL = 0.1

polling duration during negotiation

TIME_WAIT_STAGE = 3.5

wait upto 3500ms for all stages of negotiation to complete

banner ()

This method is called after the connection is initiated.

This routine happens to communicate with a wide variety of network scanners when listening on the default port on a public IP address.

run ()

Negotiate and inquire about terminal type, telnet options, window size, and tcp socket options before spawning a new session.

class `x84.telnet.TelnetClient` (*sock, address_pair, on_naws=None*)

Bases: `x84.client.BaseClient`

Represents a remote Telnet Client, instantiated from TelnetServer.

SB_MAXLEN = 65534

maximum size of telnet subnegotiation string, allowing for a fairly large value for NEW_ENVIRON.

check_local_option (*option*)

Test the status of local negotiated Telnet options.

check_remote_option (*option*)

Test the status of remote negotiated Telnet options.

recv_ready ()

Returns True if data is awaiting on the telnet socket.

request_do_binary ()

Tell the DE that we would like them to input binary 8-bit (utf8).

request_do_env ()
Request to Negotiate About Window Size. See RFC 1073.

request_do_naws ()
Request to Negotiate About Window Size. See RFC 1073.

request_do_sga ()
Request to Negotiate SGA. See ...

request_do_ttype ()
Begins TERMINAL-TYPE negotiation

request_env ()
Request sub-negotiation NEW_ENVIRON. See RFC 1572.

request_ttype ()
Sends IAC SB TTYPE SEND IAC SE

request_will_binary ()
Tell the DE that we would like to use binary 8-bit (utf8).

request_will_echo ()
Tell the DE that we would like to echo their text. See RFC 857.

request_will_sga ()
Request DE to Suppress Go-Ahead. See RFC 858.

send_unicode (*ucs*, *encoding*='utf8')
Buffer unicode string, encoded for client as 'encoding'.

socket_recv ()
Called by TelnetServer.poll() when recv data is ready. Read any data on socket, processing telnet commands, and buffering all other bytestrings to self.recv_buffer. If data is not received, or the connection is closed, x84.bbs.exception.Disconnected is raised.

class x84.telnet.TelnetOption

Bases: `object`

Simple class used to track the status of an extended Telnet option.

Attributes and their state values:

- `local_option`: UNKNOWN (default), True, or False.
- `remote_option`: UNKNOWN (default), True, or False.
- `reply_pending`: True or False.

Set attribute defaults on init.

class x84.telnet.TelnetServer (*config*)

Bases: `x84.server.BaseServer`

Poll sockets for new connections and sending/receiving data from clients.

Create a new Telnet Server.

Parameters *config* (`ConfigParser.ConfigParser`) – configuration section [telnet], with options 'addr', 'port'

client_factory

alias of `TelnetClient`

connect_factory

alias of `ConnectTelnet`

`x84.telnet.debug_option` (*func*)

This function is a decorator that debug prints the 'from' address for callables decorated with this. This helps during telnet negotiation, to understand which function sets or checks local or remote option states.

`x84.telnet.name_option` (*option*)

Perform introspection of global CONSTANTS for equivalent values, and return a string that displays its possible meanings

9.4 x84.ssh

9.5 x84.rlogin

rlogin server for x84.

This only exists to demonstrate alternative client protocols rather than only ssh or telnet. rlogin is a very insecure and not recommended!

class `x84.rlogin.ConnectRLogin` (*client*)

Bases: `x84.client.BaseConnect`

rlogin protocol connection handler.

Takes care of the (initial) handshake, terminal and session setup.

Class initializer.

TIME_NEGOTIATE = 5.0

maximum time elapsed allowed for on-connect negotiation

TIME_POLL = 0.1

poll interval for on-connect negotiation

apply_environment (*parsed*)

Cherry-pick rlogin values into client environment variables.

Parameters **parsed** (*dict*) – values identified by class method `parse_connect_data()`

Return type None

get_connect_data ()

Receive four null-terminated strings transmitted by client on-connect.

Returns bytes received, containing at least 4 NUL-terminated strings.

Return type str

Raises ValueError on-connect data timeout or bandwidth exceeded.

parse_connect_data (*data*)

Parse and return raw data received by client on-connect.

Parameters **data** (*str*) – bytes received by class method `get_connect_data()`.

Returns dictionary containing pertinent key/values

Return type dict

run ()

Perform rfc1282 (rlogin) connection establishment.

Determine rlogin on-connect data, rlogin may only negotiate session user name and terminal type.

```

class x84.rlogin.RLoginClient (sock, address_pair, on_naws=None)
    Bases: x84.client.BaseClient

    rlogin protocol client handler.

    recv_ready ()
        Whether data is awaiting on the telnet socket.

    send ()
        Send any data buffered and return number of bytes send.

        Raises Disconnected client has disconnected (cannot write to socket).

    send_ready ()
        Whether any data is buffered for delivery.

    send_urgent_str (bstr)
        Buffer urgent (OOB) message to client from bytestring.

class x84.rlogin.RLoginServer (config)
    Bases: x84.server.BaseServer

    RLogin/RSH protocol server.

    Class initializer.

    client_factory
        alias of RLoginClient

    client_fds ()
        Return list of rlogin client file descriptors.

    connect_factory
        alias of ConnectRLogin

```

9.6 x84.sftp

9.7 x84.webserve

Service Plugins

10.1 x84.fail2ban

fail2ban module for x/84.

To enable, add to default.ini:

```
[fail2ban]
enabled = yes
```

The following options are available, but not required:

- `ip_blacklist`: space-separated list of IPs on permanent blacklist.
- `ip_whitelist`: space-separated list of IPs to always allow.
- `max_attempted_logins`: max no. of logins allowed for given time window
- `max_attempted_logins_window`: the length (in seconds) of the time window for which logins will be tracked (sliding scale).
- `initial_ban_length`: ban length (in seconds) when an IP is blacklisted.
- `ban_increment_length`: amount of time (in seconds) to add to a ban on subsequent login attempts

`x84.fail2ban.get_fail2ban_function()`

Return a function used to ban aggressively-connecting clients.

This is analogous to the ‘fail2ban’ utility, for example, telnet or ssh connect scanners.

Returns a function which may be passed an IP address, returning True if the connection from address `ip` should be accepted.

Returns function accepting ip address, returning boolean

Return type callable

10.2 x84.msgpoll

x84net message poll for x/84.

`x84.msgpoll.do_poll(networks)`

Message polling process.

Function is called periodically by `poller()`.

`x84.msgpoll.get_last_msg_id(last_file)`
Get the “last message id” by data file `last_file`.

`x84.msgpoll.get_networks()`
Get list configured message networks.

`x84.msgpoll.get_token(network)`
get token for authentication

`x84.msgpoll.main(background_daemon=True)`
Entry point to configure and begin network message polling.
Called by `x84/engine.py`, function `main()` as unmanaged thread.

Parameters `background_daemon` (*bool*) – When True (default), this function returns and web modules are served in an unmanaged, background (daemon) thread. Otherwise, function call to `main()` is blocking.

Return type None

`x84.msgpoll.poll_network_for_messages(net)`
Poll for new messages of network, `net`.

`x84.msgpoll.poller(poll_interval)`
Blocking function periodically polls configured message networks.

`x84.msgpoll.prepare_message(msg, network, parent)`
turn a `Msg` object into a dict for transfer

`x84.msgpoll.publish_network_messages(net)`
Push messages to network, `net`.

`x84.msgpoll.pull_rest(net, last_msg_id)`
pull messages for a given network newer than the ‘last’ message idx

`x84.msgpoll.push_rest(net, msg, parent)`
push message for a given network and append an origin line

11.1 x84.terminal

Terminal handler for x/84

class `x84.terminal.Terminal` (*kind, stream, rows, columns*)

Bases: `blessed.terminal.Terminal`

A thin wrapper over `blessed.Terminal`.

Class initializer.

cbreak (**args, **kws*)

Dummy method yields nothing for blessed compatibility.

getch () → unicode

Read and decode next byte from keyboard stream. May return u'' if decoding is not yet complete, or completed unicode character. Should always return bytes when `self.kbhit()` returns True.

Implementors of input streams other than `os.read()` on the `stdin` fd should derive and override this method.

inkey (*timeout=None, [esc_delay, [_intr_continue]]*) → *Keypress()*

Receive next keystroke from keyboard (`stdin`), blocking until a keypress is received or `timeout` elapsed, if specified.

When used without the context manager `cbreak`, `stdin` remains line-buffered, and this function will block until return is pressed, even though only one unicode character is returned at a time..

The value returned is an instance of `Keystroke`, with properties `is_sequence`, and, when True, non-None values for attributes `code` and `name`. The value of `code` may be compared against attributes of this terminal beginning with *KEY*, such as `KEY_ESCAPE`.

To distinguish between `KEY_ESCAPE`, and sequences beginning with escape, the `esc_delay` specifies the amount of time after receiving the escape character (`chr(27)`) to seek for the completion of other application keys before returning `KEY_ESCAPE`.

Normally, when this function is interrupted by a signal, such as the installment of `SIGWINCH`, this function will ignore this interruption and continue to poll for input up to the `timeout` specified. If you'd rather this function return u'' early, specify a value of `False` for `_intr_continue`.

is_a_tty

Dummy property always returns True.

kbhit (*[timeout=None]*) → bool

Returns True if a keypress has been detected on keyboard.

When `timeout` is 0, this call is non-blocking, Otherwise blocking until keypress is detected (default).
When `timeout` is a positive number, returns after `timeout` seconds have elapsed.

If input is not a terminal, False is always returned.

raw (**args, **kws*)

Dummy method yields nothing for blessed compatibility.

session

Session associated with this terminal.

set_keyboard_decoder (*encoding*)

Set or change incremental decoder for keyboard input.

class `x84.terminal.TerminalProcess` (*client, sid, master_pipes*)

Bases: `object`

Class record for tracking “terminals”.

Probably of most interest, is that a `TerminalProcess` is an abstract association with a `multiprocessing.Process` sub-process, and its i/o queues (`master_pipes`).

This is not a really tty, or even a pseudo-tty (pty)! No `termios`, `fnctl`, or any terminal driver i/o is performed, it is all virtual.

An instance of this class is stored using `register_tty()` and removed by `unregister_tty()`, and discovered using `get_terminals()`.

Class constructor.

`x84.terminal.determine_encoding` (*env*)

Determine and return preferred encoding given session `env`.

`x84.terminal.find_tty` (*client*)

Given a client, return a matching tty, or `None` if not registered.

`x84.terminal.flush_queue` (*queue*)

Flush all data awaiting on the ipc queue.

Seeks any remaining events in queue, used before closing to prevent zombie processes with IPC waiting to be picked up.

`x84.terminal.get_terminals` ()

Returns a list of all terminals as tuples (session-id, ttys).

`x84.terminal.init_term` (*writer, env*)

Determine the final `TERM` and encoding and return a `Terminal`.

`curses` is initialized using the value of ‘`TERM`’ of dictionary `env`, as well as a starting window size of ‘`LINES`’ and ‘`COLUMNS`’. If the terminal-type is of ‘`ansi`’ or ‘`ansi-bbs`’, then the `cp437` encoding is assumed; otherwise ‘`utf8`’.

A blessed-abstracted `curses` terminal is returned.

`x84.terminal.kill_session` (*client, reason='killed'*)

Given a client, shutdown its socket and signal subprocess exit.

`x84.terminal.on_naws` (*client*)

Callback for telnet `NAWS` negotiation.

On a Telnet `NAWS` sub-negotiation, check if client is yet registered in registry, and if so, send a ‘refresh’ event down the event queue.

This is ultimately handled by `x84.bbs.session.Session.buffer_event()`.

`x84.terminal.register_tty(tty)`
 Register a `TerminalProcess` instance.

`x84.terminal.spawn_client_session(client, matrix_kwargs=None)`
 Spawn sub-process for connecting client.

Optional

`x84.terminal.start_process(sid, env, CFG, child_pipes, kind, addrport, matrix_args=None, matrix_kwargs=None)`
 A `multiprocessing.Process` target.

Parameters

- **sid** (*str*) – string describing session source (IP address & port).
- **env** (*dict*) – dictionary of client environment variables (must contain at least 'TERM').
- **CFG** (*ConfigParser.ConfigParser*) – bbs configuration
- **child_pipes** (*tuple*) – tuple of (writer, reader) for engine IPC.
- **kind** (*str*) – what kind of connection as string, 'telnet', 'ssh', etc.
- **addrport** (*tuple*) – (client-ip, client-port) as string and integer.
- **matrix_args** (*tuple*) – optional positional arguments to pass to matrix script.
- **matrix_kwargs** (*dict*) – optional keyword arguments to pass to matrix script.

`x84.terminal.translate_ttype(ttype)`
 Return preferred terminal type given the session-negotiation ttype.

This provides a kind of coercion; we know some terminals, such as `SyncTerm` report a terminal type of 'ansi' – however, the author publishes a `termcap-ansi` database for 'ansi-bbs' which he instructs should be used! So an `[system]` configuration item of `termcap-ansi` may be set to 'ansi-bbs' to coerce such terminals for `SyncTerm`-centric telnet servers – though I would not recommend it.

Furthermore, if the ttype is (literally) 'unknown', then a system-wide default terminal type may be returned, also by `[system]` configuration option `termcap-unknown`.

`x84.terminal.unregister_tty(tty)`
 Unregister a `TerminalProcess` instance.

Userland/scripting API

12.1 x84.bbs.door

Door package for x/84.

This implements the concept of “Doors”, popular for DOS BBS software.

It also supports executing external Unix paths. See wikipedia article for details: http://en.wikipedia.org/wiki/BBS_door

class x84.bbs.door.DOSDoor (*cmd*='/bin/uname', *args*=(), *env*=None, *cp437*=True)
 Bases: x84.bbs.door.Door

Door-derived class with special handlers for executing dosemu.

This Door-derived class removes the “report cursor position” query sequence, which is sent by DOSEMU on startup. It also removes the “switch to alternate screen mode” set and reset (blessings terminals provide this with the context manager, using statement `with term.fullscreen() :`).

It would appear that any early keyboard input received (esp. in response to “report cursor position”) prior to DOOR execution in DOSEMU causes all input to be bitshifted and invalid and/or broken.

This class resolves that issue by overriding `output_filter` to remove such sequences, and `input_filter` which only allows input after a few seconds have elapsed.

Class initializer.

Parameters

- **cmd** (*str*) – full path of command to execute.
- **args** (*tuple*) – command arguments as tuple.
- **cp437** (*bool*) – When true, forces decoding of external program as codepage 437. This is the most common encoding used by DOS doors.
- **env** (*dict*) – Environment variables to extend to the sub-process. You should more than likely specify values for TERM, PATH, HOME, and LANG.

RE_REPWITH_CLEAR = '\033\[(1;80H.*\033\[(1;1HH\033\[(2J\|d+;1H.*\033\[(1;1H)'
 regular expression of sequences to be replaced by `term.clear` during `START_BLOCK` delay in `output_filter`

RE_REPWITH_NONE = '\033\[(6n\|?1049[lh]\|d+;\|d+r|1;1H\033\[\|dM)'
 regular expression of sequences to strip entirely during `START_BLOCK` delay in `output_filter`.

START_BLOCK = 4.0

Number of seconds to allow to elapse for `input_filter` and `output_filter` as a workaround for stripping startup sequences and working around a strange keyboard input bug.

input_filter (*data*)

filter keyboard input (used for “throwaway” bug workaround).

output_filter (*data*)

filter screen output (removes dosemu startup sequences).

resize ()

Signal resize of terminal to DOS – does nothing.

run ()

Begin door execution.

`pty.fork()` is called, child process calls `execvpe()` while the parent process pipes telnet session IPC data to and from the slave pty until child process exits.

On exit, DOSDoor flushes any keyboard input; DOSEMU appears to send various terminal reset sequences that may cause a reply to be received on input, and later as an invalid menu command.

class `x84.bbs.door.Door` (*cmd*=`'/bin/uname'`, *args*=(), *env*=`None`, *cp437*=`False`)

Bases: `object`

Spawns a subprocess and pipes input and output over bbs session.

Class initializer.

Parameters

- **cmd** (*str*) – full path of command to execute.
- **args** (*tuple*) – command arguments as tuple.
- **cp437** (*bool*) – When true, forces decoding of external program as codepage 437. This is the most common encoding used by DOS doors.
- **env** (*dict*) – Environment variables to extend to the sub-process. You should more than likely specify values for TERM, PATH, HOME, and LANG.

input_filter (*data*)

Derive and modify to implement a keyboard-input filter.

When keyboard input is detected, this method may filter such input. This base class method simply returns data as-is.

output_filter (*data*)

Filter output (performs cp437 encoding).

Given door output in bytes, if ‘cp437’ is specified in class initializer, convert to utf8 glyphs using cp437 encoding; otherwise decode output naturally as utf8.

resize ()

Signal resize of terminal to pty.

run ()

Begin door execution.

`pty.fork()` is called, child process calls `execvpe()` while the parent process pipes session IPC data to and from the slave pty, until the child process exits.

class `x84.bbs.door.Dropfile` (*filetype*=`None`, *node*=`None`)

Bases: `object`

Dropfile export class.

From http://en.wikipedia.org/wiki/BBS_door

> the 1990s on, most BBS software had the capability to “drop to” doors. > Several standards were developed for passing connection and user > information to doors; this was usually done with “dropfiles”, small > binary or text files dropped into known locations in the BBS’s file > system.

Class initializer.

Parameters

- **filetype** (*int*) – dropfile type. One of `Dropfile.DOORSYS`, `Dropfile.DOOR32`, `Dropfile.CALLINFOBBS`, or `Dropfile.DORINFO`.
- **node** (*int*) – A node number specified by caller; for some DOS doors, this is a very specific and limited number bounded and lock-acquired per-door by `sesame.py`. For others, it is inconsequential, in which case the session’s system-wide node number is used.

alias

current session’s handle.

comhandle

Com handle (always returns 0).

comport

Com port (always returns COM1).

comspeed

Com speed (always returns 57600).

comtype

Com type (always returns 0).

filename

Filename of given dropfile.

fullname

User fullname. Returns <handle> <handle>.

lastcall_date

Date of last call (format is %m/%d/%y).

lastcall_time

Time of last call (format is %H:%M).

location

User location.

node

User’s node number.

numcalls

Number of calls by user.

pageheight

Terminal height.

parity

Data parity.

password

Password of user.

remaining_mins
Remaining minutes (always returns 256).

remaining_secs
Remaining seconds (always returns 15360).

save (*folder*)
Save dropfile to destination *folder*.

securitylevel
User security level. Always 30, or 100 for sysop.

sysopname
name of sysop.

systemname
BBS System name.

time_used
Time used (session duration) in seconds.

usernum
User record number.

xferprotocol
preferred transfer protocol.

12.2 x84.bbs.editor

Editor package for x/84.

```
class x84.bbs.editor.LineEditor (width=None, content='u', hidden=False, colors=None,
                                glyphs=None, keyset=None)
```

Bases: `object`

This unicode line editor is unaware of its (y, x) position.

It is great for prompting a quick phrase on any terminal, such as a `login:` prompt.

Class initializer.

Parameters

- **width** (*int*) – the maximum input length.
- **content** (*str*) – given default content.
- **hidden** (*str*) – When non-False, a single ‘mask’ character for output.
- **colors** (*dict*) – optional dictionary containing key ‘highlight’.
- **glyphs** (*dict*) – optional dictionary of window border characters.
- **keyset** (*dict*) – optional dictionary of line editing values.

carriage_returned

Whether the carriage return character has been handled.

hidden

When non-False, a single ‘mask’ character for hiding input.

Used by password prompts.

init_keystrokes (*keyset*)

Sets keyboard keys for various editing keystrokes.

init_theme (*colors=None, glyphs=None, hidden=False*)

Set color, bordering glyphs, and hidden attribute theme.

process_keystroke (*keystroke*)

Process the keystroke and return string to refresh.

Parameters **keystroke** (*blessed.keyboard.Keystroke*) – input from `inkey()`.

Return type str

Returns string sequence suitable for refresh.

quit

Whether a ‘quit’ character has been handled, such as escape.

read()

Reads input until the ENTER or ESCAPE key is pressed (Blocking).

Allows backspacing. Returns unicode text, or None when canceled.

refresh()

Return string sequence suitable for refreshing editor.

No movement or positional sequences are returned.

width

Limit of characters to receive on input.

`x84.bbs.editor.PC_KEYSET = {'exit': [u'\x1b'], 'backspace': [u'\x08', u'\x7f'], 'backward': [u'\x17'], 'refresh': [u'\x0c']}`,
default command-key mapping.

class `x84.bbs.editor.ScrollingEditor` (**args, **kwargs*)

Bases: `x84.bbs.ansiwin.AnsiWindow`

A single line Editor, requires absolute (`yloc`, `xloc`) position.

Infinite horizontal scrolling is enabled or limited using `max_length`.

Class initializer.

Parameters

- **width** (*int*) – width of window.
- **yloc** (*int*) – y-location of window.
- **xloc** (*int*) – x-location of window.
- **max_length** (*int*) – maximum length of input (even when scrolled).
- **colors** (*dict*) – color theme.
- **glyphs** (*dict*) – bordering window character glyphs.
- **keyset** (*dict*) – command keys, global `PC_KEYSET` is default.

add (*u_chr*)

Return output sequence of changes after adding a character to editor.

An empty string is returned if no data could be inserted. Sequences for re-displaying the full input line are returned when the character addition caused the window to scroll horizontally.

Otherwise, the input is simply returned to be displayed.

backspace ()

Remove character from end of buffer, scroll as necessary.

backward ()

Delete word behind cursor, using ‘ ‘ as boundary.

In gnu-readline this is unix-word-rubout (C-w).

bell

Whether the user has neared the margin.

carriage_returned

Whether the carriage return character has been handled.

content

The contents of the editor.

eol

Whether more input may be accepted (end of line reached).

fixate (*x_adjust=0*)

Return string sequence suitable for “fixating” cursor position.

Set *x_adjust* to -1 to position cursor ‘on’ the last character, or 0 for ‘after’ (default).

init_keystrokes (*keyset*)

Sets keyboard keys for various editing keystrokes.

init_theme (*colors=None, glyphs=None*)

Set color and bordering glyphs theme.

is_scrolled

Whether the horizontal editor is in a scrolled state.

margin_amt

Absolute number of columns from margin until bell is signaled.

Indicating that the end is near and the carriage should be soon returned.

margin_pct

Percentage of visible width from-end until bell is signaled.

Number of columns away from input length limit, as a percentage of its total visible width, that will alarm the bell. This simulates the bell of a typewriter as a signaling mechanism. Default is 10.

Unofficially intended for a faked multi-line editor: by using the bell as a wrap signal to instantiate another line editor and ‘return the carriage’.

max_length

Maximum line length.

This also limits infinite scrolling when *enable_scrolling* is True. When unset, the maximum length is infinite!

position

Tuple of shift amount and column position of line editor.

process_keystroke (*keystroke*)

Process the keystroke and return string to refresh.

Parameters *keystroke* (*blessed.keyboard.Keystroke*) – input from *inkey* () .

Return type str

Returns string sequence suitable for refresh.

quit

Whether a ‘quit’ character has been handled, such as escape.

read()

Reads input until the ENTER or ESCAPE key is pressed (Blocking).

Allows backspacing. Returns unicode text, or None when canceled.

refresh()

Return string sequence suitable for refreshing editor.

A strange by-product; if scrolling was not previously enabled, it is if wrapping must occur; this can happen if a non-scrolling editor was provided a very large .content buffer, then later .refresh()’d. – essentially enabling infinite scrolling.

scroll_amt

Number of columns from-end until horizontal editor will scroll

Calculated by scroll_pct.

scroll_pct

Percentage of visible width from-end until scrolling occurs.

Number of columns, as a percentage of its total visible width, that will be scrolled when a user reaches the margin by percent. Default is 25.

update (*ucs=u’*)

Replace or reset content.

Resets properties carriage_returned and quit to False.

12.3 x84.bbs.ini

Configuration package x/84.

x84.bbs.ini.CFG = None

Singleton representing configuration after load

x84.bbs.ini.get_ini (*section=None, key=None, getter='get', split=False, splitsep=', '*)

Get an ini configuration of section and key.

If the option does not exist, an empty list, string, or False is returned – return type decided by the given arguments.

The getter method is ‘get’ by default, returning a string. For booleans, use getter=‘get_boolean’.

To return a list, use split=True.

x84.bbs.ini.init (*lookup_bbs, lookup_log*)

Initialize global ‘CFG’ variable, a singleton to contain bbs settings.

Each variable (lookup_bbs, lookup_log) is tuple lookup path of in-order preferences for .ini files. If none are found, defaults are initialized, and the last item of each tuple is created.

x84.bbs.ini.init_bbs_ini ()

Returns ConfigParser instance of bbs system defaults.

x84.bbs.ini.init_log_ini ()

Return ConfigParser instance of logger defaults.

12.4 x84.bbs.ipc

Session IPC package for x/84.

class `x84.bbs.ipc.IPCLogHandler` (*out_queue*)
Bases: `logging.Handler`

Log handler that sends the log up the ‘event pipe’.

This is a rather novel solution that seems overlooked in documentation, a forked process must have some method to propagate its logging records up through the main process, otherwise they are lost.

Constructor method, requires `multiprocessing.Pipe`.

emit (*record*)
Emit log record via IPC output queue.

class `x84.bbs.ipc.IPCStream` (*writer*)
Bases: `object`

Connect blessed.Terminal argument ‘stream’ to ‘writer’ queue.

The `writer` queue is a `multiprocessing.Pipe` whose master-side is polled for output in `x84.engine`. Only the `write()` method of this “stream” and `is_a_tty` attribute is called or evaluated by `blessed.Terminal`. The attribute `is_a_tty` is mocked as `True`.

write (*ucs, encoding='ascii'*)
Sends unicode text to Pipe.

Default encoding is ‘ascii’, which is unset only when used with blessings, which rarely writes directly to the stream (context managers, such as “with `term.location(0, 0):`” have such side effects).

`x84.bbs.ipc.make_root_logger` (*out_queue*)
Remove and re-address the root logging handler.

Any existing handlers of the current process are removed and the root logger is re-address to send via an IPC output event queue.

12.5 x84.bbs.lightbar

Lightbar package for x/84.

class `x84.bbs.lightbar.Lightbar` (**args, **kwargs*)
Bases: `x84.bbs.ansiwin.AnsiWindow`

This Windowing class offers a classic ‘lightbar’ interface.

Instantiate with `yloc`, `xloc`, `height`, and `width`, then call the `update` method with a list of unicode strings. send keycodes to `process_keystroke()` to interactive with the ‘lightbar’.

Class initializer.

Initialize a lightbar of height, width, y and x, and position.

Parameters

- **width** (*int*) – width of window.
- **height** (*int*) – height of window.
- **yloc** (*int*) – y-location of window.
- **xloc** (*int*) – x-location of window.

- **colors** (*dict*) – color theme, only key value of `highlight` is used.
- **glyphs** (*dict*) – bordering window character glyphs.
- **keyset** (*dict*) – command keys, global `NETHACK_KEYSET` is used by default, augmented by application keys such as `home`, `end`, `pgup`, etc.
- **content** (*list*) – Lightbar content as list of tuples, an empty list is used by default. Tuples must be in form of `(key, str)`. `key` may have any suitable significance for the caller. `str`, however, must be of a unicode terminal sequence.

at_bottom

Whether current selection is pointed at final entry.

at_top

Whether current selection is pointed at the first entry.

fixate ()

Return string sequence suitable for “fixating” cursor position.

goto (*index*)

Move selection to given index.

index

Selected index of `self.content`.

init_keystrokes (*keyset*)

Sets keyboard keys for various editing keystrokes.

init_theme (*colors=None, glyphs=None*)

Set color and bordering glyphs theme.

last_index

Previously selected index of `self.content`.

move_down ()

Move selection down one row, return string suitable for refresh.

move_end ()

Move selection to final row, return string suitable for refresh.

move_home ()

Move selection to first row, return string suitable for refresh.

move_pagedown ()

Move selection down one page, return string suitable for refresh.

move_pageup ()

Move selection up one page, return string suitable for refresh.

move_up ()

Move selection up one row, return string suitable for refresh.

position

Tuple pair (row, page).

`row` is the index from top of window, and ‘page’ is number of page items scrolled.

process_keystroke (*key*)

Process the keystroke and return string to refresh.

Parameters `keystroke` (*blessed.keyboard.Keystroke*) – input from `inkey` () .

Return type `str`

Returns string sequence suitable for refresh.

quit

Whether a 'quit' character has been handled, such as escape.

read()

Reads input until the ENTER or ESCAPE key is pressed (Blocking).

Returns selection content, or None when canceled.

refresh()

Return string sequence suitable for refreshing lightbar.

refresh_quick()

Redraw only the 'dirty' portions after a 'move' has occurred.

refresh_row(row)

Return string sequence suitable for refreshing current selection.

Return unicode byte sequence suitable for moving to location ypos of window-relative row, and displaying any valid entry there, or using glyphs['erase'] if out of bounds. Strings are ansi color safe, and will be trimmed using glyphs['strip'] if their displayed width is wider than window.

selected

Whether carriage return was detected by process_keystroke.

selection

Selected content of self.content by index.

update(keyed_uchars=None)

Replace content with with sequence of (key, str).

key may have any suitable significance for the caller. str, however, must be of a unicode terminal sequence.

visible_bottom

Visible bottom-most item of lightbar.

visible_content

Returns visible content only.

vitem_idx

Relative visible item index within view.

Index of selected item relative by index to only the length of the list that is visible, without accounting for scrolled content.

vitem_shift

Index of top-most item in viewable window, non-zero when scrolled.

This value effectively represents the number of items not in view due to paging.

x84.bbs.lightbar.NETHACK_KEYSET = {'down': [u'j'], 'pgdown': [u'l', u'f'], 'exit': [u'q', u'Q', u'\x1b'], 'end': [u'n'], 'G

default command-key mapping.

12.6 x84.bbs.output

Terminal output package for x/84.

x84.bbs.output.RE_ANSI_COLOR = <_sre.SRE_Pattern object at 0x7fbc371441e0>
simple regular expression for matching simple ansi colors, for use by encode_pipe().

`x84.bbs.output.SAUCE_FONT_MAP = {'IBM VGA50 865': 'cp865', 'IBM EGA 720': 'cp720', 'IBM VGA25g 437': 'cp437'}`
 Translation map for embedded font hints in SAUCE records as documented at <http://www.acid.org/info/sauce/sauce.htm> section `FontName`. Used by `showart()` to automatically determine which codepage to be used by utf8 terminals to provide an approximate translation.

`x84.bbs.output.SYNCTERM_FONTMAP = ('cp437', 'cp1251', 'koi8_r', 'iso8859_2', 'iso8859_4', 'cp866', 'iso8859_9', 'haik8')`
 A mapping of SyncTerm fonts/code pages to their sequence value, for use as argument `font_name` of `syncterm_setfont()`.

Where matching, their python-standard encoding value is used, (fe. 'cp437'). Otherwise, the lower-case named of the font is used.

source: <http://cvs.synchro.net/cgi-bin/viewcvs.cgi/checkout/src/conio/cterm.txt>

`x84.bbs.output.decode_pipe(ucs)`
 Return ucs containing 'pipe codes' with terminal color sequences.

These are sometimes known as LORD codes, as they were used in the DOS Door game of the same name. Compliments `encode_pipe()`.

Parameters `ucs (str)` – string containing 'pipe codes'.

Return type `str`

`x84.bbs.output.echo(ucs)`
 Display unicode terminal sequence.

Parameters `ucs (str)` – unicode sequence to write to terminal.

`x84.bbs.output.encode_pipe(ucs)`
 Given a string containing ECMA-48 sequence, replace with "pipe codes".

These are sometimes known as LORD codes, as they were used in the DOS Door game of the same name. Compliments `decode_pipe()`.

Parameters `ucs (str)` – string containing ECMA-48 sequences.

Return type `str`

`x84.bbs.output.from_cp437(text)`
 Deprecated form of `bytes.decode('cp437_art')`.

`x84.bbs.output.ropen(filename, mode='rb')`
 Open random file using wildcard (glob).

`x84.bbs.output.showart(filepattern, encoding=None, auto_mode=True, center=False, poll_cancel=False, msg_cancel=None, force=False)`
 Yield unicode sequences for any given ANSI Art (of `art_encoding`).

Effort is made to parse SAUCE data, translate input to unicode, and trim artwork too large to display. If `poll_cancel` is not `False`, represents time as float for each line to block for keypress – if any is received, then iteration ends and `msg_cancel` is displayed as last line of art.

If you provide no `encoding`, the piece encoding will be based on either the encoding in the SAUCE record, the configured default or the default fallback CP437 encoding.

Alternate codecs are available if you provide the `encoding` argument. For example, if you want to show an Amiga style ASCII art file:

```
>>> from x84.bbs import echo, showart
>>> for line in showart('test.asc', 'topaz'):
...     echo(line)
```

The `auto_mode` flag will, if set, only respect the selected encoding if the active session is UTF-8 capable.

If `center` is set to `True`, the piece will be centered respecting the current terminal's width.

If `force` is set to `true` then the artwork will be displayed even if it's wider than the screen.

`x84.bbs.output`.**syncterm_setfont** (*font_name*, *font_page=0*)

Send SyncTerm's sequence for selecting a "font" codepage.

Parameters

- **font_name** (*str*) – any value of `SYNCTERM_FONTMAP`.
- **font_page** (*int*) –

Reference:

```
CSI [ p1 [ ; p2 ] ] sp D
Font Selection
Defaults: p1 = 0 p2 = 0
"sp" indicates a single space character.
Sets font p1 to be the one indicated by p2. Currently only the primary
font (Font zero) and secondary font (Font one) are supported. p2 must
be between 0 and 255. Not all output types support font selection.
Only X11 and SDL currently do.
```

source: <http://cvs.synchro.net/cgi-bin/viewcvs.cgi/checkout/src/conio/cterm.txt>

`x84.bbs.output`.**timeago** (*secs*, *precision=0*)

Return human-readable string of seconds elapsed.

Parameters

- **secs** (*int*) – number of seconds "ago".
- **precision** (*int*) – optional decimal precision of returned seconds.

Pass a duration of time and return human readable shorthand, fe:

```
>>> asctime(126.32)
' 2m 6s',
>>> asctime(70.9999, 2)
' 1m 10.99s'
```

12.7 x84.bbs.pager

Pager package for x/84.

class `x84.bbs.pager.Pager` (**args*, ***kwargs*)

Bases: `x84.bbs.ansiwin.AnsiWindow`

Scrolling viewer.

Class initializer.

Parameters

- **width** (*int*) – width of window.
- **height** (*int*) – height of window.
- **yloc** (*int*) – y-location of window.
- **xloc** (*int*) – x-location of window.

- **content** (*str*) – initial pager contents.
- **colors** (*dict*) – color theme.
- **glyphs** (*dict*) – bordering window character glyphs.
- **keyset** (*dict*) – command keys, global `VI_KEYSET` is default.

append (*ucs*)

Update content buffer with additional line(s) of text.

“pipe codes” in `ucs` are decoded by `decode_pipe()`.

Parameters `ucs` (*str*) – unicode string to append-to content buffer.

:rtype `str` :return: terminal sequence suitable for refreshing window.

bottom

Bottom-most position that contains content.

content

Content of pager.

Return value is “pipe encoded” by `encode_pipe()`. **:rtype:** `str`

init_keystrokes (*keyset*)

Sets keyboard keys for various editing keystrokes.

move_down (*num=1*)

Scroll down `num` rows and return refresh string.

Return type `str`

move_end ()

Scroll to bottom and return refresh string.

Return type `str`

move_home ()

Scroll to top and return refresh string.

Return type `str`

move_pgdown (*num=1*)

Scroll down `num` pages and return refresh string.

Return type `str`

move_pgup (*num=1*)

Scroll up `num` pages and return refresh string.

Return type `str`

move_up (*num=1*)

Scroll up `num` rows and return refresh string.

Return type `str`

position

Index of content buffer displayed at top of window.

position_last

Previous position before last move.

process_keystroke (*keystroke*)

Process the keystroke and return string to refresh.

Parameters `keystroke` (*blessed.keyboard.Keystroke*) – input from `inkey()`.

Return type `str`

Returns string sequence suitable for refresh.

quit

Whether a ‘quit’ character has been handled, such as escape.

read()

Blocking read-eval-print loop for pager.

Processes user input, taking action upon and refreshing pager until the escape key is pressed.

Return type `None`

refresh (*start_row=0*)

Return unicode string suitable for refreshing pager window.

Parameters `start_row` (*int*) – refresh from only visible row ‘start_row’ and downward. This can be useful if only the last line is modified; or in an ‘insert’ operation: only the last line need be refreshed.

Return type `str`

refresh_row (*row*)

Return unicode string suitable for refreshing pager row.

Parameters `row` (*int*) – target row by visible index.

Return type `str`

update (*ucs*)

Update content buffer with newline-delimited text.

Return type `str`

visible_bottom

Bottom-most window row that contains content.

visible_content

Content that is visible in window.

12.8 x84.bbs.session

Session engine for x/84.

`x84.bbs.session.SESSION = None`

singleton representing the session connected by current process

class `x84.bbs.session.Session` (*terminal, sid, env, child_pipes, kind, addrport, matrix_args, matrix_kwargs*)

Bases: `object`

A per-process Session. Begins by the `run()`.

Instantiate a Session.

Only one session may be instantiated per process.

Parameters

- **terminal** (*blessed.Terminal*) – interactive terminal associated with this session.

- **sid** (*str*) – session identification string
- **env** (*dict*) – transport-negotiated environment variables, should contain at least values for TERM and ‘encoding’.
- **child_pipes** (*tuple*) – tuple of (*writer*, *reader*).
- **kind** (*str*) – transport description string (ssh, telnet)
- **addrport** (*str*) – transport ip address and port as string
- **matrix_args** (*tuple*) – When non-None, a tuple of positional arguments passed to the matrix script.
- **matrix_kwargs** (*dict*) – When non-None, a dictionary of keyword arguments passed to the matrix script.

activity

Current session activity.

This is arbitrarily set by session scripts.

This also updates xterm titles, and is globally broadcasted as a “current activity” in the Who’s online script, for example.

buffer_event (*event*, *data=None*)

Buffer and handle IPC data keyed by *event*.

Parameters

- **event** (*str*) – event name.
- **data** – event data.

Return type bool

Returns True if the event was internally handled, and the caller should take no further action.

Methods internally handled by this method:

- **global**: events where the first index of *data* is AYT. This is sent by other sessions using the broadcast event, to discover “who is online”.
- **info-req**: Where the first data value is the remote session-id that requested it, expecting a return value event of **info-ack** whose data values is a dictionary describing a session. This is an extension of the “who is online” event described above.
- **gosub**: Allows one session to send another to a different script, this is used by the default board `chat.py` for a chat request.

buffer_input (*data*, *pushback=False*)

Receive keyboard input, ‘*data*’, into `input` buffer.

Updates idle time, buffering raw bytes received from telnet client via event queue. Sometimes a script may poll for, and receive keyboard data, but wants to push it back in to the top of the stack to be decoded by a later call to `term.inkey()`; in such case, `pushback` should be set.

Parameters

- **data** (*bytes*) – keyboard input data.
- **pushback** (*bool*) – whether it should be pushed to front of stack.

close ()

Close session, currently releases `node` lock..

connect_time

Time of session start (as float).

current_script

The current script being executed.

duration

Seconds elapsed since connection began (as float).

encoding

Session encoding, both input and output.

flush_event (*event*)

Flush and return all data buffered for *event*.

Parameters *event* (*str*) – event name.

Return type list

idle

Seconds elapsed since last keypress as float.

last_input_time

Time of last keypress (as epoch, float).

node

Unique numeric constant for this session.

This makes it simpler to refer to users who are online, instead of by their full session-id (such as telnet-92.32.10.132:57331) one can simply refer to node #1, etc..

pid

Process ID of this session (int).

poll_event (*event*)

Non-blocking poll for session event.

Parameters *event* (*str*) – an IPC event queue by name, such as `input`.

Returns first matching IPC event data, or `None`.

read_event (*event*, *timeout=None*)

Return data for given *event* by timeout.

Parameters

- **event** (*str*) – an IPC event queue by name, such as `input`.
- **timeout** (*int*) – Value of `None` is blocking (default), `-1` is non-blocking poll. All other values are blocking up to value of timeout.

Returns first matching IPC event data. If timeout is specified and no matching IPC event is discovered, `None` is returned.

read_events (*events*, *timeout=None*)

Return the first matched IPC data for any event specified by timeout.

Parameters

- **events** (*tuple*) – events to search for, for example (`'input'`, `'refresh'`).
- **timeout** (*int*) – Value of `None` is blocking (default), `-1` is non-blocking poll. All other values are blocking up to value of timeout.

Return type tuple

Returns first matching IPC event, data tuple, where event matches one of the given events. If timeout is specified and no matching IPC event is discovered, (None, None) is returned.

run()

Begin main execution of session.

Scripts manipulate control flow of scripts by raising the Goto exception, or the gosub function.

runscript (*script*)

Execute the main() callable of script identified by script.

Parameters **script** (*Script*) – target script to execute.

Returns the return value of the given script's main() function.

script_module

Base python module instance for userland scripts.

Return type list

script_path

Base filepath folder for all scripts.

Return type list

send_event (*event, data*)

Send data to IPC output queue in form of (event, data).

Supported event strings:

- **disconnect**: Session wishes to disconnect.
- **logger**: Data is logging record, used by IPCLogHandler.
- **output**: Unicode data to write to client.
- **global**: Broadcast event to other sessions.
- **route**: Send an event to another session.
- **db-<schema>**: Request sqlite dict method result.
- **db=<schema>**: Request sqlite dict method result as iterable.
- **lock-<name>**: Fine-grained global bbs locking.

Parameters

- **event** (*str*) – event name.
- **data** – event data.

show_traceback

Whether traceback errors should be displayed to user (bool).

tap_input

Whether keyboard input should be logged (bool).

tap_output

Whether screen output should be logged (bool).

to_dict ()

Dictionary describing this session.

user

User instance of this session.

write (*ucs*, *encoding=None*)

Write unicode data *ucs* to terminal.

`x84.bbs.session.disconnect` (*reason=u''*)

Disconnect session. Does not return.

`x84.bbs.session.getch` (*timeout=None*)

A deprecated form of `getterminal().inkey()`.

This is old behavior – upstream blessed project does the correct thing. please use `term.inkey()` and see the documentation for blessed’s `inkey()` method, it **always** returns unicode, never `None`, and definitely never an integer. However some internal UI libraries were built upon `getch()`, and as such, this remains ...

`x84.bbs.session.getsession` ()

Return `Session` instance of current process.

`x84.bbs.session.getterminal` ()

Return `blessed.Terminal` instance of current session.

`x84.bbs.session.gosub` (*script*, **args*, ***kwargs*)

Call bbs script with optional arguments, Returns value.

`x84.bbs.session.goto` (*script_name*, **args*, ***kwargs*)

Change bbs script. Does not return.

12.9 x84.bbs.userbase

Userbase record database and utility functions for x/84.

class `x84.bbs.userbase.Group` (*name*, *members=()*)

Bases: `object`

A simple group record object.

Class initializer.

add (*handle*)

Add user to group.

delete ()

Delete group record, enforces referential integrity with Users.

members

Members of this group as user handles.

name

Name of this group.

remove (*handle*)

Remove user from group.

save ()

Save group record to database.

class `x84.bbs.userbase.User` (*handle=u'anonymous'*)

Bases: `object`

A simple user record.

Class initializer.

auth (*try_pass*)

Authenticate user with given password, *try_pass*.

Return type bool

Returns whether the password is correct.

calls

Legacy, number of times user has ‘called’ this board.

delete ()

Remove user from user and group databases.

email

E-mail address. May be used for password resets.

get (*k*, *d*) → *D*[*k*] if *k* in *D*, else *d*. *d* defaults to None.

group_add (*group*)

Add user to group.

group_del (*group*)

Remove user from group.

groups

Set of groups user is a member of (set of strings).

handle

User handle, also the database key.

is_sysop

Whether the user is in the ‘sysop’ group.

lastcall

Time last called, *time.time()* epoch-formatted (float).

location

Legacy, used as a geographical location, group names, etc.

password

Password in encrypted form as tuple (salt, hash).

Not generally used directly, but by *auth()*.

The *setter* of this property is provided a password in plain-text and encrypts it as given.

If a password has not yet been set, it is (None, None).

save ()

Save user record to database.

x84.bbs.userbase.check_anonymous_user (*username*)

Boolean return when user is anonymous and is allowed.

x84.bbs.userbase.check_bye_user (*username*)

Boolean return when username matches *byecmds* in ini *cfg*.

x84.bbs.userbase.check_new_user (*username*)

Boolean return when username matches *newcmds* ini *cfg*.

x84.bbs.userbase.check_user_password (*username*, *password*)

Boolean return when username and password match user record.

`x84.bbs.userbase.check_user_pubkey(username, public_key)`
Boolean return when `public_key` matches user record.

`x84.bbs.userbase.find_user(handle)`
Discover and return matching user by `handle`, case-insensitive.

Returns matching handle as str, or None if not found.

Return type None or str.

`x84.bbs.userbase.get_digestpw()`
Returns singleton to password digest routine.

`x84.bbs.userbase.get_user(handle)`
Returns User record by handle.

Return type User

Returns instance of `User`

`x84.bbs.userbase.list_users()`
Returns all user handles.

Return type list

:returns list of user handles.

`x84.bbs.userbase.parse_public_key(user_pubkey)`
Return paramiko key class instance of a user's public key text.

12.10 x84.bbs.ansiwin

Ansi Windowing package for x/84.

class `x84.bbs.ansiwin.AnsiWindow(height, width, yloc, xloc, colors=None, glyphs=None)`

Bases: `object`

Provides position-relative drawing routines within a region.

The `AnsiWindow` base class provides position-relative window drawing routines to terminal interfaces, such as pager windows, editors, and lightbar lists, as well as some drawing niceties such as borders, text alignment

Class initializer for base windowing class.

Parameters

- **width** (*int*) – width of window.
- **height** (*int*) – height of window.
- **yloc** (*int*) – y-location of window.
- **xloc** (*int*) – x-location of window.
- **colors** (*dict*) – color theme.
- **glyphs** (*dict*) – bordering window character glyphs.

align (*text, width=None*)

Return `text` aligned to `width` using `self.alignment`.

When None (default), the visible width of this window is used.

alignment

Horizontal justification of text content for method `align`.

border ()
Return sequence suitable for drawing window border.

clear ()
Return sequence suitable for erasing contents window.

erase ()
Return sequence suitable for erasing full window (with border).

erase_border ()
Return sequence suitable for erasing only the window border.

footer (*text*)
Return sequence for displaying text on bottom border of window.

init_theme (*colors=None, glyphs=None*)
Set glyphs and colors appropriate for “theming”.

This is called by the class initializer.

isinview ()
Whether this window is in bounds of terminal dimensions.

iswithin (*win*)
Whether target window, *win* is within this windows bounds.

moved
Whether movement has occurred (bool).

pos (*yloc=None, xloc=None*)
Return sequence to move cursor to window-relative position.

resize (*height=None, width=None, yloc=None, xloc=None*)
Adjust window dimensions by given parameter.

title (*ansi_text*)
Return sequence for displaying text on top border of window.

visible_height
Visible height of window after accounting for padding.

visible_width
Visible width of window after accounting for padding.

willfit (*win*)
Whether target window, *win* is within this windows bounds.

xpadding
Horizontal padding of window border.

ypadding
Vertical padding of window border.

12.11 x84.bbs.dbproxy

Database proxy helper for x/84.

class `x84.bbs.dbproxy.DBProxy` (*schema, table='unnamed', use_session=True*)

Bases: `object`

Provide dictionary-like object interface to shared database.

A database call, such as `__len__()` or `keys()` is issued as a command to the main engine when `use_session` is `True`, which spawns a thread to acquire a lock on the database and return the results via IPC pipe transfer.

Class initializer.

Parameters

- **scheme** (*str*) – database key, becomes basename of `.sqlite3` file.
- **table** (*str*) – optional database table.
- **use_session** (*bool*) – Whether iterable returns should be sent over an IPC pipe (client is a `x84.bbs.session.Session` instance), or returned directly (such as used by the main thread engine components.)

acquire ()

Acquire system-wide lock on database.

copy () → a shallow copy of D

get (*k*, *d*) → `D[k]` if `k` in `D`, else `d`. `d` defaults to `None`.

has_key (*k*) → `True` if `D` has a key `k`, else `False`

items () → list of `D`'s (key, value) pairs, as 2-tuples

iteritems () → an iterator over the (key, value) items of `D`

iterkeys () → an iterator over the keys of `D`

itervalues () → an iterator over the values of `D`

keys () → list of `D`'s keys

pop (*k*, *d*) → `v`, remove specified key and return the corresponding value.
If key is not found, `d` is returned if given, otherwise `KeyError` is raised

popitem () → (`k`, `v`), remove and return some (key, value) pair as a 2-tuple; but raise `KeyError` if `D` is empty.

proxy_iter (*method*, **args*)

Proxy for iterable dictionary method calls.

proxy_iter_session (*method*, **args*)

Proxy for iterable-return method calls over session IPC pipe.

proxy_method (*method*, **args*)

Proxy for dictionary method calls.

proxy_method_direct (*method*, **args*)

Proxy for direct dictionary method calls.

proxy_method_session (*method*, **args*)

Proxy for dictionary method calls over IPC pipe.

release ()

Release system-wide lock on database.

setdefault (*k*, *d*) → `D.get(k,d)`, also set `D[k]=d` if `k` not in `D`

update (*[E]*, ***F*) → `None`. Update `D` from dict/iterable `E` and `F`.

If `E` present and has a `.keys()` method, does: for `k` in `E`: `D[k] = E[k]` If `E` present and lacks `.keys()` method, does: for (`k`, `v`) in `E`: `D[k] = v` In either case, this is followed by: for `k` in `F`: `D[k] = F[k]`

values () → list of `D`'s values

12.12 x84.bbs.exception

Custom exceptions for x/84.

exception `x84.bbs.exception.Disconnected`

Bases: `exceptions.Exception`

Thrown when a client is disconnected.

exception `x84.bbs.exception.Goto` (*script*, *args, **kwargs)

Bases: `exceptions.Exception`

Thrown to change script without returning.

12.13 x84.bbs.modem

File transfer routines for x/84.

`x84.bbs.modem.recv_modem` (*stream*, *protocol*='xmodem1k', *retry*=16, *timeout*=30)

Receive a file using 'xmodem1k' or 'xmodem' protocol into *stream*.

Currently, these are the only protocols supported. Returns `True` upon successful transmission, otherwise `False`.

Parameters

- **stream** – The file-like stream object to send data from.
- **retry** (*int*) – The maximum number of times to try to resend a failed packet before failing.
- **timeout** (*int*) – seconds to elapse for response before failing.

`x84.bbs.modem.send_modem` (*stream*, *protocol*='xmodem1k', *retry*=16, *timeout*=30, *callback*=None)

Send a file using 'xmodem1k' or 'xmodem' protocol.

Currently, these are the only protocols supported. Returns `True` upon successful transmission, otherwise `False`.

Parameters

- **stream** – The file-like stream object to send data from.
- **retry** (*int*) – The maximum number of times to try to resend a failed packet before failing.
- **timeout** (*int*) – seconds to elapse for response before failing.
- **callback** (*callable*) – Reference to a callback function that has the following signature. This is useful for getting status updates while a transfer is underway:

```
def callback(total_count, success_count, error_count)
```

12.14 x84.bbs.msgbase

Messaging database package for x/84.

class `x84.bbs.msgbase.Msg` (*recipient*=None, *subject*=u'', *body*=u'')

Bases: `object`

A record spec for messages held in the msgbase.

It contains many default properties to describe a conversation:

- `stime`, the time the message was sent.
- `author`, `recipient`, `subject`, and `body` are envelope parameters.
- `tags` is for use with message groupings, containing a list of strings that other messages may share in relation.
- `parent` points to the message this message directly refers to.
- `children` is a set of indices replied by this message.

ctime

Datetime message was instantiated

Return type `datetime.datetime`

queue_for_network()

Queue message for networks, hosting or sending.

save (*send_net=True, ctime=None*)

Save message to database, recording 'tags' db.

As a side-effect, it may queue message for delivery to external systems, when configured.

stime

Datetime message was saved to database

Return type `datetime.datetime`

`x84.bbs.msgbase.format_origin_line()`

Format origin line for message quoting.

`x84.bbs.msgbase.get_msg(idx=0)`

Return `Msg` record instance by index `idx`.

`x84.bbs.msgbase.get_origin_line()`

Return origin configuration item of `[msg]` section.

`x84.bbs.msgbase.list_msgs(tags=None)`

Return set of indices matching `tags`, or all by default.

`x84.bbs.msgbase.list_privmsgs(handle=None)`

Return all private messages for given user handle.

`x84.bbs.msgbase.list_tags()`

Return set of available tags.

`x84.bbs.msgbase.to_localtime(tm_value)`

convert given UTC time to local time

`x84.bbs.msgbase.to_utctime(tm_value)`

convert given local time to UTC time

12.15 `x84.bbs.selector`

Left/Right lightbar choice selector for x/84.

class `x84.bbs.selector.Selector` (*yloc, xloc, width, left, right, **kwargs*)

Bases: `x84.bbs.ansiwins.AnsiWindow`

A two-state horizontal lightbar interface.

Class initializer.

Initialize a selector of width, y x, and left/right values.

Parameters

- **width** (*int*) – width of window.
- **yloc** (*int*) – y-location of selector.
- **xloc** (*int*) – x-location of selector.
- **colors** (*dict*) – color theme, only key value of `selected` and `unselected` is used.
- **keyset** (*dict*) – command keys, global `VI_KEYSET` is used by default, augmented by application keys such as `home`, `end`, `pgup`, etc.
- **left** (*str*) – text string of left-side selection.
- **right** (*str*) – text string of right-side selection.

init_keystrokes (*keyset*)

Sets keyboard keys for various editing keystrokes.

left

Left-side value.

move_left ()

Move selection left, return string suitable for refresh.

move_right ()

Move selection right, return string suitable for refresh.

process_keystroke (*keystroke*)

Process the keystroke and return string to refresh.

Parameters **keystroke** (*blessed.keyboard.Keystroke*) – input from `inkey()`.

Return type `str`

Returns string sequence suitable for refresh.

quit

Whether a ‘quit’ character has been handled, such as escape.

read ()

Reads input until the ENTER or ESCAPE key is pressed (Blocking).

refresh ()

Return string sequence suitable for refresh.

right

Right-side value.

selected

Whether the carriage return character has been handled.

selection

Current selection.

toggle ()

Toggle selection, return string suitable for refresh.

12.16 x84.bbs.telnet

Utility functions for clients based on telnetlib for x/84.

`x84.bbs.telnet.callback_cmdopt` (*socket, cmd, opt, env_term=None, width=None, height=None*)
Callback for `telnetlib.Telnet.set_option_negotiation_callback`.

Future Directions

basic v3.0 roadmap:

- python3 using async i/o
- windows support, requires ansi.sys support emulation for PDCurses in blessed
- ftp, ftps, fxp support
- modeling (using 'schematics' project) for userbase, messagebase, etc.
- support for agoranet, zeronet, etc. messaging networks
- a classic "waiting for callers" screen and /dev/tty-line support. this was supported in previous versions, but dropped due to blessed's requirements of requiring a unique process for each terminal.
- start as daemon (-d)
- Convert messaging to data modeling format (schematics?) and rfc-compliant mail messaging.
- cron-like scheduling of scripts (fe., msgpoll.py)

Feel free to contribute ideas as a github issue.

Indexes

- *genindex*
- *modindex*

X

- x84.bbs.ansiwin, 58
- x84.bbs.dbproxy, 59
- x84.bbs.door, 39
- x84.bbs.editor, 42
- x84.bbs.exception, 61
- x84.bbs.ini, 45
- x84.bbs.ipc, 46
- x84.bbs.lightbar, 46
- x84.bbs.modem, 61
- x84.bbs.msgbase, 61
- x84.bbs.output, 48
- x84.bbs.pager, 50
- x84.bbs.selector, 62
- x84.bbs.session, 52
- x84.bbs.telnet, 64
- x84.bbs.userbase, 56
- x84.client, 26
- x84.db, 22
- x84.engine, 21
- x84.fail2ban, 33
- x84.msgpoll, 33
- x84.rlogin, 30
- x84.server, 25
- x84.telnet, 27
- x84.terminal, 35

A

accept() (in module x84.engine), 21
 acquire() (x84.bbs.dbproxy.DBProxy method), 60
 activity (x84.bbs.session.Session attribute), 53
 add() (x84.bbs.editor.ScrollingEditor method), 43
 add() (x84.bbs.userbase.Group method), 56
 addrport (x84.client.BaseClient attribute), 26
 alias (x84.bbs.door.Dropfile attribute), 41
 align() (x84.bbs.ansiwin.AnsiWindow method), 58
 alignment (x84.bbs.ansiwin.AnsiWindow attribute), 58
 AnsiWindow (class in x84.bbs.ansiwin), 58
 append() (x84.bbs.pager.Pager method), 51
 apply_environment() (x84.rlogin.ConnectRLogin method), 30
 at_bottom (x84.bbs.lightbar.Lightbar attribute), 47
 at_top (x84.bbs.lightbar.Lightbar attribute), 47
 auth() (x84.bbs.userbase.User method), 57

B

backspace() (x84.bbs.editor.ScrollingEditor method), 43
 backword() (x84.bbs.editor.ScrollingEditor method), 44
 banner() (x84.client.BaseConnect method), 27
 banner() (x84.telnet.ConnectTelnet method), 28
 BaseClient (class in x84.client), 26
 BaseConnect (class in x84.client), 27
 BaseServer (class in x84.server), 25
 bell (x84.bbs.editor.ScrollingEditor attribute), 44
 BLOCKSIZE_RECV (x84.client.BaseClient attribute), 26
 border() (x84.bbs.ansiwin.AnsiWindow method), 58
 bottom (x84.bbs.pager.Pager attribute), 51
 buffer_event() (x84.bbs.session.Session method), 53
 buffer_input() (x84.bbs.session.Session method), 53

C

callback_cmdopt() (in module x84.bbs.telnet), 64
 calls (x84.bbs.userbase.User attribute), 57
 carriage_returned (x84.bbs.editor.LineEditor attribute), 42

carriage_returned (x84.bbs.editor.ScrollingEditor attribute), 44
 cbreak() (x84.terminal.Terminal method), 35
 CFG (in module x84.bbs.ini), 45
 check_anonymous_user() (in module x84.bbs.userbase), 57
 check_bye_user() (in module x84.bbs.userbase), 57
 check_db() (in module x84.db), 22
 check_local_option() (x84.telnet.TelnetClient method), 28
 check_new_user() (in module x84.bbs.userbase), 57
 check_remote_option() (x84.telnet.TelnetClient method), 28
 check_user_password() (in module x84.bbs.userbase), 57
 check_user_pubkey() (in module x84.bbs.userbase), 57
 clear() (x84.bbs.ansiwin.AnsiWindow method), 59
 client_count() (x84.server.BaseServer method), 25
 client_factory (x84.rlogin.RLoginServer attribute), 31
 client_factory (x84.server.BaseServer attribute), 25
 client_factory (x84.telnet.TelnetServer attribute), 29
 client_factory_kwargs() (x84.server.BaseServer class method), 25
 client_fds() (x84.rlogin.RLoginServer method), 31
 client_fds() (x84.server.BaseServer method), 25
 client_list() (x84.server.BaseServer method), 25
 client_recv() (in module x84.engine), 21
 client_send() (in module x84.engine), 21
 clients (x84.server.BaseServer attribute), 25
 clients_ready() (x84.server.BaseServer method), 25
 close() (x84.bbs.session.Session method), 53
 close() (x84.client.BaseClient method), 26
 comhandle (x84.bbs.door.Dropfile attribute), 41
 comport (x84.bbs.door.Dropfile attribute), 41
 comspeed (x84.bbs.door.Dropfile attribute), 41
 comtype (x84.bbs.door.Dropfile attribute), 41
 connect_factory (x84.rlogin.RLoginServer attribute), 31
 connect_factory (x84.server.BaseServer attribute), 25
 connect_factory (x84.telnet.TelnetServer attribute), 29
 connect_factory_kwargs() (x84.server.BaseServer class method), 25
 connect_time (x84.bbs.session.Session attribute), 53

ConnectRLogin (class in x84.rlogin), 30
ConnectTelnet (class in x84.telnet), 28
content (x84.bbs.editor.ScrollingEditor attribute), 44
content (x84.bbs.pager.Pager attribute), 51
copy() (x84.bbs.dbproxy.DBProxy method), 60
ctime (x84.bbs.msgbase.Msg attribute), 62
current_script (x84.bbs.session.Session attribute), 54

D

DBHandler (class in x84.db), 22
DBProxy (class in x84.bbs.dbproxy), 59
deactivate() (x84.client.BaseClient method), 26
debug_option() (in module x84.telnet), 29
decode_pipe() (in module x84.bbs.output), 49
delete() (x84.bbs.userbase.Group method), 56
delete() (x84.bbs.userbase.User method), 57
determine_encoding() (in module x84.terminal), 36
disconnect() (in module x84.bbs.session), 56
Disconnected, 61
do_poll() (in module x84.msgpoll), 33
Door (class in x84.bbs.door), 40
DOSDoor (class in x84.bbs.door), 39
Dropfile (class in x84.bbs.door), 40
duration (x84.bbs.session.Session attribute), 54
duration() (x84.client.BaseClient method), 26

E

echo() (in module x84.bbs.output), 49
email (x84.bbs.userbase.User attribute), 57
emit() (x84.bbs.ipc.IPCLogHandler method), 46
encode_pipe() (in module x84.bbs.output), 49
encoding (x84.bbs.session.Session attribute), 54
env (x84.server.BaseServer attribute), 26
eol (x84.bbs.editor.ScrollingEditor attribute), 44
erase() (x84.bbs.answin.AnsiWindow method), 59
erase_border() (x84.bbs.answin.AnsiWindow method), 59

F

filename (x84.bbs.door.Dropfile attribute), 41
fileno() (x84.client.BaseClient method), 26
find_server() (in module x84.engine), 21
find_tty() (in module x84.terminal), 36
find_user() (in module x84.bbs.userbase), 58
fixate() (x84.bbs.editor.ScrollingEditor method), 44
fixate() (x84.bbs.lightbar.Lightbar method), 47
flush_event() (x84.bbs.session.Session method), 54
flush_queue() (in module x84.terminal), 36
footer() (x84.bbs.answin.AnsiWindow method), 59
format_origin_line() (in module x84.bbs.msgbase), 62
from_cp437() (in module x84.bbs.output), 49
fullname (x84.bbs.door.Dropfile attribute), 41

G

get() (x84.bbs.dbproxy.DBProxy method), 60
get() (x84.bbs.userbase.User method), 57
get_connect_data() (x84.rlogin.ConnectRLogin method), 30
get_database() (in module x84.db), 22
get_db_filepath() (in module x84.db), 22
get_db_func() (in module x84.db), 22
get_db_lock() (in module x84.db), 22
get_digestpw() (in module x84.bbs.userbase), 58
get_fail2ban_function() (in module x84.fail2ban), 33
get_ini() (in module x84.bbs.ini), 45
get_input() (x84.client.BaseClient method), 26
get_last_msg_id() (in module x84.msgpoll), 33
get_msg() (in module x84.bbs.msgbase), 62
get_networks() (in module x84.msgpoll), 34
get_origin_line() (in module x84.bbs.msgbase), 62
get_servers() (in module x84.engine), 21
get_session_output_fds() (in module x84.engine), 21
get_terminals() (in module x84.terminal), 36
get_token() (in module x84.msgpoll), 34
get_user() (in module x84.bbs.userbase), 58
getch() (in module x84.bbs.session), 56
getch() (x84.terminal.Terminal method), 35
getsession() (in module x84.bbs.session), 56
getterminal() (in module x84.bbs.session), 56
gosub() (in module x84.bbs.session), 56
Goto, 61
goto() (in module x84.bbs.session), 56
goto() (x84.bbs.lightbar.Lightbar method), 47
Group (class in x84.bbs.userbase), 56
group_add() (x84.bbs.userbase.User method), 57
group_del() (x84.bbs.userbase.User method), 57
groups (x84.bbs.userbase.User attribute), 57

H

handle (x84.bbs.userbase.User attribute), 57
handle_lock() (in module x84.engine), 21
has_key() (x84.bbs.dbproxy.DBProxy method), 60
hidden (x84.bbs.editor.LineEditor attribute), 42

I

idle (x84.bbs.session.Session attribute), 54
idle() (x84.client.BaseClient method), 26
index (x84.bbs.lightbar.Lightbar attribute), 47
init() (in module x84.bbs.ini), 45
init_bbs_ini() (in module x84.bbs.ini), 45
init_keystrokes() (x84.bbs.editor.LineEditor method), 42
init_keystrokes() (x84.bbs.editor.ScrollingEditor method), 44
init_keystrokes() (x84.bbs.lightbar.Lightbar method), 47
init_keystrokes() (x84.bbs.pager.Pager method), 51
init_keystrokes() (x84.bbs.selector.Selector method), 63

init_log_ini() (in module x84.bbs.ini), 45
 init_term() (in module x84.terminal), 36
 init_theme() (x84.bbs.answin.AnsiWindow method), 59
 init_theme() (x84.bbs.editor.LineEditor method), 43
 init_theme() (x84.bbs.editor.ScrollingEditor method), 44
 init_theme() (x84.bbs.lightbar.Lightbar method), 47
 inkey() (x84.terminal.Terminal method), 35
 input_filter() (x84.bbs.door.Door method), 40
 input_filter() (x84.bbs.door.DOSDoor method), 40
 input_ready() (x84.client.BaseClient method), 26
 IPLogHandler (class in x84.bbs.ipc), 46
 IPCStream (class in x84.bbs.ipc), 46
 is_a_tty (x84.terminal.Terminal attribute), 35
 is_active() (x84.client.BaseClient method), 26
 is_scrolled (x84.bbs.editor.ScrollingEditor attribute), 44
 is_sysop (x84.bbs.userbase.User attribute), 57
 isinview() (x84.bbs.answin.AnsiWindow method), 59
 iswithin() (x84.bbs.answin.AnsiWindow method), 59
 items() (x84.bbs.dbproxy.DBProxy method), 60
 iteritems() (x84.bbs.dbproxy.DBProxy method), 60
 iterkeys() (x84.bbs.dbproxy.DBProxy method), 60
 itervalues() (x84.bbs.dbproxy.DBProxy method), 60

K

kbhit() (x84.terminal.Terminal method), 35
 keys() (x84.bbs.dbproxy.DBProxy method), 60
 kill_session() (in module x84.terminal), 36
 kind (x84.client.BaseClient attribute), 26

L

last_index (x84.bbs.lightbar.Lightbar attribute), 47
 last_input_time (x84.bbs.session.Session attribute), 54
 lastcall (x84.bbs.userbase.User attribute), 57
 lastcall_date (x84.bbs.door.Dropfile attribute), 41
 lastcall_time (x84.bbs.door.Dropfile attribute), 41
 left (x84.bbs.selector.Selector attribute), 63
 Lightbar (class in x84.bbs.lightbar), 46
 LineEditor (class in x84.bbs.editor), 42
 list_msgs() (in module x84.bbs.msgbase), 62
 list_privmsgs() (in module x84.bbs.msgbase), 62
 list_tags() (in module x84.bbs.msgbase), 62
 list_users() (in module x84.bbs.userbase), 58
 LISTEN_BACKLOG (x84.server.BaseServer attribute), 25
 location (x84.bbs.door.Dropfile attribute), 41
 location (x84.bbs.userbase.User attribute), 57
 log_db_cmd() (in module x84.db), 23

M

main() (in module x84.engine), 21
 main() (in module x84.msgpoll), 34
 make_root_logger() (in module x84.bbs.ipc), 46
 margin_amt (x84.bbs.editor.ScrollingEditor attribute), 44
 margin_pct (x84.bbs.editor.ScrollingEditor attribute), 44

MAX_CONNECTIONS (x84.server.BaseServer attribute), 25
 max_length (x84.bbs.editor.ScrollingEditor attribute), 44
 members (x84.bbs.userbase.Group attribute), 56
 move_down() (x84.bbs.lightbar.Lightbar method), 47
 move_down() (x84.bbs.pager.Pager method), 51
 move_end() (x84.bbs.lightbar.Lightbar method), 47
 move_end() (x84.bbs.pager.Pager method), 51
 move_home() (x84.bbs.lightbar.Lightbar method), 47
 move_home() (x84.bbs.pager.Pager method), 51
 move_left() (x84.bbs.selector.Selector method), 63
 move_pagedown() (x84.bbs.lightbar.Lightbar method), 47
 move_pageup() (x84.bbs.lightbar.Lightbar method), 47
 move_pgdown() (x84.bbs.pager.Pager method), 51
 move_pgup() (x84.bbs.pager.Pager method), 51
 move_right() (x84.bbs.selector.Selector method), 63
 move_up() (x84.bbs.lightbar.Lightbar method), 47
 move_up() (x84.bbs.pager.Pager method), 51
 moved (x84.bbs.answin.AnsiWindow attribute), 59
 Msg (class in x84.bbs.msgbase), 61

N

name (x84.bbs.userbase.Group attribute), 56
 name_option() (in module x84.telnet), 30
 NETHACK_KEYSET (in module x84.bbs.lightbar), 48
 node (x84.bbs.door.Dropfile attribute), 41
 node (x84.bbs.session.Session attribute), 54
 numcalls (x84.bbs.door.Dropfile attribute), 41

O

on_naws() (in module x84.terminal), 36
 output_filter() (x84.bbs.door.Door method), 40
 output_filter() (x84.bbs.door.DOSDoor method), 40

P

pageheight (x84.bbs.door.Dropfile attribute), 41
 Pager (class in x84.bbs.pager), 50
 parity (x84.bbs.door.Dropfile attribute), 41
 parse_connect_data() (x84.rlogin.ConnectRLogin method), 30
 parse_dbevent() (in module x84.db), 23
 parse_public_key() (in module x84.bbs.userbase), 58
 password (x84.bbs.door.Dropfile attribute), 41
 password (x84.bbs.userbase.User attribute), 57
 PC_KEYSET (in module x84.bbs.editor), 43
 pid (x84.bbs.session.Session attribute), 54
 poll_event() (x84.bbs.session.Session method), 54
 poll_network_for_messages() (in module x84.msgpoll), 34
 poller() (in module x84.msgpoll), 34
 pop() (x84.bbs.dbproxy.DBProxy method), 60
 popitem() (x84.bbs.dbproxy.DBProxy method), 60
 pos() (x84.bbs.answin.AnsiWindow method), 59

- position (x84.bbs.editor.ScrollingEditor attribute), 44
- position (x84.bbs.lightbar.Lightbar attribute), 47
- position (x84.bbs.pager.Pager attribute), 51
- position_last (x84.bbs.pager.Pager attribute), 51
- prepare_message() (in module x84.msgpoll), 34
- process_keystroke() (x84.bbs.editor.LineEditor method), 43
- process_keystroke() (x84.bbs.editor.ScrollingEditor method), 44
- process_keystroke() (x84.bbs.lightbar.Lightbar method), 47
- process_keystroke() (x84.bbs.pager.Pager method), 51
- process_keystroke() (x84.bbs.selector.Selector method), 63
- proxy_iter() (x84.bbs.dbproxy.DBProxy method), 60
- proxy_iter_session() (x84.bbs.dbproxy.DBProxy method), 60
- proxy_method() (x84.bbs.dbproxy.DBProxy method), 60
- proxy_method_direct() (x84.bbs.dbproxy.DBProxy method), 60
- proxy_method_session() (x84.bbs.dbproxy.DBProxy method), 60
- publish_network_messages() (in module x84.msgpoll), 34
- pull_rest() (in module x84.msgpoll), 34
- push_rest() (in module x84.msgpoll), 34
- ## Q
- queue_for_network() (x84.bbs.msgbase.Msg method), 62
- quit (x84.bbs.editor.LineEditor attribute), 43
- quit (x84.bbs.editor.ScrollingEditor attribute), 44
- quit (x84.bbs.lightbar.Lightbar attribute), 48
- quit (x84.bbs.pager.Pager attribute), 52
- quit (x84.bbs.selector.Selector attribute), 63
- ## R
- raw() (x84.terminal.Terminal method), 36
- RE_ANSI_COLOR (in module x84.bbs.output), 48
- RE_REPWITH_CLEAR (x84.bbs.door.DOSDoor attribute), 39
- RE_REPWITH_NONE (x84.bbs.door.DOSDoor attribute), 39
- read() (x84.bbs.editor.LineEditor method), 43
- read() (x84.bbs.editor.ScrollingEditor method), 45
- read() (x84.bbs.lightbar.Lightbar method), 48
- read() (x84.bbs.pager.Pager method), 52
- read() (x84.bbs.selector.Selector method), 63
- read_event() (x84.bbs.session.Session method), 54
- read_events() (x84.bbs.session.Session method), 54
- recv_modem() (in module x84.bbs.modem), 61
- recv_ready() (x84.client.BaseClient method), 27
- recv_ready() (x84.rlogin.RLoginClient method), 31
- recv_ready() (x84.telnet.TelnetClient method), 28
- refresh() (x84.bbs.editor.LineEditor method), 43
- refresh() (x84.bbs.editor.ScrollingEditor method), 45
- refresh() (x84.bbs.lightbar.Lightbar method), 48
- refresh() (x84.bbs.pager.Pager method), 52
- refresh() (x84.bbs.selector.Selector method), 63
- refresh_quick() (x84.bbs.lightbar.Lightbar method), 48
- refresh_row() (x84.bbs.lightbar.Lightbar method), 48
- refresh_row() (x84.bbs.pager.Pager method), 52
- register_tty() (in module x84.terminal), 36
- release() (x84.bbs.dbproxy.DBProxy method), 60
- remaining_mins (x84.bbs.door.Dropfile attribute), 41
- remaining_secs (x84.bbs.door.Dropfile attribute), 42
- remove() (x84.bbs.userbase.Group method), 56
- request_do_binary() (x84.telnet.TelnetClient method), 28
- request_do_env() (x84.telnet.TelnetClient method), 28
- request_do_naws() (x84.telnet.TelnetClient method), 29
- request_do_sga() (x84.telnet.TelnetClient method), 29
- request_do_ttype() (x84.telnet.TelnetClient method), 29
- request_env() (x84.telnet.TelnetClient method), 29
- request_ttype() (x84.telnet.TelnetClient method), 29
- request_will_binary() (x84.telnet.TelnetClient method), 29
- request_will_echo() (x84.telnet.TelnetClient method), 29
- request_will_sga() (x84.telnet.TelnetClient method), 29
- resize() (x84.bbs.ansiwin.AnsiWindow method), 59
- resize() (x84.bbs.door.Door method), 40
- resize() (x84.bbs.door.DOSDoor method), 40
- right (x84.bbs.selector.Selector attribute), 63
- RLoginClient (class in x84.rlogin), 30
- RLoginServer (class in x84.rlogin), 31
- ropen() (in module x84.bbs.output), 49
- run() (x84.bbs.door.Door method), 40
- run() (x84.bbs.door.DOSDoor method), 40
- run() (x84.bbs.session.Session method), 55
- run() (x84.client.BaseConnect method), 27
- run() (x84.db.DBHandler method), 22
- run() (x84.rlogin.ConnectRLogin method), 30
- run() (x84.telnet.ConnectTelnet method), 28
- runscript() (x84.bbs.session.Session method), 55
- ## S
- SAUCE_FONT_MAP (in module x84.bbs.output), 48
- save() (x84.bbs.door.Dropfile method), 42
- save() (x84.bbs.msgbase.Msg method), 62
- save() (x84.bbs.userbase.Group method), 56
- save() (x84.bbs.userbase.User method), 57
- SB_MAXLEN (x84.telnet.TelnetClient attribute), 28
- script_module (x84.bbs.session.Session attribute), 55
- script_path (x84.bbs.session.Session attribute), 55
- scroll_amt (x84.bbs.editor.ScrollingEditor attribute), 45
- scroll_pct (x84.bbs.editor.ScrollingEditor attribute), 45
- ScrollingEditor (class in x84.bbs.editor), 43
- securitylevel (x84.bbs.door.Dropfile attribute), 42
- selected (x84.bbs.lightbar.Lightbar attribute), 48
- selected (x84.bbs.selector.Selector attribute), 63

- selection (x84.bbs.lightbar.Lightbar attribute), 48
 selection (x84.bbs.selector.Selector attribute), 63
 Selector (class in x84.bbs.selector), 62
 send() (x84.client.BaseClient method), 27
 send() (x84.rlogin.RLoginClient method), 31
 send_event() (x84.bbs.session.Session method), 55
 send_modem() (in module x84.bbs.modem), 61
 send_ready() (x84.client.BaseClient method), 27
 send_ready() (x84.rlogin.RLoginClient method), 31
 send_str() (x84.client.BaseClient method), 27
 send_unicode() (x84.client.BaseClient method), 27
 send_unicode() (x84.telnet.TelnetClient method), 29
 send_urgent_str() (x84.rlogin.RLoginClient method), 31
 Session (class in x84.bbs.session), 52
 SESSION (in module x84.bbs.session), 52
 session (x84.terminal.Terminal attribute), 36
 session_recv() (in module x84.engine), 21
 session_send() (in module x84.engine), 22
 set_keyboard_decoder() (x84.terminal.Terminal method), 36
 setdefault() (x84.bbs.dbproxy.DBProxy method), 60
 show_traceback (x84.bbs.session.Session attribute), 55
 showart() (in module x84.bbs.output), 49
 shutdown() (x84.client.BaseClient method), 27
 socket_recv() (x84.client.BaseClient method), 27
 socket_recv() (x84.telnet.TelnetClient method), 29
 spawn_client_session() (in module x84.terminal), 37
 START_BLOCK (x84.bbs.door.DOSDoor attribute), 39
 start_process() (in module x84.terminal), 37
 stime (x84.bbs.msgbase.Msg attribute), 62
 stopped (x84.client.BaseConnect attribute), 27
 SYNCTERM_FONTMAP (in module x84.bbs.output), 49
 syncterm_setfont() (in module x84.bbs.output), 50
 sysopname (x84.bbs.door.Dropfile attribute), 42
 systemname (x84.bbs.door.Dropfile attribute), 42
- ## T
- tap_input (x84.bbs.session.Session attribute), 55
 tap_output (x84.bbs.session.Session attribute), 55
 TelnetClient (class in x84.telnet), 28
 TelnetOption (class in x84.telnet), 29
 TelnetServer (class in x84.telnet), 29
 Terminal (class in x84.terminal), 35
 TerminalProcess (class in x84.terminal), 36
 threads (x84.server.BaseServer attribute), 26
 TIME_NEGOTIATE (x84.rlogin.ConnectRLogin attribute), 30
 TIME_NEGOTIATE (x84.telnet.ConnectTelnet attribute), 28
 TIME_POLL (x84.rlogin.ConnectRLogin attribute), 30
 TIME_POLL (x84.telnet.ConnectTelnet attribute), 28
 time_used (x84.bbs.door.Dropfile attribute), 42
 TIME_WAIT_STAGE (x84.telnet.ConnectTelnet attribute), 28
 timeago() (in module x84.bbs.output), 50
 title() (x84.bbs.ansiwin.AnsiWindow method), 59
 to_dict() (x84.bbs.session.Session method), 55
 to_localtime() (in module x84.bbs.msgbase), 62
 to_utctime() (in module x84.bbs.msgbase), 62
 toggle() (x84.bbs.selector.Selector method), 63
 translate_ttype() (in module x84.terminal), 37
 TTYPE_UNDETECTED (x84.client.BaseClient attribute), 26
- ## U
- unregister_tty() (in module x84.terminal), 37
 update() (x84.bbs.dbproxy.DBProxy method), 60
 update() (x84.bbs.editor.ScrollingEditor method), 45
 update() (x84.bbs.lightbar.Lightbar method), 48
 update() (x84.bbs.pager.Pager method), 52
 User (class in x84.bbs.userbase), 56
 user (x84.bbs.session.Session attribute), 55
 usernum (x84.bbs.door.Dropfile attribute), 42
- ## V
- values() (x84.bbs.dbproxy.DBProxy method), 60
 visible_bottom (x84.bbs.lightbar.Lightbar attribute), 48
 visible_bottom (x84.bbs.pager.Pager attribute), 52
 visible_content (x84.bbs.lightbar.Lightbar attribute), 48
 visible_content (x84.bbs.pager.Pager attribute), 52
 visible_height (x84.bbs.ansiwin.AnsiWindow attribute), 59
 visible_width (x84.bbs.ansiwin.AnsiWindow attribute), 59
 vitem_idx (x84.bbs.lightbar.Lightbar attribute), 48
 vitem_shift (x84.bbs.lightbar.Lightbar attribute), 48
- ## W
- width (x84.bbs.editor.LineEditor attribute), 43
 willfit() (x84.bbs.ansiwin.AnsiWindow method), 59
 write() (x84.bbs.ipc.IPCStream method), 46
 write() (x84.bbs.session.Session method), 56
- ## X
- x84.bbs.ansiwin (module), 58
 x84.bbs.dbproxy (module), 59
 x84.bbs.door (module), 39
 x84.bbs.editor (module), 42
 x84.bbs.exception (module), 61
 x84.bbs.ini (module), 45
 x84.bbs.ipc (module), 46
 x84.bbs.lightbar (module), 46
 x84.bbs.modem (module), 61
 x84.bbs.msgbase (module), 61
 x84.bbs.output (module), 48

x84.bbs.pager (module), 50
x84.bbs.selector (module), 62
x84.bbs.session (module), 52
x84.bbs.telnet (module), 64
x84.bbs.userbase (module), 56
x84.client (module), 26
x84.db (module), 22
x84.engine (module), 21
x84.fail2ban (module), 33
x84.msgpoll (module), 33
x84.rlogin (module), 30
x84.server (module), 25
x84.telnet (module), 27
x84.terminal (module), 35
xferprotocol (x84.bbs.door.Dropfile attribute), 42
xpadding (x84.bbs.ansiwin.AnsiWindow attribute), 59

Y

ypadding (x84.bbs.ansiwin.AnsiWindow attribute), 59